Performance Issues in the Design of Hierarchical-ring and Direct Networks for Shared-memory Multiprocessors

by

Govindan Ravindran

A thesis submitted in conformity with the requirements
for the degree of Doctor of Philosophy
Graduate Department of Electrical and Computer Engineering
University of Toronto

Abstract

Performance Issues in the Design of Hierarchical-ring and Direct Networks for Shared-memory Multiprocessors

Govindan Ravindran
Doctor of Philosophy
Graduate Department of Electrical and Computer Engineering
University of Toronto
1998

This dissertation explores performance issues in the design of interconnection networks for shared-memory multiprocessors. In particular, it considers low-dimensional direct (e.g., mesh and tori) and hierarchical-ring networks, and studies issues in topology, buffer management, switching, routing and flow control. The performance evaluation is primarily done by simulating the target systems at the register transfer level on a cycle-by-cycle basis using both synthetic and real (SPLASH-2) workloads.
The contributions of the dissertation include:

- the first extensive performance study specifically for shared-memory multiprocessor interconnection networks. Prior research focused primarily on distributed memory systems. It is important to consider shared-memory systems separately because of the different class of traffic they must support.

- the first extensive simulation study of hierarchical-ring networks. Topologies that perform well are derived, and various cut-through switching techniques, such as wormhole, virtual cut-through, cell switching, are evaluated under both blocking and non-blocking flow control policies.

- a comprehensive comparative performance study of 2D mesh, 2D tori and hierarchical-ring networks. It is shown that the hierarchical-ring networks outperform the mesh and tori networks for system sizes smaller than 64 processors.

- a novel deadlock free routing technique for wormhole switched hierarchical-ring networks using a virtual channel approach.

- a novel priority-based network design that uses dynamic virtual channels and prioritized link arbitration with priority inheritance that results in improved system throughput and can be used to support multiple classes of traffic.

Acknowledgements

First, I would like to thank Professor Michael Stumm for his supervision and guidance throughout. Among many valuable things I learned in his association include attention to detail, simplicity in presentation, and the virtue of hard work.

I would like to thank the members of my examination committee, Professors Zvonko Vranesic, Ken Sevcik, Paul Chow, Anindo Banerjea, and Charles Clarke for their careful reading of my dissertation. Their valuable suggestions helped to vastly improve the quality of presentation in this dissertation.

I would like to thank Prof. Ted Szymanski of McGill University for accepting to serve as the external examiner in my committee. His critical evaluation of my dissertation and numerous insights are highly valuable.

I would also like to thank Prof. Carl Hamacher of Queen's University for his many inputs on my research during the past several years.

I am blessed with great parents and a wonderful wife. My wife, Vijaya, deserves special mention for her patience and support throughout my graduate study. Also, special thanks for her several perl scripts that I used extensively for output analysis of simulation results.

I greatly appreciate the help and support of my colleagues, Kulki, Sudarsan, Jaseemuddin, and Kiran who provided me great company and invaluable wisdom during my stay in the department.

My friends Singar, Ram, Kala, Dev, and Vani enriched my social life during my stay in this wonderful city, Toronto. For sure, I will miss their company in the years to come.

Finally, I would like to thank the Government of Canada for their financial support through a Commonwealth Doctoral Fellowship.

Contents

List of Tables

List of Figures

Chapter 1
Introduction

The design of a multiprocessor's interconnection network is important, as it significantly affects the performance and cost of the multiprocessor. In this dissertation, we present the results of a performance study of issues in the design of multiprocessor interconnection networks in general, and low-dimensional direct and hierarchical-ring networks in particular. Although the performance of wide-area and local-area networks has been studied extensively, as has the performance of interconnection networks for distributed memory machines, few studies of interconnection networks for shared-memory machines exist. This work, therefore, fills a void in existing research that will become increasingly important as shared-memory multiprocessors become more widespread.

Multiprocessor interconnection networks differ from wide-area and local-area networks significantly in that they have (*i*) regular topologies allowing for simple algorithmic routing as opposed to table-driven routing, (*ii*) smaller node sizes that do not allow the buffering of a larger number of packets at the nodes, (*iii*) bit-parallel links, (*iv*) higher speed, and (*v*) error-free short transmission links. These network characteristics demand switching, routing, and flow-control techniques that are different from those used in communication networks.

Shared-memory systems also differ from distributed memory systems in a significant way. In a shared-memory system, the processors are more tightly coupled than in a distributed memory system, where a programmer sees a collection of separate computers that communicate only by sending explicit messages to one another. Memory is accessible to all processors in a shared-memory system. Thus, interprocessor communication in a shared-memory system is indirect, where the producer typically leaves the data in the memory for the consumer to later fetch. Also, communication in a shared-memory system requires no intervention on the part of a run-time library or operating system. This results in low start-up costs for communication, whereas in a distributed memory system, access to the network is typically managed by system software, resulting in high start-up costs. As a result, shared-memory multiprocessing produces unique traffic patterns that change dynamically and that have short variable sized packets (mostly

Shared-memory Networks	Distributed Memory Networks
Shorter message size	Longer message size
Packet switched	Packet switched or Circuit switched
Dynamic traffic pattern	Static traffic pattern
Network is the primary source of overhead	System software and network interfaces are primary sources of overhead

Table 1.1: The traffic and network characteristics of shared-memory and distributed memory multiprocessors.

bi-modal). These and other characteristics are tabulated in Table 1.1. More importantly, it is not clear how factors such as cache line size (that influences the largest packet size), node buffer size, and locality in the memory access pattern affect system performance.

In our work, we have focussed on low-dimensional direct and hierarchical-ring networks, such as 2-dimensional meshes and tori. In a *direct network*, every node in the network is a processing node and is connected directly to a small set of *neighboring* nodes typically using a very regular topology. Hierarchical rings belong to the class of networks known as *hybrid networks*, where some processing nodes are connected to each other by point-to-point links, while other processing nodes are connected through switching nodes. While low-dimensional direct networks are currently popular in research and commercial environments, hierarchical-ring networks present a viable alternative to direct networks, mainly because:

1. the hierarchical-ring network is attractive due to its low dimension resulting in simple routers, wider link widths and shorter packet switching times.

2. the physical locality of hierarchical rings blends naturally with that of computational locality often exhibited in parallel programs,

3. the hierarchical-ring structure allows efficient broadcasting capabilities useful for implementing cache coherence [29] and multicast protocols [92],

Table 1.2 categorizes multiprocessor networks according to different classes and programming models. The areas where a lot of work has already been done are marked by $\surd$. Those areas that have not been studied much are addressed in this dissertation (with the exception of distributed memory hybrid networks). There have been only a few performance studies of hierarchical-ring and direct networks for shared-memory multiprocessors [28, 38, 39, 42, 47, 54, 66, 95]. Holliday and Stumm [42] analyzed hierarchical-ring multiprocessor networks under cell

	Direct Networks	Indirect Networks	Hybrid Networks
Shared-memory	?	√	?
Distributed Memory	√	√	?

Table 1.2: Categorizing multiprocessor interconnection networks. into direct, hybrid, and indirect networks for shared-memory and distributed memory multiprocessing.

switching and parametric simulations. Hamacher and Jiang [39] presented an analytical queuing network model for hierarchical-ring networks and derived optimal configurations. Jaseemuddin [47] proposed and evaluated bidirectional, ring-connected multiprocessors as an alternative to hierarchical, ring-based multiprocessors. Oi and Ranganathan [66] presented a performance analysis of bidirectional, ring-connected shared-memory multiprocessors. Zhang and Yan [95] compared NUMA and COMA hierarchical-ring architectures using analytical models. Kumar and Bhuyan [54] evaluated the performance impact of virtual channels in 2-dimensional torus connected shared-memory multiprocessors. All these studies assume specific switching, routing, and flow-control techniques and do not study issues such as scalability, buffer management, or priorities. Also, they do not study network performance under program-driven simulations (with the exception of Jaseemuddin [47] and Kumar and Bhuyan [54]) and do not present an extensive performance comparison of direct and hierarchical-ring topologies.

Low-dimensional direct networks have been studied extensively for distributed memory multiprocessors, however, they are being used in today's larger commercial and research shared-memory multiprocessors. Recent commercial and research shared-memory multiprocessors that use direct networks include SGI Origin [16], Flash [55], Cray T3E [81], and Sequent NUMA-Q [84]. The recent systems have followed the success of some earlier direct network shared-memory multiprocessors that include Alewife [4], and DASH [57]. Examples of some research in shared-memory multiprocessors based on hierarchical-ring networks include Hector [93] and NUMAchine [92].

This dissertation tries to extend previous work with an extensive study of various performance issues in the design of low-dimensional direct and hierarchical-ring networks in the context of shared-memory multiprocessing. We study the impact of topology, switching, routing, flow-control, network buffer size, and priorities on the performance of such networks. We also present an extensive performance comparison of hierarchical-ring and three different direct networks for shared-memory multiprocessing.

	Topology	Switching	Buffer management	Flow-control	Routing	Priority
Hierarchical-ring networks	Chap 5	Chap 6	Chap 6	Chap 6	Chap 7	not studied
2D direct networks	not interesting	Chap 6	Chap 6	Chap 6 & Chap 7	studied by others	Chap 8

Table 1.3: Design issues and organization of dissertation.

1.1 Contributions of Dissertation

A rough design space of multiprocessor networks is presented in Table 1.3. The table indicates which chapters address which areas. In particular, the dissertation makes the following specific contributions:

- *Comprehensive performance study of shared-memory multiprocessor interconnection networks*: We believe this study constitutes the first comprehensive performance study of low-dimensional direct and hierarchical-ring interconnection networks for shared-memory systems.

- *Comparative performance study*: We present a detailed comparative performance evaluation of low-dimensional direct and hierarchical-ring networks using both synthetic workload and program-driven simulations. We show, in particular, that hierarchical-ring networks perform better than 2-dimensional direct networks for system sizes up to 64 processors at low request rates either when there is locality in the memory access pattern or for large cache line sizes.

- *Topology*: We derive several high throughput and low latency hierarchical-ring topologies and study the impact of locality (in the memory access pattern) and constant bisection bandwidth constraints on system performance.

- *Switching techniques and buffer management*: We study the performance of several cut-through switching techniques for hierarchical-ring networks under both blocking and non-blocking flow-control policies. These include wormhole, virtual cut-through, and cell switching techniques. We show that increasing the buffer size in wormhole routers can result in significant performance improvement, but that too large a buffer size sometimes hurts performance. We also show that while non-blocking cell switching is a good choice for hierarchical-ring networks, it requires large buffers to minimize the number of packets dropped. On the other hand, wormhole switching requires virtual channels to

prevent deadlock although it uses less buffer space. We study wormhole switching in low-dimensional direct networks, namely the 2-dimensional mesh, 2-dimensional torus, and the bidirectional rings under blocking flow-control and show that buffered wormhole switching in direct networks results in optimal performance.

- *Routing*: We propose a deadlock free minimal routing technique for wormhole switched hierarchical-ring networks using a virtual channel approach.

- *Priority*: We propose dynamic virtual channels and present the design of prioritized direct networks that also use priority inheritance and priority-based link arbitration. We show how such prioritized networks can be used to improve system throughput and to support multiple classes of traffic.

1.2 Organization of Dissertation

Chapter 2 presents, in tutorial style, background necessary for understanding this dissertation. We introduce most of our terminology there. Chapter 3 gives an overview of hierarchical-ring and direct network based multiprocessor systems, their network interfaces, and it presents the methodology we use in the performance evaluation of these networks. Chapter 4 presents a strong motivation for considering hierarchical rings: we compare the performance of hierarchical-ring, 2-dimensional mesh, 2-dimensional torus, and single bi-directional networks, and show that hierarchical-ring systems are competitive from a performance point of view.

Since hierarchical-ring networks are highly configurable, topology issues of such networks are studied in Chapter 5. In Chapter 6 we discuss switching, buffer management, and flow-control issues in hierarchical-ring and direct networks. Since we only consider deterministic routing protocols, we focus on deadlock freedom in such routing algorithms for hierarchical-ring networks in Chapter 7. Finally, we present the design of a novel priority direct network with dynamic virtual channels in Chapter 8.

Chapter 2
Issues in the Design of Multiprocessor Networks

In a direct network, every node in the network is a processing node and is connected to its neighboring nodes by direct links. In a hierarchical-ring network, there are processing nodes and there are switching nodes; the processing nodes are connected in some cases directly but in some cases through switching nodes. The communication architecture of a hierarchical-ring or direct network is characterized, among other things, by the topology, and how it does switching, routing and flow-control [19, 74]. *Topology* of a network defines how the nodes are interconnected by channels. *Switching* is the mechanism by which a router removes a packet from one of its input links and places it on an output link, thereby allocating channels and buffers to the packet as it travels through the network. *Routing* is the selection of a path for a packet from its source to its destination node. *Flow-control* is the mechanism that regulates the transmission of packets in a network; the flow-control unit in a router, for example, informs neighboring nodes to stop sending packets to avoid buffer overflows.

In this chapter, we introduce different topologies and different techniques for switching, routing, and flow-control. We show how these factors can affect system performance.

2.1 Terminology

In this section, we introduce the terminology essential for understanding this dissertation, using the simple direct network shown in Figure 2.1. This two-dimensional network consists of nine nodes connected by *links* or *channels*.[1] Each node contains a processing module that includes a processor and local memory, and it contains a *router* that connects the node to the network. Each link connects a pair of nodes and consists essentially of a set of wires, most of which are used to transfer bit-parallel data while the others are used for control. The links in the figure are *bidirectional*, but in other networks they may be *unidirectional*. The number of wires in a link that transmit data in any one direction is defined as the *link* or *channel width*.

[1] We will use the terms *links* and *channels* interchangeably throughout this dissertation.

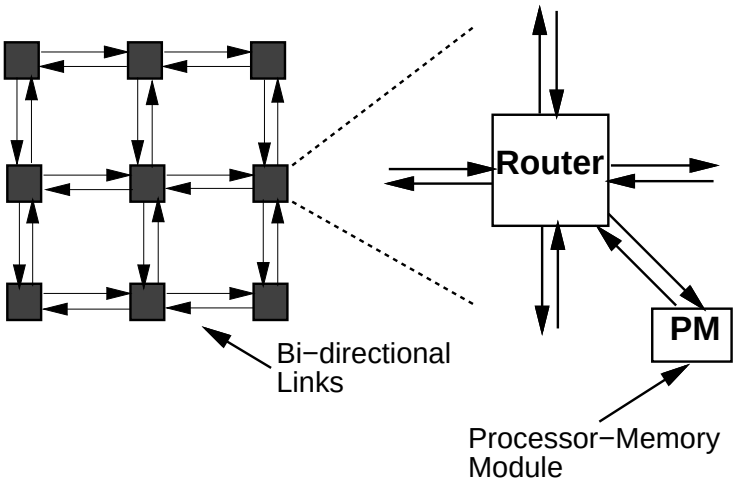

Figure 2.1: A 2-dimensional 3×3 mesh connected network with bidirectional links between nodes.

In a shared-memory multiprocessor, if a processor accesses memory that is not local to the node, then the hardware implicitly generates and sends a memory request message to the node with the target memory. Messages are usually sent as packets over the network. A *packet* contains a header with all the information required to deliver it to the correct destination, such as the address of the target node. The packet is the smallest unit for allocating channels. In *wormhole switching*, a popular switching technique we will describe later in this chapter, every packet is broken into a number of *flits*,[2] and buffering, forwarding and flow-control are performed at the flit level. The flits of a packet are sent consecutively over a channel, so the flits of two packets are never interleaved. Flits themselves are actually transmitted a *phit* (physical transfer unit) at a time, which is typically the size of the link width, something that can be transferred in a single clock cycle. Figure 2.2 illustrates how a message is partitioned into packets, into flits and then into phits.

Routers connect the processing nodes to the network and manage the links to the neighboring nodes. A router normally contains communication processing logic as well as a set of buffers to hold flits. It handles all communication related tasks to allow computation (by the processor) and communication at the node to take place concurrently. The communication tasks include relaying packets in the direction of the packets' destination nodes (switching and routing), preventing buffer overflow (flow-control), removing packets from the network if destined for the local node, and injecting packets from the local node into the network (switching). In addition, some routers also assemble packets into messages and disassemble messages into packets. How

[2] A flit as defined in [18] is the smallest unit of information that a node may refuse or accept. Thus, in a simple implementation, the number of bits in a flit may correspond to the number of data lines in a single physical channel.

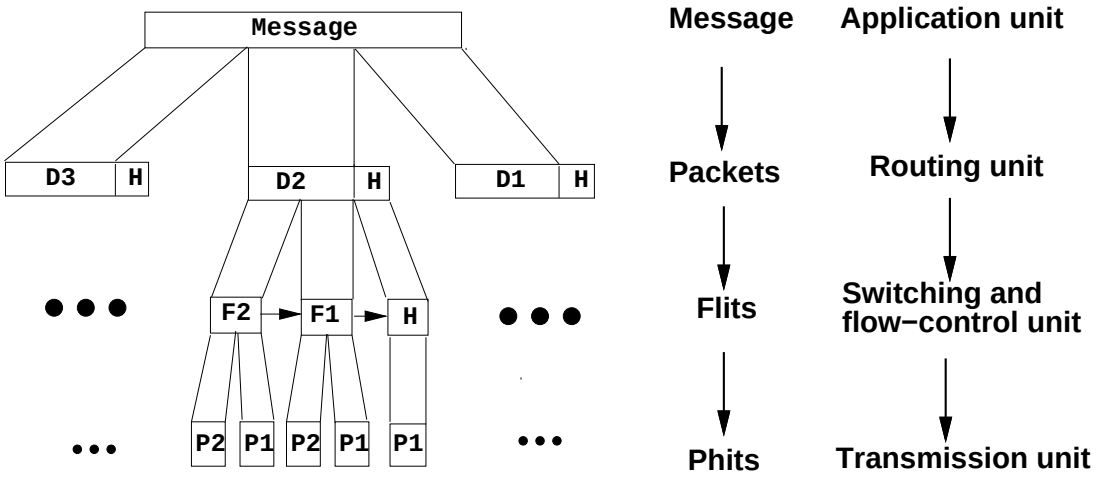

Figure 2.2: The figure illustrates the message, packets, flits and phits in processor networks.

effectively a router carries out these functions is critical to system performance.

The router has network input and output links to each neighboring node and a processor input and output link to the local processing module. The number of network input/output links is dependent on the network dimension; the higher the dimension of a network, the higher the number of links at a node. The *out-degree* of a node is the number of links leaving the node, whereas the *in-degree* is defined as the number of links arriving at the node. For the topologies we consider, the in-degrees and out-degrees are equal and we will just refer to the *degree of a node*. There are buffers associated with input links which are referred to as input buffers.[3] The *torus routing chip*, developed at the California Institute of Technology, was one of the first general purpose routers for a direct network-based multiprocessor [17].

A packet originates at a *source node* and is consumed at a *destination node*.[4] The *distance* between two nodes is the minimum number of links in any path between the nodes. *Network diameter* is the maximum internode distance in the network [30]. It is analogous to the diameter of a circle (which is the maximum distance between any two points on the circle). Higher dimensional networks normally have a smaller network diameter than lower dimensional networks. The average distance traveled by a packet is dependent on the application traffic pattern; the average delay of a packet will be a function of the distance it must travel.

Contention for a network resource (a link or a buffer) occurs when two or more packets compete for the same resource. Network link or buffer contention leads to one or more packets being blocked until the resources they need are freed. The *network latency* is the total elapsed time between when a packet is injected into the network and when it is received by the des-

[3]Buffers may also be associated with output links, or with both input and output links.
[4]In the case of multicasting, there is more than one destination node.

tination node. It is the sum of the internode distance and the amount of time the packet is blocked in the network waiting for resources. The multiprocessor *system throughput* is defined as the total number of memory requests completed per unit time.

Bisection width is defined as the minimum number of links that must be removed to partition the network into equal halves [3]. *Bisection bandwidth* is the total bandwidth of the bisection links. The significance of the bisection width is that if memory request destinations are selected at random, then half of the requests must traverse the bisection channels. Therefore, for an application that exhibits poor memory access behavior, bisection bandwidth of a network becomes critical to the application's performance when the bisection links become congested.

Ideally a direct network should be scalable to a large size simply by adding more nodes to the network. *Scalability* of a network is constrained by various factors such as constant bisection bandwidth, which becomes a performance bottleneck when the network size is increased [71]. Also, while it is desirable to increase the size of a network by adding an arbitrary number of nodes, most direct networks allow only fixed-size increments, the exception being one-dimensional rings.

2.2 Topology

2.2.1 Direct Networks

Most common direct network topologies can be broadly classified into two general classes namely, the n-dimensional meshes and the k-ary n-cubes. In both cases, n is the dimension of the network. A k-ary n-cube has k nodes in each dimension and, by definition, has *wraparound* channels that connect the first and the last node in any dimension. It is the wraparound channels that differentiates k-ary n-cubes from n-dimensional meshes. As a consequence of having wraparound channels, all nodes in a k-ary n-cube have the same number of neighbors, namely $2n$ for $k > 2$ and n for $k = 2$, making it a *symmetric* interconnection network [76]. An n-dimensional mesh has normally the same number of nodes along each dimension but does not possess wraparound channels. As a result, the number of neighbors for a node depends on the position of the node, making it an *asymmetric* interconnection network. A k-ary n-cube has a smaller diameter than an n-dimensional mesh because of the wraparound channels. A single ring is a special case of k-ary n-cube where $n = 1$, and a binary hypercube is a special case of both an n-dimensional mesh and a k-ary n-cube where $k = 2$.

A direct network router might be implemented on a single or small set of VLSI chips, or it might be implemented on a printed circuit board (PCB). In either case, packaging constraints limit the number of wires to the neighboring nodes and the processing module, placing a bound

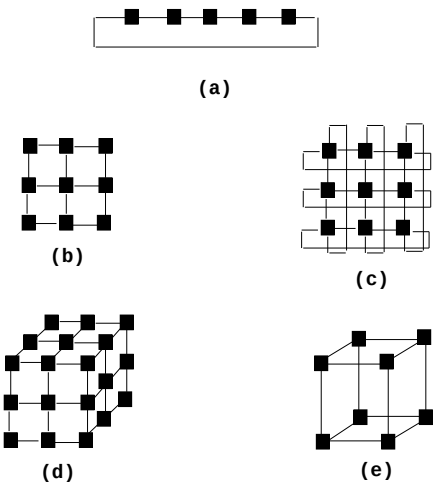

Figure 2.3: Popular direct topologies: (a) 5-ary 1-cube (1-dimensional ring), (b) 2-dimensional mesh (3×3), (c) 3-ary 2-cube (2-dimensional torus), (d) 3-dimensional mesh ($3\times3\times3$), (e) 2-ary 3-cube (3-dimensional hypercube).

on the I/O bandwidth available for communication links. Assuming that the I/O bandwidth of a router is divided equally among the links to its neighbors, adding more links to a node (and thus increasing the dimension of the network), decreases the I/O bandwidth available per link proportionately. Since increasing the number of links also reduces the diameter (and thus the average internode distance) of the network, a trade-off exists between per link bandwidth and diameter in the choice of the number of links per node [31].

Figure 2.3 illustrates different direct topologies. Earlier systems, such as hypercubes, tended to use higher dimensional networks and therefore had a smaller network diameter at the expense of lower per link bandwidth. More recent multiprocessor systems use lower dimensional direct networks, after it was shown that lower dimension networks (with at most 3 dimensions) generally perform better than their higher dimension counterparts [3, 20]. The most common direct network topology in use today is the 2-dimensional (2D) mesh because of its low degree, which permits efficient layouts and construction with standard components (see Figure 2.1). An interesting variation is the cube-connected cycle, where each node of an n dimensional binary hypercube is replaced with a ring of n nodes [69]. Each ring node connects to one of the n links incident on the vertex. As a result, the node degree remains fixed at three, irrespective of the hypercube dimension.

Figure 2.1 shows a 2-dimensional mesh with bidirectional links between nodes. Throughout this study, we will illustrate the various switching, routing and flow-control techniques using this topology.

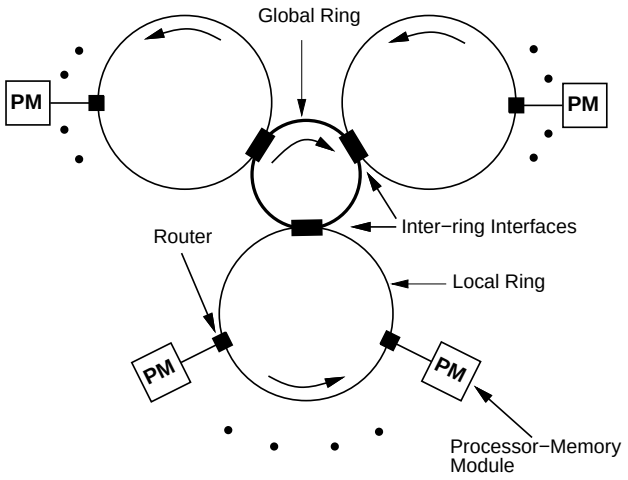

Figure 2.4: A 2-level hierarchical-ring connected network.

2.2.2 Hierarchical-ring Networks

A hierarchical-ring network falls under the category of *hybrid networks* in that it is only partially a direct network. In a two-level hierarchical-ring network, several direct single ring networks, referred to as *local rings*, are connected by a *global ring* consisting of switches or inter-ring interfaces [92, 93]. The global ring is itself an ensemble of switches connected by a direct network topology but contains no processing nodes. The fact that there are two levels in the network hierarchy is transparent to the processing nodes, and it is possible to extend the hierarchy to more than two levels; for example, a three-level hierarchical-ring network consists of a global ring connecting multiple two-level hierarchical-ring networks. Figure 2.4 shows a two-level hierarchical-ring network where the global and local network nodes are connected by unidirectional rings [92].

Hierarchical-ring networks have a diameter that grows more slowly with system size than comparably sized direct networks, indicating that they might scale well. On the other hand, they have a constant bisection bandwidth that can limit their scalability. However, it is reasonable to expect that when mapping parallel applications onto a multiprocessor network it will usually be possible to do so such that the tasks that communicate frequently are placed close to one another so that most communication will be local to one cluster. If there is locality in the communication pattern of the applications, then hierarchical-ring networks can scale to a larger number of nodes [73].

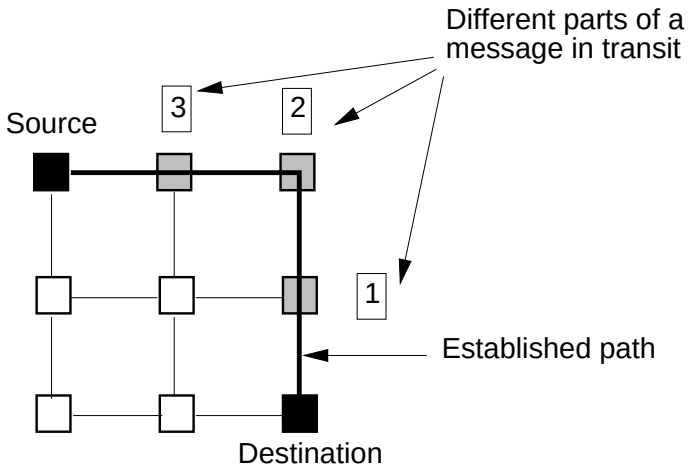

Figure 2.5: Circuit switching in a 2-dimensional mesh network. The source and destination nodes are shown in dark and the intermediate nodes are shown in grey. The established path is illustrated by the bolder links connecting the source and destination. A path between the source and the destination node is established and is not released until the entire message is received by the destination node.

2.3 Switching Techniques

Switching deals with allocating buffers and channels to packets. The three main switching techniques include circuit switching, store-and-forward, and cut-through switching. In this section, we briefly discuss each of these switching techniques. They can be applied to both hierarchical-ring and direct networks even though we always use a 2-dimensional mesh network for illustration. We cover cut-through switching in more depth in this section, and we consider only cut-through switching in our performance evaluation in subsequent chapters, because 1) it exhibits lower network latency, and 2) is used in many recent networks such as those of the Cray's T3D [80], T3E [81], and SGI Origin [56].

2.3.1 Circuit Switching

In circuit switching a header containing the destination address is first sent through the network to build a path from the source node to the destination node [49]. At each intermediate node on the path, a connection is established between an input port and an output port. Which output port will be chosen is determined by the routing algorithm. As the header progresses through the network it reserves the links over which it is being transmitted. When the destination node receives the header, it sends an acknowledgment back to the source indicating that a direct connection between the source and the destination has been established, thereby allowing the message transfer to commence. This prior reservation of links makes it unnecessary to buffer

(portions of) the message at any intermediate node when it is in transit. The path is released either by the last byte of the message as it passes through each node along the path or by an acknowledgment that is sent by the destination node when it receives the last byte. Figure 2.5 shows an example of circuit switching in a 2-dimensional mesh network.

The transmission time of a message of length L over d hops takes $3d + L/W$ cycles, where W is the width of the communication channels. It takes $2d$ time units for a header to establish a path (with the acknowledgment) and d time units for the first byte of the message to reach the destination and L/W time units thereafter for the remainder of the message. When, L, the length of the message, dominates in the above expression, it results in message transmission time that is largely independent of the distance between source and destination nodes.

The main disadvantage of circuit switching is that it under-utilizes network bandwidth by reserving a set of links for each transfer, making them unavailable for other transmissions even when they are not currently being used. Consequently, modern shared-memory multiprocessor networks do not use circuit switching, although some of the earlier distributed memory multiprocessor networks used circuit switching [45, 65].

2.3.2 Store-and-forward Switching

Store-and-forward switching [50] differs from circuit switching in that no path is established prior to the transfer of data. It is the protocol of choice in data communication networks. In a store-and-forward network, the unit of data transfer is a packet and a message is broken down into one or more packets. A node accepts the header of a packet only when it can buffer the entire packet, and it does not forward a transit packet to a neighboring node until it receives the entire packet. Figure 2.6 illustrates store-and-forward switching in a 2-dimensional mesh network. Cosmic Cube [83] and Intel's iPSC-1 [65] are some earlier systems that used store-and-forward switching.

The network latency for store-and-forward switching is $d(L/W)$, where d is the number of hops, L is the length of a message (consisting of one or more packets including the header), and W is the channel width. Store-and-forward switching suffers from the following drawbacks:

- The latency of store-and-forward networks is proportional to the distance between the source and destination nodes and is therefore dependent on the diameter of the network. This makes store-and-forward switching expensive in low-dimensional networks that have large diameters.

- Each store-and-forward node must have buffers large enough to store entire packets. Buffers of this size require large silicon area in routers.

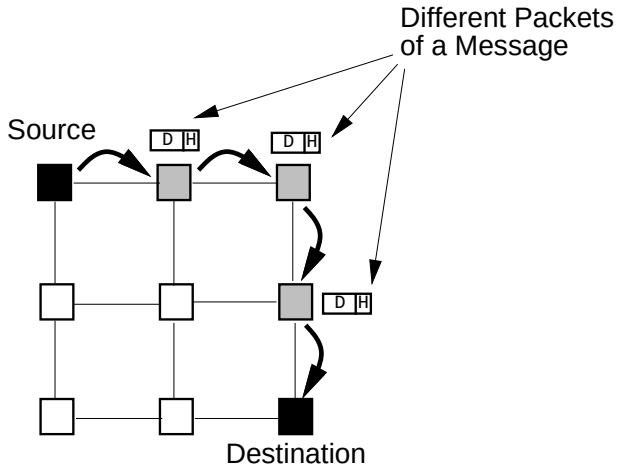

Figure 2.6: Store-and-forward switching in a 2-dimensional mesh network. The source and destination nodes are shown in darker shades and the intermediate nodes are shown in lighter shades. Entire packets hop from one node to another in the path between the source and the destination node. The intermediate nodes between the source and the destination receive packets in their entirety before forwarding them to the next node.

2.3.3 Cut-through Switching

Cut-through switching is an improvement over both store-and-forward and circuit switching in the following two ways: 1) when the header of a packet arrives at a cut-through switched node, it is forwarded to a neighboring node without necessarily waiting for the entire packet to arrive, and 2) the packet header reserves a path as it traverses the links while the packet tail releases the path after it passes through. Similar to store-and-forward switching, a message is divided into one or more packets where the packet is the unit of data transfer. A packet is in turn divided into a number of flits. As flits are forwarded, a packet may be spread out over multiple links, and a packet is sometimes referred to as a *worm* in this context. Since only the head flit of a packet contains the routing information, it is essential that the flits of a packet not to be interleaved with flits of another packet. The head flit of a packet acquires network resources (links and buffers) as it proceeds through the network, while the tail flit frees them.

There are two important variations of cut-through switching, namely *virtual cut-through* and *wormhole* switching. They both send packets as a sequence of flits with the header flit containing the routing and sequencing information, but they differ in how they handle blocked packets. Virtual cut-through switching, first introduced by Kermani and Kleinrock [51], buffers packets in their entirety when they are blocked. Thus when the header of a packet becomes blocked at a node,[5] the remainder of the packet will continue to be transmitted to the node

[5]The packet may be blocked at a node when either the output link it requires is busy transmitting another packet or the input buffer at the next node is full.

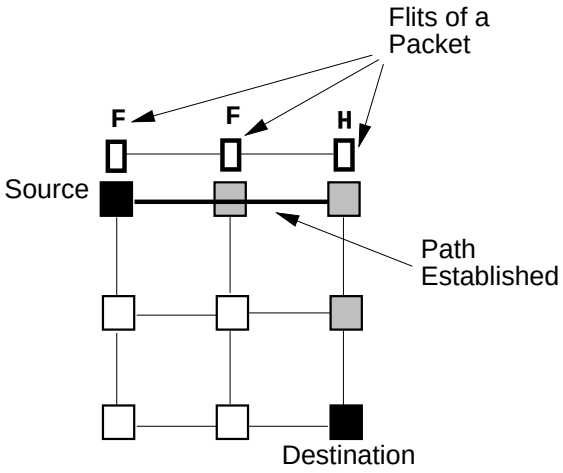

Figure 2.7: Wormhole switching in a 2-dimensional mesh network. The header flit establishes a path and the body flits follow. The reserved links along the path are shown.

until the entire packet has arrived. This requires large buffer spaces in the routers (similar to store-and-forward switching) capable of buffering entire packets.

In wormhole switching, a blocked packet may span multiple nodes, residing in the flit buffers of those nodes (see Figure 2.7). When a packet header cannot move forward it is blocked in place and continues to hold the resource it just acquired. When the local buffers become full, flow control will prevent the neighboring node from transmitting further flits of the packet over the incoming link. This can possibly cause the neighboring router buffers to fill as well, and flow-control will propagate further back. A blocked packet can thus span multiple nodes. This, in turn, can cause the blockage of other packets. Under heavier load conditions, a single "hot-spot" node may cause *tree saturation* where contention for the hot-spot can back propagate to affect other traffic that has no need to reach the hot-spot node [40, 67, 68].

In the absence of contention, network latency in cut-through switched networks is given by $d + L/W$, where d is the number of hops, L is the length of a message (consisting of one or more packets including the header), and W is the data channel width. When L dominates in the above expression, the path length d will not significantly affect the network latency. As a result, the latency of cut-through switched networks is less dependent on network diameter than store-and-forward networks. Cut-through switching is thus a sensible choice for both low and high dimensional networks. Figure 2.8 compares the communication latency of cut-through switching with that of store-and-forward switching, in the case where no blocking occurs. It is obvious that cut-through switching can significantly reduce network latency compared to store-and-forward switching when there is no contention.

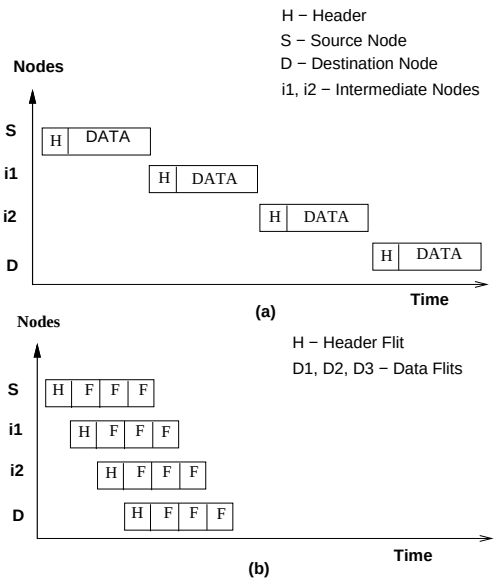

Figure 2.8: Latencies of different switching techniques in the case where no blocking occurs: a) store-and-forward switching and b) cut-through switching.

Wormhole vs. Virtual Cut-through

Both wormhole and virtual cut-through switching block a packet when it cannot move forward, the former buffering it across multiple nodes and the latter buffering it in its entirety in a single router. However, it is not clear which switching technique is superior, and both have their advantages and disadvantages [85]. At low network loads, both switching schemes behave similarly because link contention occurs infrequently. At high loads close to network saturation, virtual cut-through switching exhibits higher throughput and lower latency but it requires larger buffers at the routers, capable of storing entire packets. Virtual cut-through switching performs better at high loads, because the probability of a packet blocking is high in this situation, and since blocked packets are stored in their entirety in routers, the network links are still available for transmitting other packets going to other destinations. This allows higher average link utilization and thus higher network throughput.

In contrast, wormhole networks block packets across nodes, making links unavailable for other use, preventing their bandwidth to be fully exploited. Nevertheless, wormhole switching is attractive in that it requires only a few flit buffers at the routers, since it does not need to buffer entire packets.[6] This reduced buffer requirement makes wormhole routers less expensive and faster [13]. Thus, in choosing between the two schemes, there is a trade-off between performance and cost. There are some who argue however, that virtual cut-through will become prevalent in the near future, because with advances in VLSI technology, large buffers can be integrated

[6] Traditional wormhole routers use only single-flit buffers.

into a router at reasonable cost and also the propagation delay associated with large buffers can be reduced significantly [26].

To take advantage of inexpensive and faster wormhole routers yet address the low throughput problem at high loads, a number of alternatives have recently been proposed recently:

1. *hybrid switching* switches from wormhole to virtual cut-through at high loads by selectively buffering entire packets [85],

2. *buffered wormhole switching* improves throughput by increasing the buffer size at the routers to more than just a few flits, thereby reducing significantly the number of links a packet can block [73].[7]

3. *wave switching* combines circuit switching and wormhole switching, whereby circuit switching is used between nodes that are going to communicate frequently, while wormhole switching is used to transmit packets for which circuit switching is not efficient [24].

In this dissertation we propose buffered wormhole switching and will show in Chapter 6 that this can reduce latency and improve system throughput in both hierarchical-ring and direct networks.

2.3.4 Cell Switching

An important variation of cut-through switching is cell switching [5, 42, 43, 72, 92, 93]. In *cell switching*, packets are divided into equi-sized cells that are routed independently (see Figure 2.9). In this sense it is the same as virtual cut-through switching on a per cell basis. Each cell contains its own routing information: the first cell of a packet carries the full target memory address, while the remaining cells of the packet only identify the destination node. Note that there is no need for sequencing information in a cell if we assume a deterministic routing protocol. The fact that the target node address exists in each cell allows the cells to be routed independently, but it adds overhead to the size of the cells. For example, a 128 processor network requires 7 bits to address each processor and 7 more bits to identify the source node in order to distinguish between cells from different source nodes to the same destination node. This amounts to a total of 14 bits of extra overhead per cell which translates into about 11% if we assume a 128-bit cell size. Note that there is no need to identify cells of different packets from the same source node if cells arrive in order at the destination node.

An advantage of cell switching is that it does not require the buffering of entire packets, yet blocked packets do not block links. Hence, cell routers can be fast and still inexpensive

[7] Increasing the buffer size beyond the largest worm size results in diminishing returns in network throughput, however.

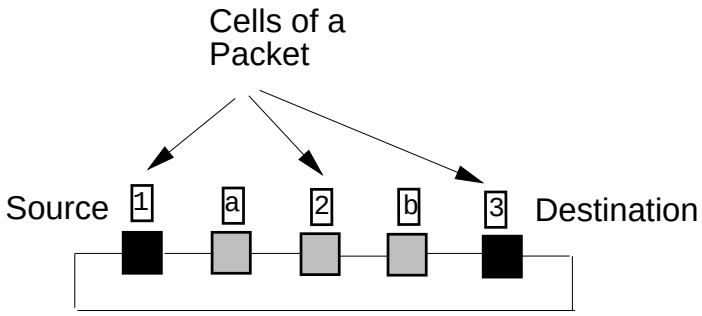

Figure 2.9: Cell switching in an 1-dimensional ring connected network. The cells of a packet can be interleaved with the cells of another packet. The links between nodes are not reserved for an entire packet and are released after a cell is forwarded.

to build. For example, in one-dimensional networks, links can be acquired and then released in the same clock cycle for the transmission of a single cell (assuming cell size is equal to the phit size), and an incoming transit cell can always be transmitted on the outgoing link without being buffered [92]. We will show in Chapter 6 that cell switching is more effective than either wormhole or virtual cut-through switching in hierarchical-ring networks, especially when combined with non-blocking flow-control (described in Section 2.6) [72]. It should be noted that in a single ring, the cell switching is same as the slotted ring protocol [5, 43]. We also propose and evaluate in Chapter 6 a hybrid cell-wormhole switching for hierarchical-ring networks.

2.4 Routing Techniques

Routing determines the path selected by a packet to reach its destination. Routing is critical to network performance and a large amount of research has been done on this topic [7, 14, 18, 19, 21, 23, 25, 26, 27, 34, 35, 46, 53, 60, 62, 80, 89]. The regular topologies typically used for multiprocessor direct or hierarchical networks permit *algorithmic routing*, as opposed to routing based on tables. In *distributed algorithmic routing*, the path selection for a packet is distributed across the nodes: when the packet arrives at a node, the router decides along which link to forward the packet according to a routing algorithm. In *source routing*, the entire path for a packet is decided by the source node itself. Source routing does not allow alternate path selections to accommodate faulty links or to avoid a heavily congested area. Also source routing normally requires a larger packet header to accommodate the specification of the route. For these reasons, source routing is not normally used in modern multiprocessor networks.

Distributed algorithmic routing techniques are common in multiprocessor networks and can be classified as either deterministic or adaptive. In *deterministic routing*, the entire route is

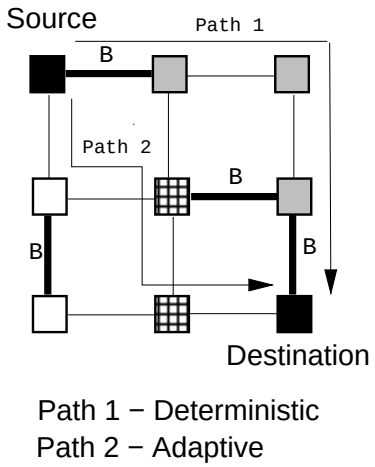

Figure 2.10: The figure illustrates deterministic and adaptive routing in a 2-dimensional mesh connected network. 'B' represents unavailable links due to blocking. Path 1 is chosen by a deterministic routing algorithm while path 2 is chosen by an adaptive routing algorithm. It is seen that adaptive routing can route around blocked nodes whenever possible thus improving the throughput of the network significantly.

determined by the source and destination node addresses alone. The Intel Paragon [6], MIT J-machine [23, 64], and Cray T3D [80] all use deterministic routing. *Adaptive routing*, on the other hand, exploits the fact that there is more than one path between any source and destination node pair (in a multi-dimensional network), and bases its decision on which output link to forward a packet to on such factors as present network conditions and the distance from the destination node (see Figure 2.10). If the choice of output link is always guaranteed to lead to the shortest path to the destination node, then the routing is said to be *minimal*. Also, one can further differentiate between fully adaptive and partially adaptive routing techniques. In a *fully-adaptive* router, the set of legal output links for a packet includes all possible output links the packet can take to reach its destination; hence, by definition a fully-adaptive router cannot be minimal. A *partially-adaptive* router, on the other hand, takes into account only a subset of all possible output links.

In the subsequent chapters we only consider minimal, deterministic, distributed algorithmic routing. Hence, we focus on this class of routing in the rest of this section, but for interested readers we do give some pointers for adaptive routing schemes.

2.4.1 Deterministic Routing

Livelock, starvation, and deadlock are the three major issues that any routing algorithm must address. A routing algorithm that guarantees forward progress of each packet, where every hop the packet makes takes it a step closer to its destination, is said to be *livelock free*. Minimal,

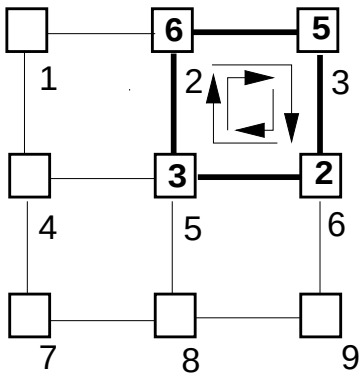

Figure 2.11: The figure shows a simple example of a deadlock in a 4-node-cycle involving nodes 2, 3, 5, and 6. The buffers of each of these nodes is full with a packet, destined for nodes 6, 5, 3, and 2, respectively. The destination node numbers for packets are shown inside the nodes where they are buffered, whereas the present node numbers are shown outside the corresponding nodes.

deterministic routing by definition is livelock free. In non-minimal routing, on the other hand, the routing algorithm must be carefully designed to avoid livelock; otherwise a packet could continuously be routed in a cycle such that it does not reach its destination. A simple solution to prevent livelock is to adopt an age-based priority scheme, where older packets are routed along minimal paths [62].

Starvation occurs when a node is permanently blocked from injecting messages into the network. This may happen when there is heavy transit traffic. A solution to prevent starvation based on the message injection rate of nodes is proposed in [62]. This solution requires extra handshaking lines and additional hardware in routers. Another solution is to use injection tokens in the network, allowing a source node to inject a message only after it consumes one injection token [53]. Since starvation did not occur in any of our simulations, we did not resort to any explicit starvation prevention techniques, although there is no guarantee that it will never occur.

Deadlock is a condition that occurs when some packets are blocked forever because of full network buffers [18, 46, 60]. Cut-through and store-and-forward switching are susceptible to deadlock, because blocked packets occupy buffers and/or channels (network resources) while requesting other resources resulting in cyclic waits for network resources. Figure 2.11 shows an example of a simple deadlock in a 2-dimensional mesh network involving four nodes 2, 3, 5 and 6 (a 4-node-cycle). The buffers of each node in this cycle are filled with messages destined for the diametrically opposite node. No message can advance towards its destination (under a minimal routing algorithm), so there is a deadlock situation.

The techniques proposed to deal with deadlocks in direct networks fall under two general categories, namely (*i*) deadlock prevention and (*ii*) deadlock detection and recovery. *Deadlock prevention* techniques rely on designing routing algorithms that do not allow deadlock to occur in the first place. *Deadlock detection and recovery* techniques, on the other hand, deal with first detecting and then recovering from deadlocks. We use the deadlock prevention technique in direct networks in our further study and extend it to apply to hierarchical-ring networks in Chapter 6.

One way to prevent deadlock in wormhole (cut-through) networks is to divide a physical channel into a number of *virtual channels* and restrict the assignment of packets to these channels. All virtual channels together share the same physical channel, but each virtual channel has its own set of buffers. This *virtual channel approach* is widely used to prevent deadlocks in wormhole networks, and interestingly it improves throughput [18, 19].

Seitz and Dally proposed a necessary and sufficient condition for a minimal deterministic algorithm to be deadlock-free [18]. For a given interconnection network and its routing function, they define a channel dependency graph and state that the deterministic routing function is deadlock free *if and only if* there are no cycles in the channel dependency graph. As a result, for a routing function to be deadlock free according to this theory, it must restrict the use of channels for routing packets so as to eliminate cycles in the channel dependency graph. Given an interconnection network, we can derive its channel dependency graph as follows. The vertices of the channel dependency graph represent the edges of the interconnection network and a vertex v_1 is connected to a vertex v_2 if the routing algorithm allows a packet to be routed from the channel represented by v_1 to the channel represented by v_2. Figure 2.12 shows a 2-dimensional mesh network and its channel dependency graph for dimension-ordered routing (described below).

For n-dimensional meshes *dimension-ordered* routing is a minimal and deterministic routing algorithm. It routes a packet along the lowest dimension first for as far as it must go, before routing it on the next higher dimension, and so on until the packet reaches its destination. This algorithm is simple to implement, elegant and the most widely used [6, 57, 80]. In a 2-dimensional mesh, for example, each node is represented by a 2-digit radix k number, where k is the number of nodes along a dimension, with the first digit representing the node's position in the first dimension and the second digit representing the node's position in the second dimension. The packet is routed in the first dimension (along the x-axis), until it reaches a node whose subscript matches the destination address in the first position. The packet is then routed along the second dimension. In this case at most one turn is allowed and the turn is from the first dimension to the second dimension. A total of $n - 1$ turns are allowed, in general, for

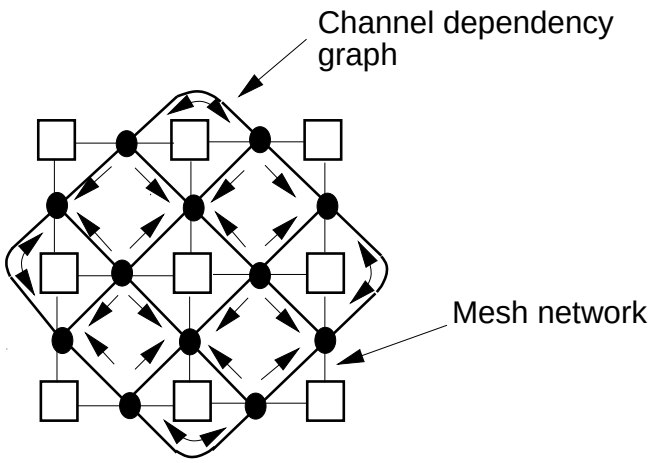

Figure 2.12: A 2-dimensional mesh network and its channel dependency graph (shown by darker lines) for dimension-ordered routing.

n-dimensional meshes. Dimension-ordered routing guarantees deadlock and livelock freedom in n-dimensional meshes by enforcing a strictly monotonic order on the dimensions traversed, but it does so at the cost of adaptivity. It is easy to see that there are no cycles in the channel dependency graph of Figure 2.12.

For k-ary n-cubes, dimension-ordered routing is still minimal and deterministic, but not deadlock-free. In this case a deadlock would involve wraparound channels within a given dimension (since cyclic channel dependencies involving multiple dimensions cannot occur). Seitz and Dally developed deadlock-free minimal deterministic routing protocols for k-ary n-cubes by splitting each physical channel into two virtual channels to prevent cyclic channel dependencies in a given dimension. As an example, consider the 1-dimensional, 5-ary 1-cube (a 5-node ring) and its channel dependency graph shown in Figure 2.13. There is a cycle in the channel dependency graph, so deadlock is possible. Such cycles can be broken by splitting each physical channel along a cycle into two virtual channels known as *high* and *low* virtual channels. A packet currently at node n_i is routed to the high virtual channel if i is smaller than the subscript of the destination node, and into the low virtual channel otherwise. The low virtual channel out of node 0 is not used. It is easy to see that such a routing function has no cycles in the resulting channel dependency graph, as shown in Figure 2.14.

This routing function can be extended to general k-ary n-cubes. Each node of the k-ary n-cube can be identified by an n-digit radix k number, with the ith digit of the number representing the node's position in the ith dimension. Similar to an n-dimensional mesh, we route in the order of dimension, with the most significant dimension first; in each dimension, i, a packet is routed in that dimension until it reaches a node whose subscript matches the

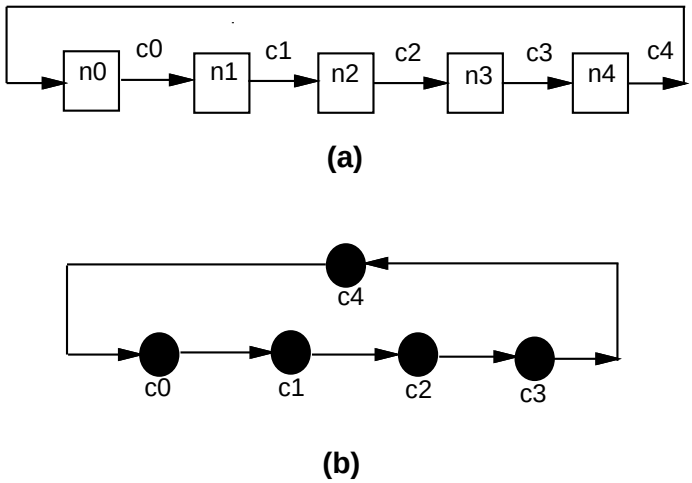

Figure 2.13: A unidirectional 5-node ring topology: a) interconnection network b) channel dependency graph.

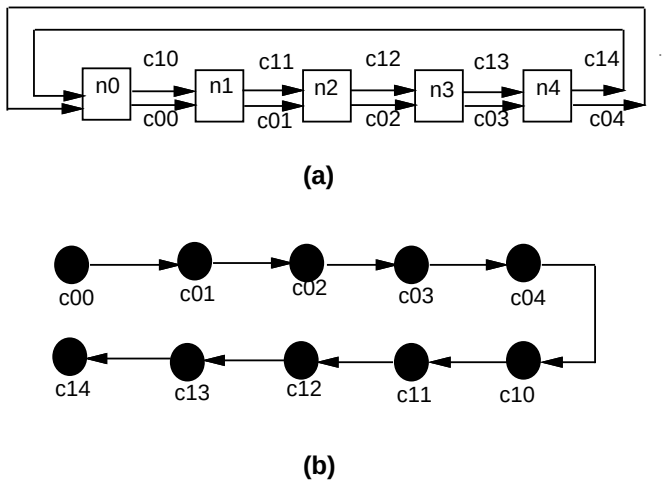

Figure 2.14: A unidirectional 5-node ring topology with virtual channels: a) interconnection network and its b) channel dependency graph.

destination address in the ith position. The message is routed on the high channel if the ith digit of the destination address is greater than the ith digit of the present node's address; otherwise, the message is routed on the low channel.

In general, it is not possible to develop minimal deterministic deadlock free routing algorithms for k-ary n-cubes without virtual channels, with the exception of k-ary 1-cubes for values of $k \leq 4$ [63]. This makes n-dimensional meshes attractive, since dimension ordered routing alone can guarantee deadlock freedom without requiring virtual channels, thus resulting in reduced router complexity [13].

Topology	Deterministic	Turn model	Planar	Linder and Harden
2D Mesh	1	1	2	2
k-ary 2-cube	2	2	4	6
3D Mesh	1	1	3	4
k-ary 3-cube	2	2	6	16
n-D Mesh	1	1	3	2^{n-1}
k-ary n-cube	2	2	6	$2^{n-1}(n+1)$

Table 2.1: Virtual channel requirement for deadlock free routing. With the exception of deterministic routing, all are adaptive routing techniques.

2.4.2 Adaptive Routing

Deterministic routing algorithms are oblivious to dynamic network conditions. Adaptive routing protocols were developed to take advantage of the multiple paths that may exist between source and destination pairs to avoid points of congestion. Adaptive routing can significantly improve network throughput and lower latency when compared to deterministic routing techniques [14, 63], but it requires more complex routers and typically additional virtual channels for deadlock freedom [13].

Seitz and Dally's theory of deadlock free deterministic routing has been extended to adaptive routing [7, 14, 34]. However, it was shown later by Duato that Dally's necessary and sufficient condition for deterministic deadlock-free routing is only a sufficient condition in the case of adaptive routing [25, 26, 27]. Duato shows that by separating virtual channels on a link into deterministic and adaptive classes, wormhole routing can be both adaptive and deadlock free.

Adaptive wormhole routing algorithms that are deadlock-free require virtual channels for both k-ary n-cubes and n-dimensional meshes. The number of virtual channels required per physical channel depends on the degree of adaptivity sought. A fully-adaptive algorithm requires a large number of virtual channels per physical channel when compared to a partially adaptive algorithm. Table 2.1 compares the virtual channel hardware requirements of deadlock-free deterministic routing with three different deadlock-free adaptive routing schemes namely the Turn model [34], planar adaptive routing [14], and the Linder and Harden method [58]. The Turn model, proposed by Glass and Ni, deserves special mention as it does not require more virtual channels than deterministic, yet is adaptive. Instead, it is based on restricting the directions in which packets may turn and prohibits just enough turns to prevent cycles in the channel dependency graph, but allows more turns than the restrictive dimension-ordered routing.

2.5 Flow-control Techniques

When a buffer in a network node has become full and is about to overflow, then there are two options to take. Either packets can be dropped or else incoming traffic from the neighboring nodes can be throttled by sending a flow-control signal to the neighbor so that it can block the next packet from being sent. The first option results in *non-blocking networks*, while the latter results in *blocking networks*.

In non-blocking networks, routers drop packets when they cannot practically be buffered [72, 93]. Non-blocking networks reduce contention for hot-spot resources and prevent tree-saturations at high load conditions [72]. The recovery of a dropped packet is carried out either through negative acknowledgments or through time-outs. Sending a negative acknowledgment back to the source node reduces the time for retransmitting a packet; otherwise the source node has to time-out before retransmitting a dropped packet. Unfortunately time-outs need to be large: to prevent duplicate packets in the network the time-out must be larger than the maximum round-trip delay. A larger number of time-outs therefore increases the average round-trip latency of memory requests, especially at high loads.

A blocking network router blocks a packet when it cannot be buffered at the next neighbor because the buffers there are full. A blocking network thus controls the flow of packets into a node when the node's buffers are full. Flow-control can be either receiver initiated, where the receiving node back propagates a flow-control signal to its neighbor to request it to stop sending packets, or sender initiated, where the transmitting node keeps a count of the number of free flit buffers on neighboring receiving nodes by incrementing the counter whenever a flit is sent to that neighbor and decrementing when the receiver signals that it has removed a flit from the receiving buffer. In the receiver initiated case, there may be packets in transit when a transmitting node receives a flow-control signal; hence, the receiving node should either have enough buffer storage for transit packets or otherwise it must send flow-control signals in advance of its buffers becoming full. An example of a router that incorporates receiver initiated blocking is the Reliable Router from MIT [22]. It asserts or deasserts a *clear-to-send* signal depending on whether there is enough buffer space to receive the next flit. An example of a router that uses sender initiated blocking is the arctic routing chip, also developed at MIT [8].

End-to-end flow-control between the source and destination node is often used in data communication protocols and has the purpose of preventing speed mismatches between the source and destination, thereby avoiding buffer congestion at the receiving node [32]. End-to-end flow control schemes typically have some form of admission control at the source node, where a packet is injected into the network only if the destination is able to accept the packet.

However, end-to-end flow-control is not typically used in multiprocessor networks, except in a primitive form, mainly because of the overhead involved in implementing it. A rare example of a router that supports end-to-end flow-control is the NIFDY chip developed at the University of California, Berkeley [10]. NIFDY uses admission control to perform end-to-end flow-control by restricting the number of outstanding requests a processor can issue to at most one per destination processor.

In this dissertation, we study both blocking and non-blocking flow-control in hierarchical-ring networks, but only blocking flow-control in direct networks since our preliminary simulation results showed that non-blocking flow-control in 2-dimensional direct networks results in poor performance. This is mainly because the longer worms (packet length) result in higher network cycles to drop and recover packets in direct networks.[8] End-to-end flow control is also applied by imposing a maximum limit on the number of outstanding transactions a processor can issue.

2.6 Other Issues

Recently, there has been some interest in providing service guarantees for new applications requiring real-time communications with continuous data types that include audio and video. Kim and Chien [52] proposed a novel queuing and scheduling algorithm for direct networks that guarantees deterministic delay bounds and bandwidth for real-time traffic. J. Rexford, et. al. [77, 78] proposed a router architecture for direct networks that supports multiple classes of traffic that includes best-effort and real-time traffic. Both these techniques assume connection-oriented networks, where a prior connection is established to forward real-time traffic. This under-utilizes the network resources and penalizes best-effort traffic.

In Chapter 7, we propose dynamic virtual channel flow-control to implement a connectionless priority-based direct network. It can be used to either support multiple classes of traffic, as seen in multiprocessor video/transaction servers or improve the throughput of best-effort traffic. We implement such a network using dynamic virtual channels, priority inheritance, and priority based link arbitration.

[8] Under constant router pin constraints, the number of flits of a packet increases with the dimension of the network.

Chapter 3
System Description, Methodology, and Workloads

In this chapter, we describe the hierarchical-ring and direct networks we consider in this dissertation. As representative direct networks, we consider 2-dimensional meshes, 2-dimensional tori, and bidirectional rings. We use wormhole and cell switching techniques for hierarchical rings, while our predominant switching technique for direct networks is wormhole. We use the minimal, deterministic, dimension-ordered routing without virtual channels for 2-dimensional meshes and two virtual channels per physical channel to prevent deadlock involving wrap-around channels in the torus and the bidirectional ring. We consider blocking flow-control for direct networks, and both blocking and non-blocking flow-control for hierarchical-ring networks.

Our approach to performance evaluation is primarily through simulations, although we at times also use semi-empirical analytical models. Synthetic workload models are complemented by program-driven simulations with programs chosen from the SPLASH-2 suite [94]. Synthetic workload models allow us to accurately control the network load, and to study network behavior for a range of operating points. This makes it feasible to find important network parameters such as the maximum achievable throughput and the processor request rate at network saturation. On the other hand, a set of real applications subject a network to some realistic operating points; however, it would be difficult to predict the network behavior under other operating points from just the real ones. Another advantage of using synthetic workloads is that the number of transactions that need to be issued by each processor to obtain reliable system performance measures is much smaller than the number of transactions needed when simulating the execution of real application programs.

3.1 System Description

Low-dimensional meshes and tori are currently popular for use as interconnection backplanes in large-scale shared-memory multiprocessors. A number of commercial products use these types of networks [16, 80, 81], as do a number of experimental and research systems [55, 57]. This is

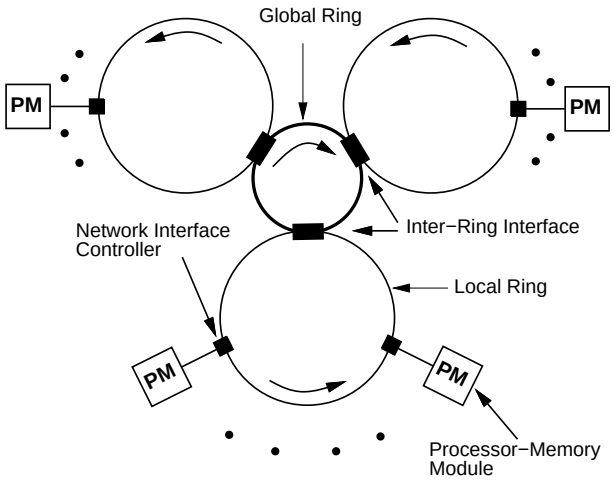

Figure 3.1: A hierarchical-ring system with two levels.

partly because of their perceived scalability characteristics and partly because routers for these types of networks exist and hence it is relatively easy to build such systems using off-the-shelf routers and processors. Nevertheless, we will show in Chapter 4 that hierarchical unidirectional ring-based multiprocessors are, from the point of view of performance, interesting alternatives to two-dimensional direct networks.

Figures 3.1 and 3.2 show shared-memory multiprocessor systems containing a number of processing modules, in the former case connected by a two-level hierarchy of unidirectional rings [92], and in the latter case connected by a square 2-dimensional bidirectional mesh.[1] Each processing module (PM) contains a processor, a local cache and a portion of the main memory. In the case of hierarchical rings, all processing modules are connected to lowest level rings, which we also refer to as *local rings*. A *global ring* connects several of these local rings. The channel width (data path) of the ring is assumed to be 128 bits wide in our study.

For a mesh connected system, a number of variations on the basic topology shown are possible. The connection between each pair of adjacent nodes in a mesh is bidirectional (implemented as two 32-bit wide unidirectional channels[2]). We consider both mesh-connected networks that have no *wrap-around* connections and tori with wrap-around channels. The main differences between a mesh and a torus network are: (1) for deadlock-free, dimension-ordered routing, a mesh network does not require virtual channels, whereas a torus network requires its physical channel to be multiplexed between two virtual channels, (2) the wrap-around channels in a torus reduce the network diameter by a factor of 2 when compared to a mesh network, (3) the bisection

[1] These figures are repeated from Chapter 2 for convenience.

[2] The assumption is valid under constant router pin constraints where a 128-bit wide channel for unidirectional rings reduces to a 32-bit wide channel for 2-dimensional meshes

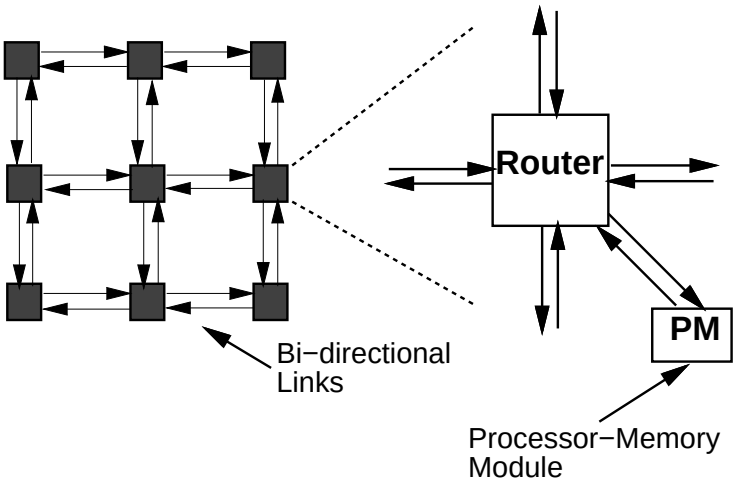

Figure 3.2: A 2-dimensional 3×3 mesh with 9 nodes.

bandwidth of a torus is twice that of a mesh network, again due to the wraparound channels, and (4) all nodes in a torus network have the same number of neighbors making it a regular topology. The bidirectional ring, which we also study, is considered to be a 1-dimensional torus.

We only consider shared-memory multiprocessors in our study. Hence, we assume that both the hierarchical-ring and the direct networks provide a flat, global (physical) address space, and that each PM is assigned a unique contiguous portion of that address space, determined by its location. All processors can transparently access all memory locations in the system. The target memory is determined by the address of the memory being accessed. The memory transactions we consider are cache line reads and writes. Each memory transaction involves a request and response sub-transaction. Local memory accesses do not involve the network. Remote memory accesses require a request packet to be sent to the target memory, followed by a response packet from the target memory to the requesting processor. Packets sent are of variable size and are transferred in flits,[3] bit-parallel, along a unique path through the network. In a hierarchy of rings, a packet containing a remote request whose target memory is in a different ring than its source, first travels up the hierarchy to the level needed to reach the target node, and then descends the hierarchy to the target node where it is removed from the local ring. The target node sends a response packet back to the requesting PM along a similar path.

3.1.1 Hierarchical-ring System Description

For a hierarchical ring, there are two types of network nodes: *Network Interface Controllers* (NIC) connect processing modules (PM) to local rings and *Inter-ring Interfaces* (IRI) connect two rings of adjacent levels. The NIC examines the header of a packet and switches (1) incoming

[3]No distinction is made between a phit (physical transfer unit) and a flit in our study.

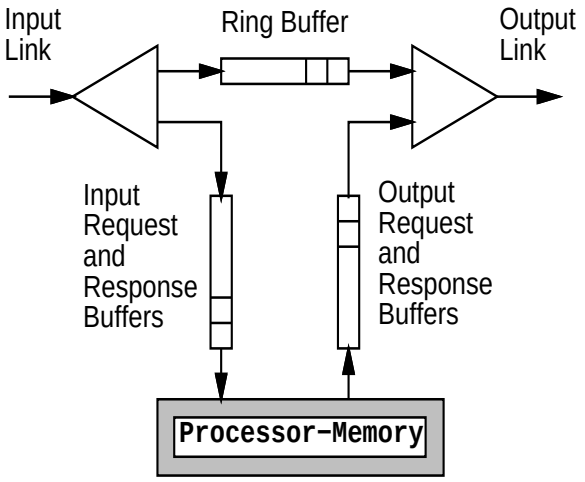

Figure 3.3: A network interface controller (NIC) for hierarchical-ring connected multiprocessor network.

packets from the ring to a PM, (2) outgoing packets from the PM to the ring, and (3) continuing packets from the input link to the output link. The IRI controls the traffic between two rings and is modeled as a 2×2 crossbar switch. Possible implementations of these network nodes are depicted in Figures 3.3 and 3.4.

The NIC has a FIFO *ring buffer* to temporarily store transit packets arriving from the network not destined to the local PM when the output link is currently transmitting another packet from the local PM. If the ring buffer is empty and no packet is currently being transmitted, then an incoming transit packet will be forwarded to the output link directly, bypassing the ring buffer. The NIC also has a FIFO *input buffer* for storing packets destined for the local PM and a FIFO *output buffer* for storing packets originating from the PM destined for nodes elsewhere in the network (see Figure 3.3). Both of these are split into request and response queues (not shown) to avoid deadlocks involving multiple classes of traffic [37]. Priority for transmission to the next node is given to ring packets either waiting in the bypass buffer or having just arrived from the previous node. Otherwise, if there are packets in one of the output queues then priority is given to response packets over request packets. The reason behind this prioritized link arbitration is to minimize the time a packet spends in the network.

Superficially, the NIC model described appears to be similar to the buffer insertion ring access scheme used in ring topology local area networks [44, 91]. There are, however, significant differences. In the cut-through switched ring,

1. blocking flow-control is used to block a packet when it cannot be buffered at the next neighbor because the buffers are full. For example, in a wormhole switched ring with single-flit ring buffers, a blocked (header) flit of a packet will block the following flits, and

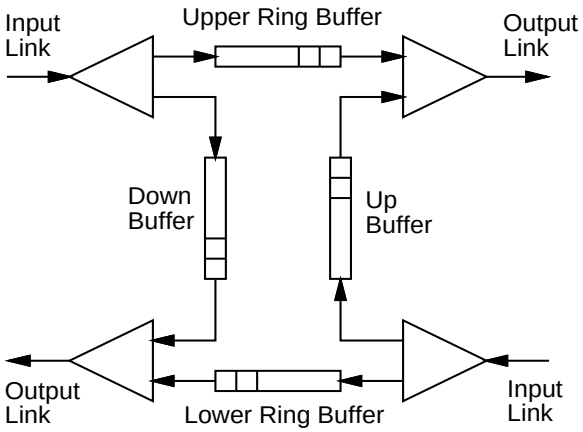

Figure 3.4: An inter-ring interface controller for hierarchical-ring connected multiprocessor network.

when the entire packet becomes blocked, it typically spans multiple nodes.

2. priority is always given to packets in the ring when there is a competing request for an output link from the local PM.

3. small, fixed size insertion buffers are used, and

4. dedicated transmit and receive buffers (input and output buffers) to and from the local PM are used.

An IRI has two ring buffers, one for the lower ring and one for the upper ring. It also has a *down* buffer and an *up* buffer (see Figure 3.4). The down buffer stores packets arriving from the upper ring destined for the lower ring, while the up buffer stores packets arriving from the lower ring destined for the upper ring. Switching takes place independently at the lower and upper ring sides. We use a prioritized link arbitration where priority is given to packets that do not change rings. Arriving transit packets block and are placed in the ring buffer (1) when the output link is in the process of transmitting a packet from the up/down buffer or (2) when packets are already waiting in the ring buffer.

Both the NIC and IRI have flow control units (for blocking networks) that are used to signal upstream neighbors when to stop sending packets. We consider NIC and IRI buffers with sizes ranging from single flit size to multiple cache line sizes large enough to accommodate one or more packets containing cache lines.

We assume that all communications occur synchronously: that is, within a network clock cycle, each NIC can transfer one flit to the next adjacent node (if the link is not being blocked), and receive a flit from the previous node it connects to; an IRI, in one network clock cycle, can transmit and receive a flit on each ring (if there is no blocking).

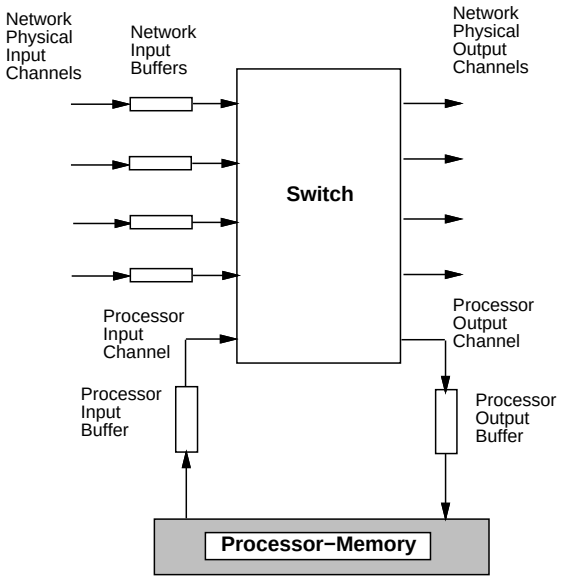

Figure 3.5: A network interface controller for a 2-dimensional mesh or a torus. The input/output links from/to neighboring nodes are referred to as network input/output links and the input/output link from/to local processor-memory module is referred to as processor input/output link. The schematic also shows the network input buffers and processor input/output buffers.

3.1.2 Mesh System Description

In a mesh (torus) connected system, there is only one type of network node, namely the mesh Network Interface Controller (NIC) that connects processing modules to the mesh. A NIC for a 2-dimensional bidirectional mesh or torus is shown schematically in Figure 3.5. The NIC is modeled as a 5×5 crossbar switch with four input/output links from and to its four direct neighbors and one input/output link from and to the local PM. The input links have FIFO buffers to store flits that are blocked in the network. An output buffered NIC can be considered to be an input buffered NIC by associating the buffers on the output of each NIC with the inputs of the next NIC.

The NIC provides the basic cut-through switching function from router inputs to outputs. It examines the header flit of an incoming packet to determine which output link the packet should be forwarded to. The NIC also does proper arbitration if there are competing requests for an output link. The arbitration policy could be round-robin or priority-based or both. If a requested output link is not available, then the requesting flit is blocked and stored in the corresponding input buffer. It is assumed that our mesh NIC can connect all inputs to outputs in a single network clock cycle. Once a switch connection between an input and output link is established, it is broken only after the last flit of a packet has been transferred. We consider

	Blocking		Non-blocking	
	Direct Networks	Hierarchical-ring Networks	Direct Networks	Hierarchical-ring Networks
Wormhole	√	√		
VCT	√			√
Cell	√	√		√

Table 3.1: Switching and flow-control techniques used in evaluating hierarchical-ring and direct multiprocessor networks.

flit-sized and (single or multiple) cache line sized buffers. Under the assumption of constant pin constraints, a 128-bit wide channel for rings with one input and output connection per ring NIC translates into a 32-bit wide channel for meshes with four input and four output connections per mesh NIC.

3.2 Switching and Flow-control Techniques

In this section, we briefly mention the different switching and flow-control techniques used in our performance evaluation. We consider three different switching techniques: wormhole (WH), virtual cut-through (VCT), and cell switching. The switching schemes are examined for both blocking and non-blocking networks (with some exceptions). Wormhole (both single-flit and buffered) switching is a natural candidate for switching in blocking networks, while cell switching is a natural candidate in non-blocking networks. Virtual cut-through switching in blocking networks results in a similar performance to buffered wormhole switching and therefore is used only in non-blocking networks. In the non-blocking variant of virtual cut-through switching a node drops the packet whenever a node cannot buffer it in its entirety. The dropped packet is recovered through negative acknowledgments and through time-outs.

Table 3.1 presents the switching and flow-control combinations considered in this dissertation. We do not consider non-blocking flow-control in direct networks since our preliminary simulations showed that they result in poor performance.[4] For direct networks, we consider both wormhole and buffered wormhole switching; virtual cut-through switching results in similar performance as buffered wormhole switching and is therefore not further considered for direct networks. For hierarchical-ring networks, we consider cell switching under both blocking and non-blocking flow-control, virtual cut-through under non-blocking flow-control, and buffered wormhole switching under blocking flow-control.

[4]This is mainly due to the overhead involved in dropping and recovering packets, since the packets are much longer (number of flits) in direct networks compared to their hierarchical-ring counterparts.

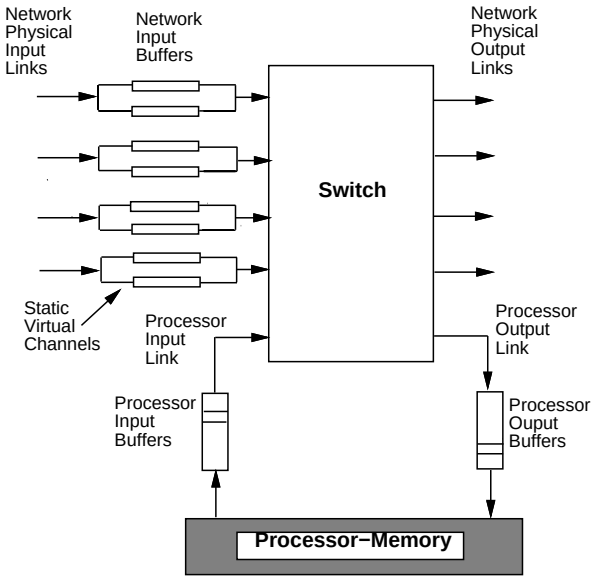

Figure 3.6: A network interface controller for a 2-dimensional mesh or a torus with two static virtual channels per physical link.

Virtual channels were introduced in Chapter 2 as a means for providing deadlock free routing. We use two virtual channels in wormhole switched hierarchical ring, bidirectional ring and 2-dimensional tori networks, where deadlock might otherwise occur. We will also use them at times to improve system throughput in 2-dimensional mesh networks (Chapters 4 and 7). Figure 3.6 presents a block diagram for a mesh NIC with two virtual channels sharing the bandwidth of each physical link. Each virtual channel has its own input buffer. The virtual channels in this case are referred to as *static*, since their number per physical link remains constant. In Chapter 7, we introduce dynamic virtual channels, where the number of virtual channels per physical channel can vary, in the context of priority networks.

3.3 Simulator

The simulator we use reflects the behavior of a system at the register-transfer level on a cycle-by-cycle basis. It was implemented using the *smpl* simulation library [59]. For the version of our simulator that is driven by synthetic workload models, the batch means method of output analysis was used, with the first batch discarded to account for initialization bias. In the batch means method, a single long run is divided into sub-runs called *batches*. A separate sample mean is computed for each batch. These batch means are then used to compute the grand mean and confidence interval. The batch termination criterium was that each processor had to complete at least some minimum number of requests as opposed to using a total number of requests completed over the entire system as this could substantially underestimate the mean

response times since requests with long response times are underrepresented.

A hierarchical-ring base simulator was validated against measurements taken from the Hector prototype, a hierarchical-ring architecture [93]. The Hector prototype and the base simulator use cell switching and non-blocking flow-control. The base simulator was then extended to model other switching techniques, such as wormhole and virtual cut-through and flow-control techniques such as blocking and hybrid flow-control. For meshes, tori, and bidirectional rings, the processor and memory modules are essentially the same as in the hierarchical-ring simulator with new NIC modules added that incorporate appropriate switching, routing and flow-control techniques.

A *program-driven* simulator is used to run real applications on hierarchical-ring and 2-dimensional mesh and torus connected multiprocessor systems. The program-driven simulator controls the scheduling of processes so that the interleaving of memory references is the same as it would be on the simulated machine. Our program-driven simulator is partitioned into two main parts: a memory reference generator (the front end) and a target system simulator (the back end), as illustrated in Figure 3.7. The reference generator models the execution of an application program on some number of processors. When an application generates a memory reference, the front end sends an event to the back end. The back end typically models the system interconnect and the memory hierarchy. When the back end completes the operations for an event, it signals the front end to continue with process execution. A simulation library manages and schedules events and processes.

We use MINT (a MIPS interpreter) as our front-end that encompasses the memory reference generator and simulation library components [90]. The input to MINT is a statically-linked Irix executable file compiled for the MIPS R3000 processor. Our back-end target system simulator is the same as we use for the synthetic workload. Hence, the difference between the two simulators is that we use the MINT event scheduler in the former and the *smpl* event scheduler in the latter. However, we use *smpl* routines in both simulations to collect statistics. Using the same back-end allows us to directly compare the results obtained from both simulations.

For the synthetic workloads, our main performance measures are transaction latency and system throughput, whereas for the program-driven simulations they are transaction latency and execution time of parallel applications. These and other performance measures are defined as follows:

Transaction latency is the elapsed time between when a request is first issued and the time the corresponding response is received.[5] It is measured in processor clock cycles and includes

[5]For writes the response packet returns upon queuing of the request at the target memory, so it is possible for the target memory to be still processing a write request after the transaction completes in the above sense.

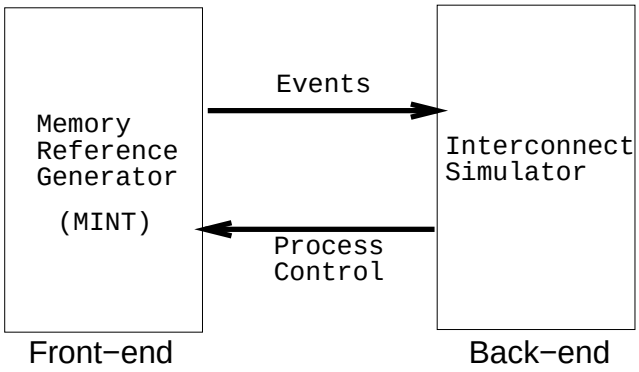

Figure 3.7: A program-driven simulator contains two major components: a memory reference generator and a target system (interconnect) simulator.

any time-outs and retransmissions that might occur in the non-blocking networks.

System throughput is defined as average number of requests issued in the entire system per processor cycle.

Average NIC and IRI delays are the average time a packet waits in NIC and IRI buffers, respectively, and are measured in processor cycles.

Maximum achievable throughput is the system throughput just before network saturation. At network saturation, a small increase in request rate produces a large increase in transaction latency accompanied by no increase in throughput.

Link utilization measures the fraction of time a link is utilized for data transfer. It gives a measure of network contention. Ideally, we would like link utilization to be low.

3.4 System and Workload Parameters

A shared-memory multiprocessor interconnection network can be characterized in part by the following parameters:

1. system size (number of processors),
2. the relative processor, memory and network cycle times,
3. the maximum number of transactions a processor may have outstanding at a time, and
4. the topology.

Allowing only one outstanding transaction per processor results in a low processor efficiency. A number of techniques such as relaxed memory consistency models, prefetching, non-blocking reads, and multiple hardware contexts have been proposed to minimize the time a processor

is blocked waiting for some transaction to complete and thus improve processor efficiency [36]. We model this effect by allowing up to four outstanding transactions per processor. Thus, a processor does not block until the number of outstanding transactions has exceeded the maximum.

The topology of hierarchical-ring networks is specified by the branching factor at each level of the hierarchy, starting at the local ring up to the global ring. A topology specified as $8 \times 4 \times 2$ refers to a three-level hierarchy with 8 nodes per local ring, 4 level-1 rings per level-2 ring, and 2 level-2 rings connected to the global ring. To specify the topology of 2-dimensional direct networks, we use the $n \times n$ notation where n is the number of nodes in a single dimension, assuming square meshes or tori.

The main parameters in our synthetic workload model include the request rate, which is the mean time between cache misses[6] given a non-blocked processor, the probability that the cache miss is a read, and a measure of communication locality. We subject the network to a wide range of request rates from 0.0001 to 0.1. This corresponds to a range from 1 cache miss per 10000 cycles to 1 miss in 10 cycles. We will show later in this chapter that this range includes most request rates exhibited by real programs. Given a cache miss, we assume the probability of it being a read is 0.7, which is consistent with observed behavior [33]. For cache line transfers, we assume the availability of page-mode DRAM, where the first word of a cache line is provided by the memory after, say, 10 processor cycles, and successive words are provided in consecutive processor cycles [15].

For our synthetic workload model, two main memory transactions, namely the *read* and *write* transaction and four types of packets, namely, read request, read response, write request and write response are simulated. In the case of a read transaction, the request packet contains the target memory address and the response packet contains the requested cache line data. In the case of a write transaction, the request packet contains the cache line data to be written and the response packet contains an acknowledgment. For writes, the response packet is sent back to the requesting station as soon as the write is queued at the target memory, so the latency of the actual memory operation is hidden.[7]

Our synthetic workload model does not take coherence traffic explicitly into account. Cache coherence traffic could be included within a low-level workload model such as ours by providing a translation from a high-level workload model to a low-level workload model. Holliday and Stumm [42] did a preliminary study of the effect of such a translation for the case of software

[6] The mean time between cache misses follows a negative exponential distribution.

[7] To isolate the issues related to network performance, we assume that the target node always accepts all requests to it without generating negative acknowledgments.

cache coherence using the approach developed by Adve et.al. [1]. The resulting ranges for the low-level workload parameters were consistent with the ranges we consider.[8]

Communication locality can greatly affect system throughput in shared-memory multiprocessor networks. We use *clusters of locality* to model locality in our synthetic workloads [42]. This communication model logically organizes all processors into clusters and assigns a probability for each cluster being the target of a transaction. Hence, two parameters, where each is a vector, specify a specific locality model. The vector $S = (S_1, S_2, \cdots, S_n)$ specifies the size of each cluster. Here, $S_1, S_2, \cdots, S_n$ represent the size (number of processors) of clusters $0, 1, \cdots, n$, respectively. The vector $P = (P_1, P_2, \cdots, P_n)$ specifies the probability of each cluster being the target of a transaction. Given that the target memory is in a particular cluster, the probability of a processor module within that cluster is uniformly distributed. This definition of clusters is independent of the topology of the network. For a hierarchy of rings, the clusters are defined in terms of the absolute difference (modulo the size of the system) between the two processor numbers, when numbered them left to right when processors are viewed as the leaves of a tree defined by the ring hierarchy. For 2-dimensional mesh or torus connected networks, clusters are defined in terms of the link distance between processors. The communication model described is similar to ones described in other studies of direct networks [2, 3].

In our studies, we used two specific workloads derived from this model:

1. Workload T_{loc}, represented as $S = (1, 4, n-5)$ and $P = (0.5, 0.8, 1.0)$, models 3 clusters where the first cluster is the source processor module itself, the second cluster contains the source processor module's four closest neighbors, and the third cluster contains all other processor modules. Cluster 1 has probability 0.5 of being the target, cluster 2 has probability 0.8 of being the target, given that the target is not in cluster 1, and cluster 3 has probability 1.0 of containing the target, given that the target is not in cluster 1 or cluster 2. This workload models high communication locality where there is a probability of $0.5 + (1 - 0.5) * 0.8 = 0.9$ that the target memory lies within the first two clusters.

2. Workload $T_{uniform}$, represented as $S = (n)$ and $P = (1.0)$ with n being the total number of processors, models a single cluster that has a probability 1.0 of containing the target memory. This workload models poor communication locality where there is an equal probability of a processor accessing any other processor's memory (including its own memory). We include $T_{uniform}$ in particular because many other studies have used this workload [2, 14, 19, 27, 34, 39, 53, 85], allowing easier comparisons.

[8] It is possible to include traffic due to write-backs and invalidations within the read and write parameters.

Parameter	*Value*	*Description*
n	$16, 64$	Number of processors
b	1	Number of memory banks
$n_{L1} \times n_{L2} \times \cdots \times n_{Lh}$	$8 \times 2, 8 \times 4 \times 2$	Hierarchical-ring Topology
	$4 \times 4, 8 \times 8$	2D Mesh/Torus Topology
$NXMY$	$N2M10$	Ratio of network and memory cycles to processor cycle
T	4	Maximum number of outstanding transactions
λ	0.0001-0.4	Request rate
R	0.7	Probability that a cache miss is a read
$S = (S_1, S_2, \cdots, S_m)$	$(N), (1, 4, n-5)$	Cluster size
$P = (P_1, P_2, \cdots, P_m)$	$(1), (0.5, 0.8, 1.0)$	Cluster probabilities

Table 3.2: System and synthetic workload parameters and their range of values used in our simulations.

The system and workload parameters used in our study are summarized in Table 3.2. We consider system sizes of 16 and 64 processors to represent small and medium-scale multiprocessor networks, respectively. We define the *cycle ratio* as the relative speed of the processor, network, and memory [42]. It is specified as $NXMY$ which means that each network cycle is X times as slow as a processor cycle and the memory requires Y processor cycles to service one memory request. We define network cycle time as the time required for a packet to move from the input of one node to the input of the next node. Such a transfer need not occur in a single network cycle. Our assumption that the network cycle time is a factor of two slower than the processor cycle time is justified from the fact that for a 5ns processor cycle time (200 MHz), our ring cycle time of 10ns is close to that used in SCI performance studies [82].[9]

All simulation results have confidence interval half-widths of 1% or less at a 95% confidence level, except near saturation where the confidence interval half-width may increase to a few percent.

3.5 Program-driven Simulation

For our program-driven simulation, we simulate a cache coherent shared address space multiprocessor with physically distributed memory and one processor per node. Every processor has a two-level cache that is kept coherent using a directory-based protocol [88]. Table 3.3 presents some important system parameters used in our program-driven simulations.

[9]SCI specifies a ring cycle time of 2ns [41] with 4 ring cycles required to transfer a packet from the input of one node to the input of the neighboring node.

Description	*Value*
Ratio of network to processor cycles	2
Memory read/write time	10 Proc cycles
Memory Tag time	6 Proc cycles
Cache	2-level, Direct-mapped
L1/L2 data-cache size	32 KB/1 MB
L1/L2 cache line size	32 Bytes

Table 3.3: System parameters used in program-driven simulations.

We use application programs from the SPLASH-2 suite [94], which consists of a mixture of complete applications and computational kernels. It has applications and kernels drawn from a variety of disciplines that include scientific, engineering and graphics computing. We briefly discuss here those applications and kernels we use in our simulations. Detailed descriptions of all programs are available in [86, 94] and the characteristics of their memory access behavior can be found on the SPLASH web page [87]. In each case, the input data sets we use are those specified for the programs in the SPLASH-2 suite and the data are distributed among the processing nodes according to the SPLASH-2 guideline.

FFT: The FFT kernel is a complex 1-D version of Bailey's six step FFT algorithm. The data set consists of n complex data points to be transformed and another n data points referred to as the *roots of unity*. Both sets of data are organized as $\sqrt{n} \times \sqrt{n}$ matrices and partitioned so that every processor is assigned a contiguous set of rows. Communication occurs in three matrix transpose steps, which require all-to-all interprocessor communication. Every processor transposes a contiguous sub-matrix of $\frac{\sqrt{n}}{p} \times \frac{\sqrt{n}}{p}$ from every other processor, where p is the number of processors, and transposes one sub-matrix locally. The FFT implementation is optimized to minimize the interprocessor communication.

LU: The LU kernel factors a dense matrix into the product of a lower and upper triangular matrix. The dense $n \times n$ matrix is partitioned into an $N \times N$ array of $B \times B$ blocks, where $n = NB$, to allow the exploitation of temporal locality on sub-matrix elements. The block ownership is assigned using a 2D scatter decomposition and the blocks are allocated locally to processors that own them. The blocks are updated only by the processors that own them. Elements within a block are allocated contiguously to improve spatial locality benefits. Block size B is critical to performance and a smaller size (B=8 or 16) is chosen to strike a good balance between cache miss rate and load balance.

Radix: The integer radix sort kernel sorts a set of k-bit integer keys by examining r-bits in each

iteration. The r-bit field is called a digit. The algorithm is iterative and performs one iteration for each radix r digit of the keys. The keys are stored in a global array of integers. The sorted keys are stored in another global array. Both arrays are partitioned across the processors. In each iteration, a processor passes over its assigned keys and generates a local histogram. The local histograms are then accumulated into a global histogram, thereby generating all-to-all communication. Finally, each processor uses the global histogram to permute its keys into a new array for the next iteration. The predominant pattern of communication in the radix sort is bursty.

Ocean: The Ocean application studies large-scale ocean movements based on eddy and boundary currents. The computation is performed in a number of time steps. At each time step several independent calculations are made over a number of grids. It is optimized to minimize the communication-to-computation ratio by partitioning the grids into square-like sub-grids. The main data structure is a number of 2-dimensional arrays which represent different grids. The partitioning of computation over all processors is done by decomposing the data domain. Each processor performs computation over its assigned sub-domain which consists of a few grid points. The exchange of data takes place only for the boundary elements with a near-neighbor communication pattern. The data written by local processors in one step are read by remote processors in the subsequent step allowing for optimization by the cache coherence protocol and prefetching.

Raytrace: Raytrace renders a three-dimensional scene using ray tracing. A hierarchical uniform grid is used to represent the scene. A ray is traced through each pixel in the image plane, and reflects, in unpredictable ways, off the objects it strikes. Each contact generates multiple rays, and the recursion results in a ray tree per pixel. The image tree is partitioned among processors in contiguous blocks of pixel groups, and distributed task groups are used with task stealing. Major data structures represent rays, ray trees, the hierarchical uniform grid, task queues, and the primitives that describe the scene. The data access patterns are highly unpredictable in this application.

Tables 3.4 and 3.5 summarize the characteristics of the applications for 16 and 64 processor systems, respectively, corresponding to the two system sizes we used for running real applications.[10] The tables present for each application its input data set, average number of memory requests per processor (L2 cache misses), the average memory request rate, the average interval between memory requests in processor cycles, and the fraction of such memory requests that are reads. The numbers are shown for 32-byte and 64-byte cache line sizes. It

[10] These characteristics were obtained running the applications on a hierarchical ring connected multiprocessor, with a two-level hierarchy for 16 processors (8×2) and a three-level hierarchy for 64 processors ($8 \times 4 \times 2$).

Application	Problem Size	Requests per Proc		Proc Request Rate		Request Interval		Fraction Reads
		32B CL	64B CL	32B CL	64B CL	32B CL	64B CL	
FFT	64K points	22741	11433	0.0166	0.0083	61	121	0.65
Radix	1M integer	63152	36447	0.00834	0.00481	120	208	0.63
LU	512×512 16×16 blocks	90973	53472	0.0048	0.00284	209	353	0.67
Ocean	258×258	153303	106602	0.0079	0.0059	127	170	0.82
Raytrace	Teapot Geometry	44134	29935	0.0035	0.00235	286	426	0.73

Table 3.4: Characteristics of some real applications from SPLASH-2 suite. The network simulated is a 16 processor 2-level 8×2 hierarchical ring with 32-byte cache lines.

Application	Problem Size	Requests per Proc		Proc Request Rate		Request Interval		Fraction Reads
		32B CL	64B CL	32B CL	64B CL	32B CL	64B CL	
FFT	64K points	6015	3029	0.0174	0.00874	58	115	0.65
Radix	1M integer	19836	14329	0.00999	0.00722	100	139	0.63
LU	512×512 16×16 blocks	27046	13899	0.00571	0.00293	176	342	0.67
Ocean	258×258	55132	44041	0.01087	0.00868	92	116	0.82
Raytrace	Teapot Geometry	11926	9002	0.00373	0.00282	268	355	0.73

Table 3.5: Characteristics of some real applications from SPLASH-2 suite. The network simulated is a 64 processor 3-level $8 \times 4 \times 2$ hierarchical ring with 32-byte cache lines.

should be noted that the request rates shown here are on the low side because they do not include the requests generated to maintain coherent caches. Nevertheless, the communication frequency of applications varies widely as seen from the average memory request rate. This suggests that its quite difficult to predict the typical communication behavior. However, none of the applications are embarrassingly parallel and most of them demonstrate regular communication patterns. By taking into account a range of request rates (from very low to very high values), our synthetic workload model contains communication patterns of most present and future parallel applications.

Chapter Summary

This chapter presented system descriptions of 2-dimensional mesh and hierarchical-ring connected multiprocessor networks. We presented our parametric and program-driven simulation methodology with the description of synthetic workload models and real applications. The system and workload parameters we use in this dissertation were summarized.

Chapter 4
Mesh, Torus, and Ring Networks: Comparative Performance

The most popular direct multiprocessor network topology in use today is the 2-dimensional mesh or torus. This is partly because routers are available off the shelf and partly because the bisection bandwidth of such networks grows with system size, thereby allowing them to scale to a larger number of nodes. Examples of mesh-connected shared memory multiprocessors include Stanford University's FLASH [55], MIT's Alewife [4], and SGI's Origin [16]. However, when compared to ring networks, a 2-dimensional mesh or torus network has smaller link width under identical pin constraints due to twice the number of directly connected neighbors. This also increases their switching and routing complexity. Moreover, their network diameter increases more rapidly with the number of nodes than when compared to hierarchical rings.

In this chapter we will show that, from a performance point of view, hierarchical-ring networks are interesting alternatives to low-dimensional direct networks for shared-memory multiprocessors. We compare the performance of hierarchical-ring and direct shared-memory multiprocessor networks and show that hierarchical-ring networks generally perform better than 2-dimensional meshes and tori when connecting a small number of nodes ($\leq$ 16), but that meshes and tori perform better in systems with a large number of nodes. This is primarily because of the constant bisection bandwidth constraints of hierarchical-ring networks [71].

For direct networks, we consider 2-dimensional meshes and tori. Recently, there has been some interest in single bidirectional ring networks for shared-memory multiprocessing [47, 66], and we therefore include them in our study as a special case of tori networks of one dimension. Since wormhole switching is the predominant switching technique in direct networks, we will use them in both our direct and hierarchical-ring networks, even though we will show in Chapter 6 that cell switching performs somewhat better in hierarchical-ring networks. For both 2-dimensional meshes and tori networks, we use *dimension ordered* routing which is minimal and deterministic. For wormhole switched hierarchical-ring networks, we extend Dally's [18] deterministic deadlock-free routing in a single ring (k-ary 1-cube) to a hierarchy of rings. The routing technique is explained in Chapter 7.

	Channel width	Input/ Output Channels	VCs per Physical Channel	Cache line size	NIC memory requirements		
					1/4 CL	1/2 CL	1 CL
Hierarchical Rings	128b	1/1	2	32B	-	-	96B
				128B	-	-	288B
Bidirectional Rings	64b	2/2	2	32B	-	96B	-
				128B	-	288B	-
Meshes & Tori	32b	4/4	2	32B	96B	-	-
				128B	288B	-	-
Meshes	32b	4/4	1	32B	48B	-	-
				128B	144B	-	-

Table 4.1: A comparison of memory requirements for ring and mesh NIC buffers of different sizes.

4.1 Comparative Performance Evaluation

We evaluate the systems using $T_{uniform}$ and T_{loc} as our synthetic workloads and five real applications from the SPLASH-2 suite for our program-driven simulations. Since hierarchical-ring networks are highly configurable, we use topologies that exhibit low latency and high throughput for most memory access patterns. These topologies are derived in Chapter 5. Also, we assume optimal buffer sizes at NICs and IRIs, as derived in Chapter 6.

In our comparison, we generally try to be fair, although we slightly favor direct networks. For example, our T_{loc} workload model favors meshes, as it minimizes the number of hops in meshes, but not so for ring-based systems. We assume the hierarchical rings are wormhole switched, even though cell switched rings tend to perform somewhat better (see Chapter 6). Also, we assume the same routing time for rings, meshes, and tori, although routing in rings is simpler and hence faster. Since we assume two virtual channels per physical channel for both hierarchical-ring and tori networks (for deadlock free routing), we also consider the case of two virtual channels per physical channel for meshes. However, in meshes, we use virtual channels purely for improving network throughput by allocating these channels to packets so as to avoid head-of-line blocking, while using dimension ordered routing to prevent deadlock.

Under the assumption of constant pin constraints, a 128-bit wide channel for rings with one input and one output connection per ring NIC translates into a 32-bit wide channel for meshes and tori with four input and four output connections per NIC, and a 64-bit wide channel for bidirectional rings. When considering the on-chip memory requirement for buffers, assuming cache line sized buffers for meshes favor meshes and tori, while assuming single-flit buffers for meshes favors hierarchical rings. For a fair comparison we assume that the memory requirement in a router is the same for both direct and hierarchical-ring networks. For a hierarchical ring,

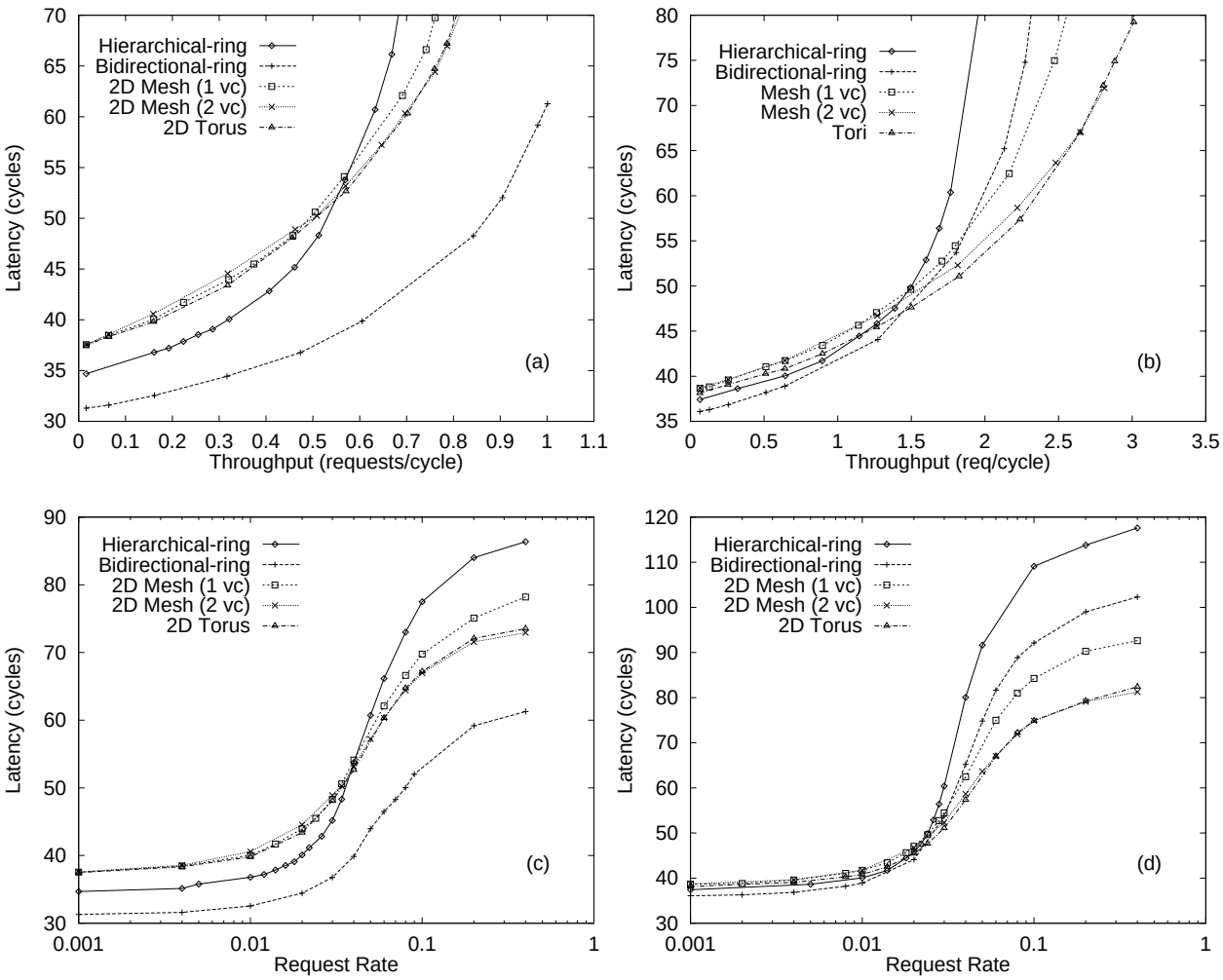

Figure 4.1: Performance of systems with 32-byte cache lines under the T_{loc} workload: a) throughput-latency curves for 16 processor systems, b) throughput-latency curves for 64 processor systems, c) latency as a function of request rate for 16 processor systems, and d) latency as a function of request rate for 64 processor systems.

we assume a cache-line (CL) sized NIC buffer, which translates into a NIC buffer size that can hold one-fourth of a cache line in a mesh and torus and one-half of a cache line in a bidirectional ring. Table 4.1 summarizes the memory requirement in routers of direct and hybrid networks.

4.1.1 Access Patterns with Memory Locality

We consider in this section the T_{loc} workload that exhibits high locality in the memory access pattern. Figure 4.1 presents latency, both as a function of throughput and the request rate for four different topologies for 16 (the first column of graphs) and 64 (the second column of graphs) processor systems with 32-byte cache lines. Each throughput-latency curve is obtained

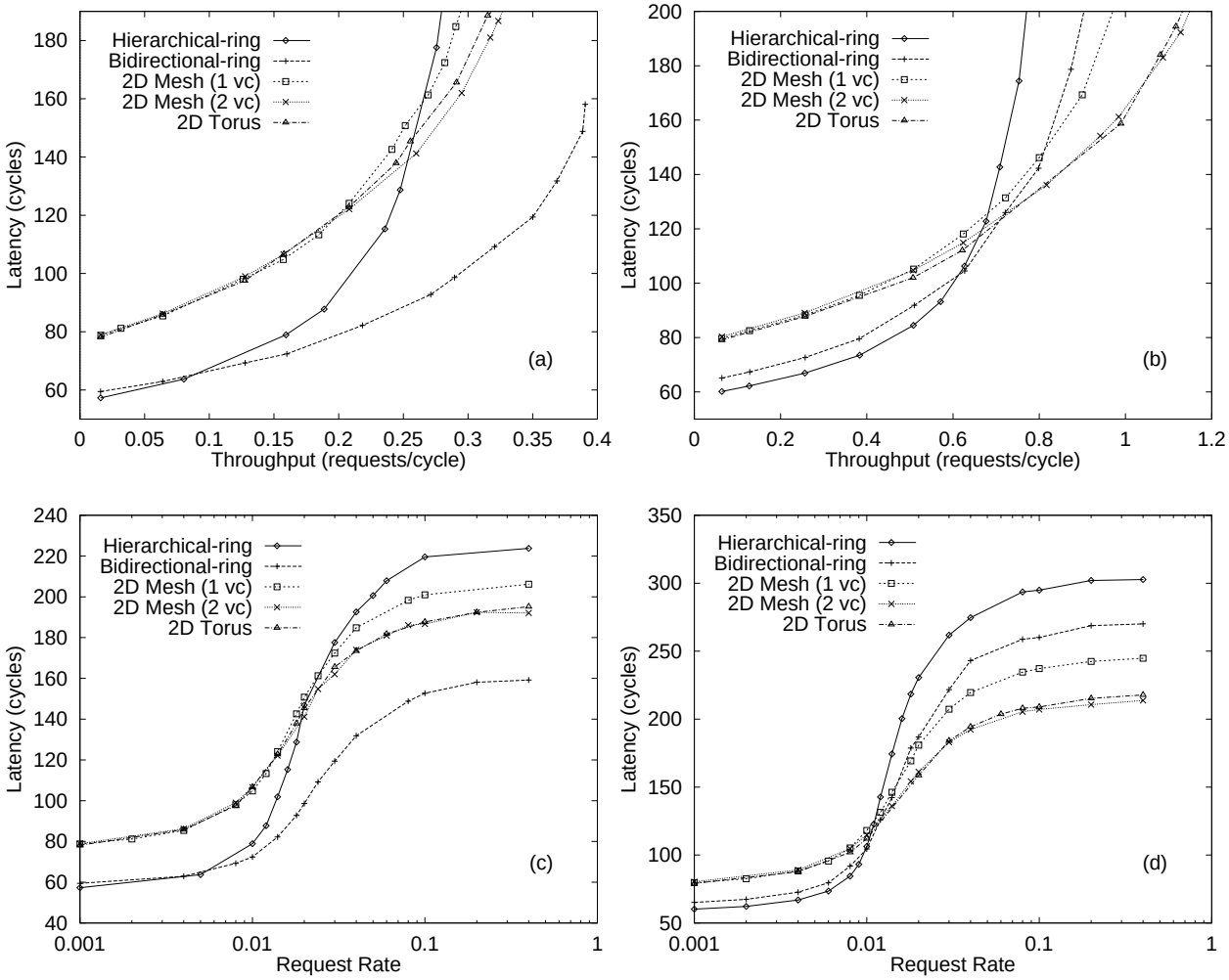

Figure 4.2: Performance of systems with 128-byte cache lines under the T_{loc} workload: a) throughput-latency curves for 16 processor systems, b) throughput-latency curves for 64 processor systems, c) latency as a function of request rate for 16 processor systems, and d) latency as a function of request rate for 64 processor systems.

from two curves that plots throughput and latency independently against request rate [9].[1] The points at the lower-end of the throughput scale in a throughput-latency curve correspond to low request rates, while those at the higher-end of the throughput scale correspond to high request rates. Initially, when we increase the request rate the throughput scales with little or no increase in latency and reaches a point after which any increase in request rate results in little increase in throughput accompanied by a large increase in latency. We refer the throughput at this point as the maximum achievable throughput for a given network topology.

In Figure 4.1, we consider meshes with both one and two virtual channels. There are two important observations from these graphs. For a smaller system size of 16 processors, the

[1] We will mainly use the throughput-latency representation in this dissertation.

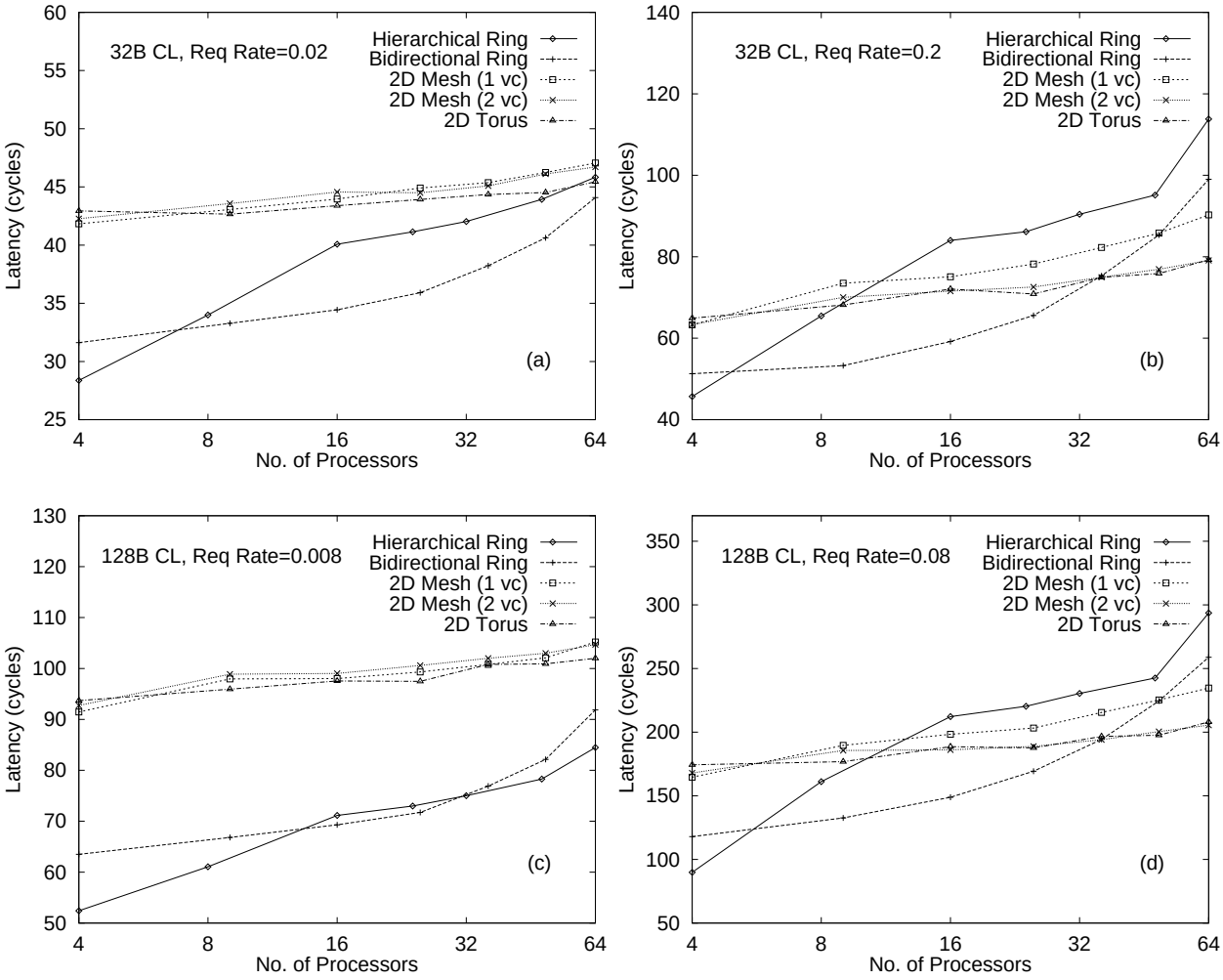

Figure 4.3: Average latency of rings, meshes, and tori networks when scaled under T_{loc} workload for a) 32-byte cache line and low request rate, b) 32-byte cache line and high request rate, c) 128-byte cache line and low request rate, and d) 128-byte cache line and high request rate.

bidirectional ring exploits locality much better than a hierarchical ring and a 2-dimensional mesh or torus, resulting in a low latency and high throughput curve. This is mainly because of the dynamic clustering effect of the bidirectional ring for memory access patterns that exhibit high locality [47]. Among hierarchical-ring and 2-dimensional direct networks, the hierarchical-ring network exhibits lower latency at low request rates, but suffers from early saturation at high request rates. However, for a large system size of 64 processors, both hierarchical and bidirectional rings suffer from early saturation at high request rates (because of the bisection bandwidth constraints) even though they exhibit latency values that are marginally lower than in 2-dimensional direct networks at low request rates.

The trend is similar for a large cache line size of 128 bytes except that both hierarchical and bidirectional rings exhibit latency values that are significantly lower (about 25%) than in

2-dimensional meshes and tori at low request rates. This is shown in Figure 4.2.

Figure 4.3 compares the latency of rings, meshes and tori as we scale the system size from 4 to 64 processors under the T_{loc} workload. The graphs in the top row of this figure are for systems with 32-byte cache lines, while the bottom two are for systems with 128-byte cache lines. We present two graphs for each cache line size; one at a low request rate and the other at a high request rate. Generally, rings (bidirectional and hierarchy) perform better than meshes and tori at low request rates, but meshes and tori scale better to a large number of nodes especially under high request rates. Also, hierarchical and bidirectional rings perform better than mesh and torus at larger cache line sizes. This is because with large cache line sizes, the relative length of a worm in a mesh or a torus network is higher than in a ring network, so the probability of blocking is higher in meshes and tori and because in the absence of contention, latency is dominated by the length of the worm (see Chapter 2).

We define the *cross-over* point as the number of nodes where the switch-over occurs. The results show that at low request rates, the cross-over point is sensitive to the cache line size, but is independent of the cache line size at high request rates. At low request rates, the cross-over points (for rings) are 64 processors or more. At higher request rates, the cross-over point for bidirectional ring is around 36 processors, while it is much lower (less than 16 processors) for hierarchical rings irrespective of the cache line size. This is because at high request rates, the constant bisection bandwidth of the ring limits its scalability, while meshes and tori scale well to large system sizes.

4.1.2 Access Patterns with No Memory Locality

In this section we consider the $T_{uniform}$ workload model that exhibits poor memory access locality. Figure 4.4 presents the throughput-latency curves for four different topologies for 16 and 64 processor systems with 32-byte cache lines (the first row of graphs) and 128-byte cache lines (the second row of graphs). There are two important observations from these graphs. For a smaller system size of 16 processors, both the hierarchical and bidirectional rings perform better than the 2-dimensional meshes and tori (with the difference in performance being higher at 128-byte cache line size than at 32-byte cache line size). Among ring networks, the bidirectional ring has a slightly superior throughput-latency curve than a unidirectional hierarchical ring. However, for a system size of 64 processors, ring networks perform worse than 2-dimensional networks. In this case, the 2-dimensional torus is clearly superior to all other networks under all request rates. Among rings, the hierarchical ring has a lower latency than the bidirectional ring under low request rates, although it suffers from earlier network saturation under high request rates. The trend is similar for a large cache line size of 128 bytes.

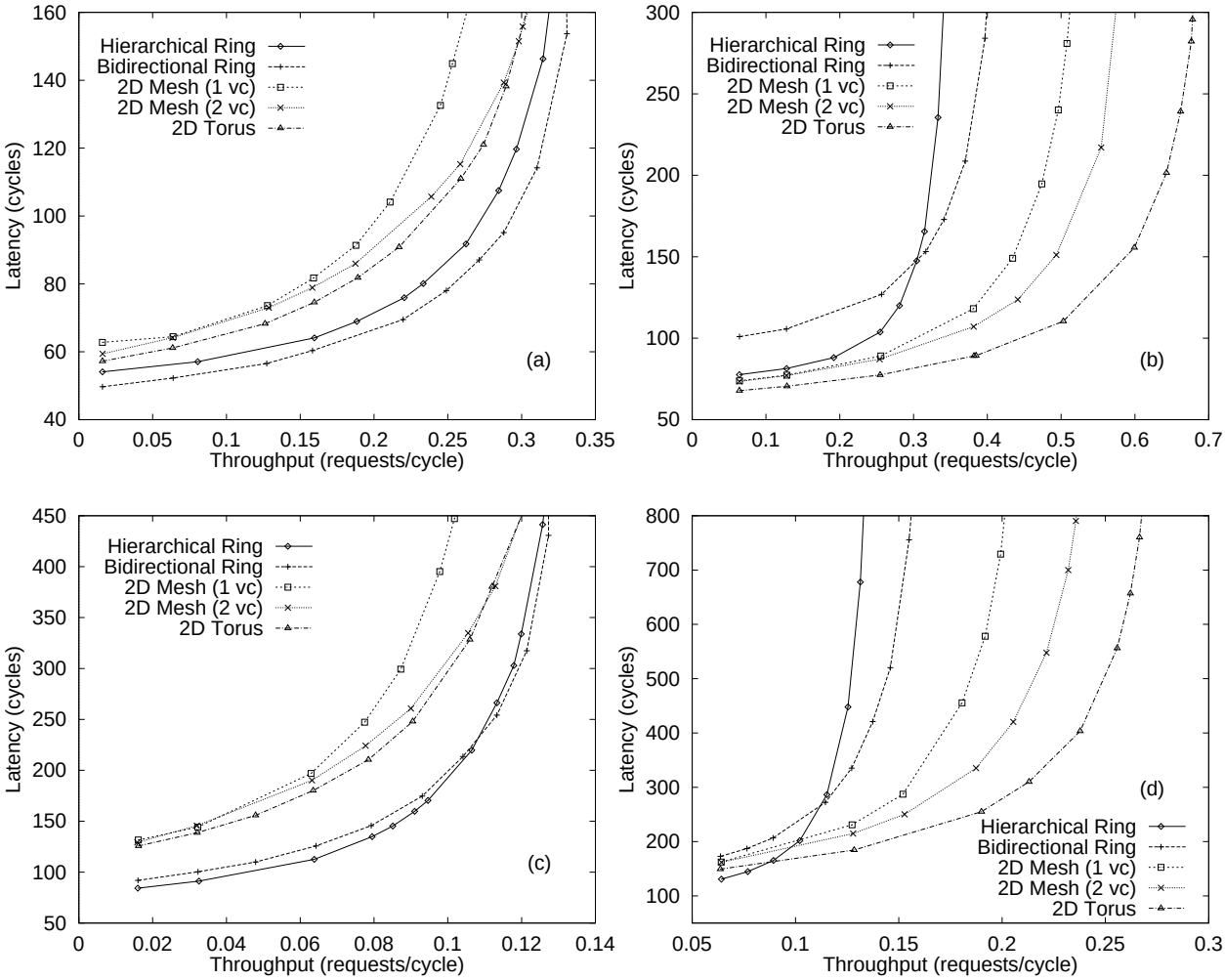

Figure 4.4: Throughput-latency curves of rings, meshes, and tori networks under $T_{uniform}$ workload for: a) 16 processor systems with 32-byte cache lines, b) 64 processor systems with 32-byte cache lines, c) 16 processor systems with 128-byte cache lines, and d) 64 processor systems with 128-byte cache lines.

Figure 4.5 compares the latency of rings, meshes and tori as we scale the system size from 4 to 64 processors for systems with 32-byte and 128-byte cache lines. We observe that rings (bidirectional and hierarchy) perform better than meshes and tori when connecting a small number of nodes, but meshes and tori perform better in systems with a large number of nodes.[2] Similar to the results for the T_{loc} workload, hierarchical rings perform better at lower request rates and at higher cache line sizes.

At low request rates, the cross-over points (for rings) are around 30 processors for 32-byte cache line systems, while the cross-over point for bidirectional rings is around 50 processors and is greater than 64 processors for hierarchical rings for 128-byte cache line systems. At higher

[2] An exception to this is the case of a system with a 128-byte cache line and low request rates, where hierarchy of rings perform better than meshes and tori even for the large system.

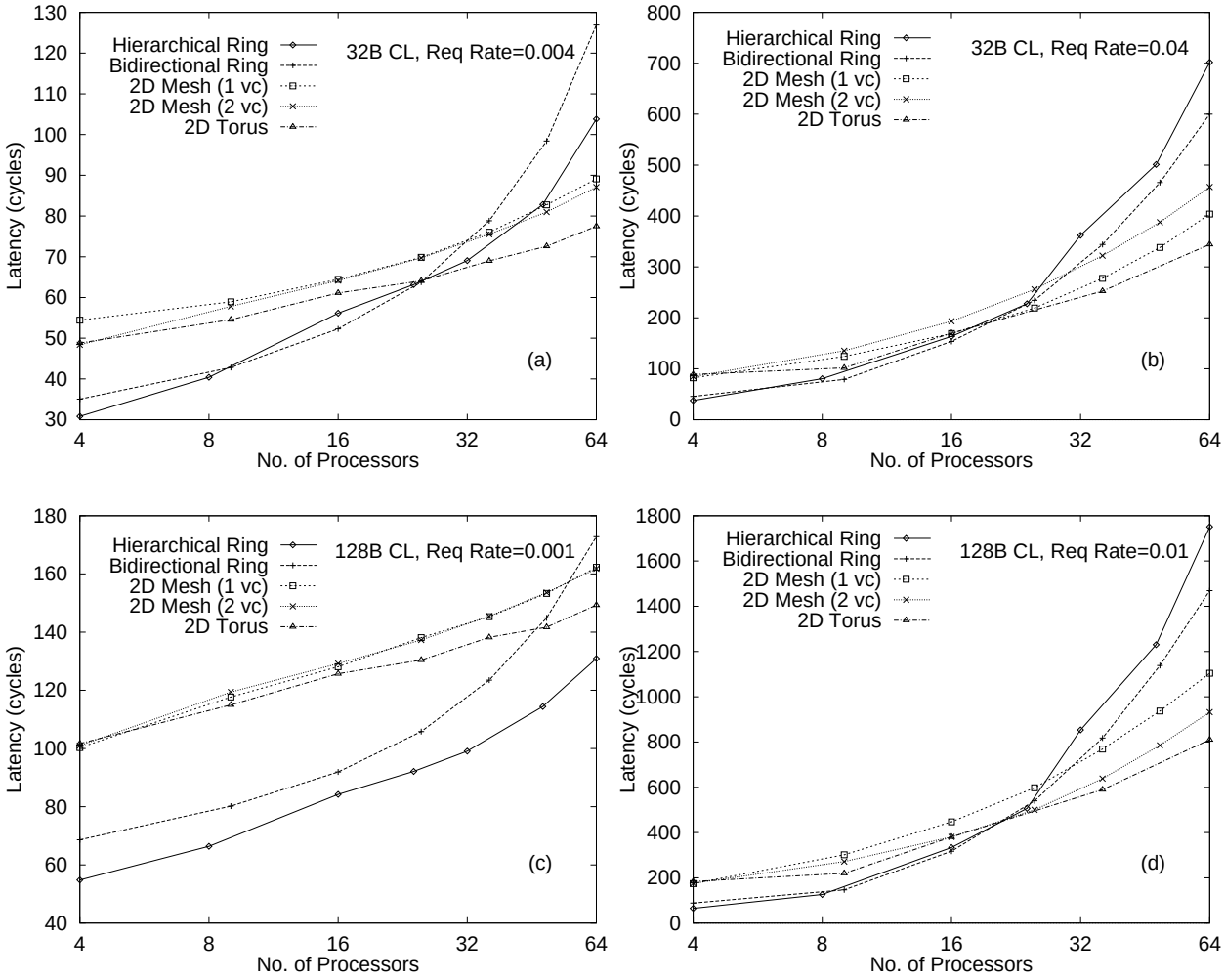

Figure 4.5: Average latency of rings, meshes, and tori networks when scaled under $T_{uniform}$ workload for a) 32-byte cache line and low request rate, b) 32-byte cache line and high request rate, c) 128-byte cache line and low request rate, and d) 128-byte cache line and high request rate.

request rates, the cross-over points (for rings) are around 25 processors irrespective of the cache line size. This is again because of the constant bisection bandwidth of the rings that limits their scalability, while meshes and tori scale well to large system sizes. To show this limitation we plot in Figure 4.6, the system throughput as a function of the number of processors (for high request rates). While the throughput of meshes and tori scale well with the system size, the ring throughput flattens out after about 24 processors. This observation makes us believe that ring-based networks are ill-suited for large system sizes, unless their bisection bandwidth is increased, at least for applications with little locality in their memory access patterns.

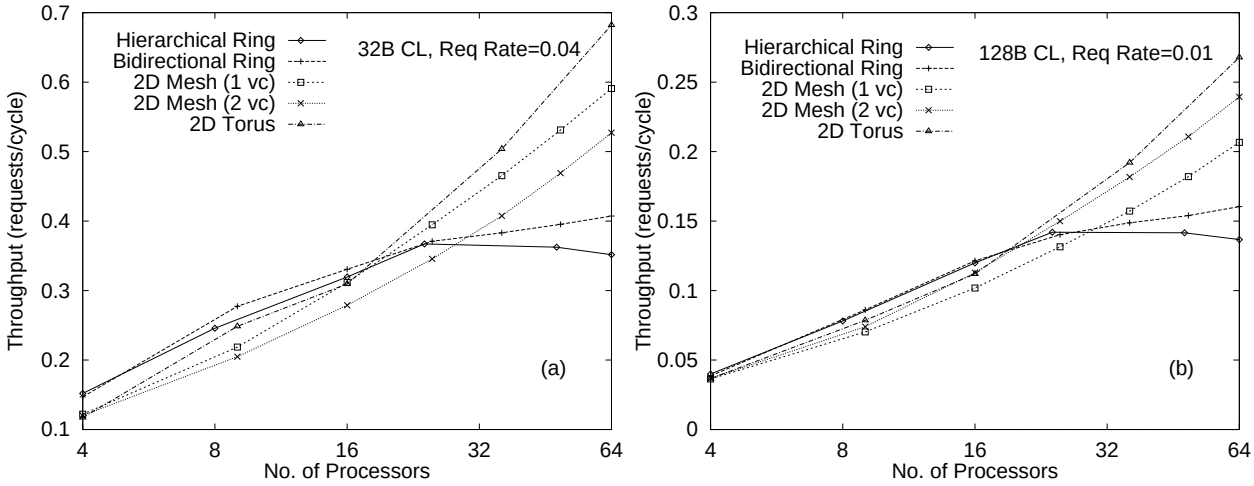

Figure 4.6: Throughput versus the number of processors for high request rates with cache line sizes of a) 32 bytes, and b) 128 bytes.

4.1.3 Program-driven Simulation

Here, we present the comparative performance of hierarchical ring and direct multiprocessor networks by running real applications using the program-driven simulator. The direct network topologies considered include 2-dimensional torus, 2-dimensional mesh, and bidirectional ring. The experimental set-up is the same as used for synthetic workload driven simulations. We present results for two system sizes, 16 and 64 processors with 32-byte cache lines. The performance measures used are normalized execution time and the average transaction latency (after L2 cache misses). Normalized execution time is computed as the ratio of the execution time of an application in a system with a particular network topology to that in a 2-dimensional mesh-connected system. As a result, the normalized execution time of a 2-dimensional mesh network will always be 1 for all applications. The parallel speedups[3] of these applications in the mesh network ranges from 12 for FFT to 14 for LU in 16 processor system and from 13 for Raytrace to 36 for FFT in 64 processor system. If the application running in a system with a particular network topology has a normalized execution time of less than 1, it means that the application runs faster than in a 2-dimensional mesh-connected system.

Figure 4.7 presents the normalized execution time of five different applications namely FFT, LU, Radix, Raytrace, and Ocean when run on 16 processor, 32-byte cache line systems connected by direct and hierarchical-ring networks. The input data sets for these applications have been summarized in Chapter 3. We can see that the execution times are all close, regardless of the

[3]To measure parallel speedup we consider only the *parallel section* of the code, and ignore the *sequential section*. In the SPLASH-2 suite, the parallel section is defined as the time from the creation of the master thread, until the master thread has successfully completed a wait() call for all of its children.

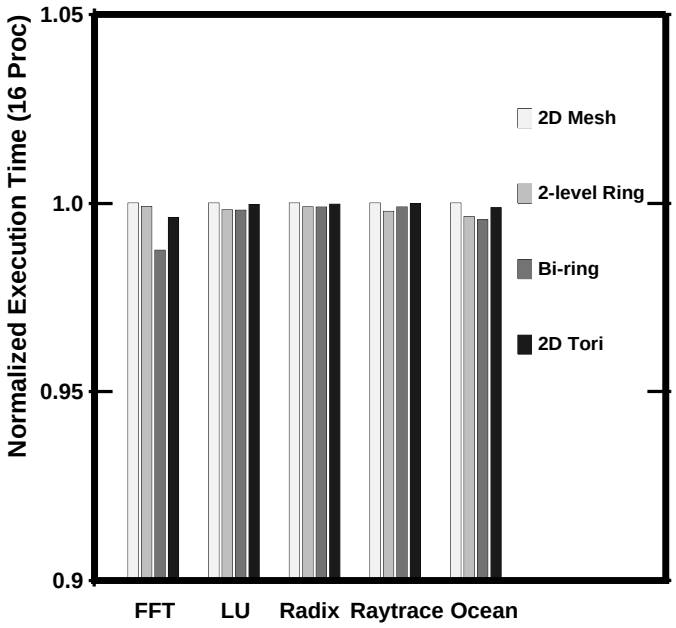

Figure 4.7: Execution times of SPLASH-2 applications normalized to the execution time of a 2-dimensional mesh-connected system. It is assumed that the applications run on 16 processor 32-byte cache line systems under program driven simulations. A 2-level 8×2 topology is used for the hierarchical-ring network and a 4×4 topology is used for the mesh and torus network.

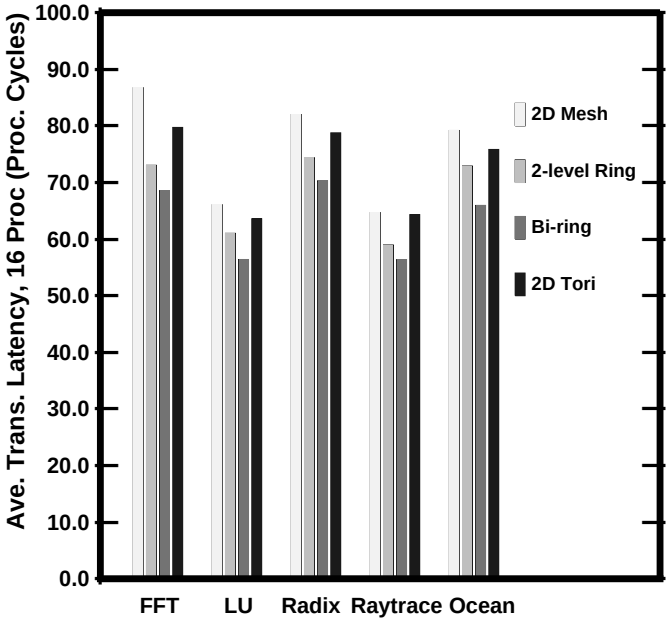

Figure 4.8: Average transaction latency (after L2 miss) for SPLASH-2 applications. It is assumed that the applications run on 16 processor 32-byte cache line systems under program driven simulations. A 2-level 8×2 topology is used for the hierarchical-ring network and a 4×4 topology is used for the mesh and torus network.

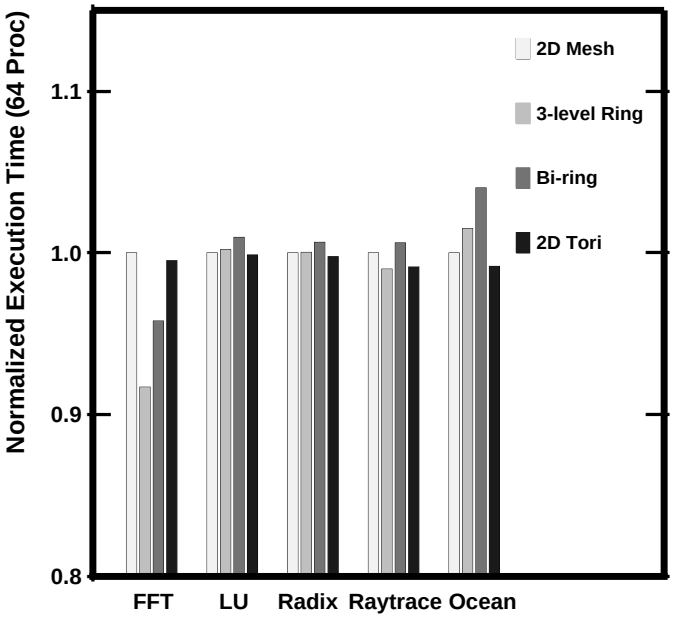

Figure 4.9: Execution times of SPLASH-2 applications normalized to the execution time of a 2-dimensional mesh-connected system. It is assumed that the applications run on 64 processor 32-byte cache line systems under program driven simulations. A 3-level $8 \times 4 \times 2$ topology is used for the hierarchical-ring network and an 8×8 topology is used for the mesh and torus network.

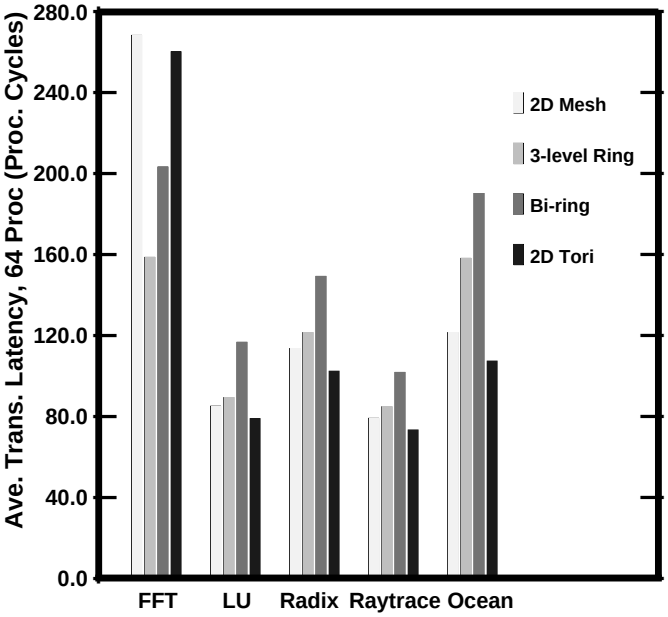

Figure 4.10: Average transaction latency (after L2 miss) for SPLASH-2 applications. It is assumed that the applications run on 64 processor 32-byte cache line systems under program driven simulations. A 3-level $8 \times 4 \times 2$ topology is used for the hierarchical-ring network and an 8×8 topology is used for the mesh and torus network.

system on which they run. The hierarchical and bidirectional ring-connected systems generally result in slightly lower execution times than the torus and mesh-connected systems. The average transaction latency (after L2 miss) in the above systems shown in Figure 4.8 explains the small difference in the execution time of applications, with bidirectional ring systems exhibiting the lowest latency, followed by the hierarchical-ring, torus, and mesh-connected systems, in that order. This confirms our earlier findings from simulating synthetic workloads: for small scale systems, rings exhibit lower average transaction latency than 2-dimensional direct networks.

For 64 processor, 32-byte cache line systems, we observe in Figure 4.9 that the meshes and tori systems perform somewhat better than the hierarchical and bidirectional ring systems, with the exception of FFT, where the hierarchical-ring system results in lower execution time. The average transaction latency for the applications is shown in Figure 4.10. With the exception of FFT, it is clear that 2-dimensional direct networks exhibit slightly lower latency than rings, and bidirectional rings exhibit the highest latencies.

From the execution times of applications on different networks for both 16 and 64 processor system sizes, we can conclude that though there are differences in transaction latencies the resulting execution times are so close that it effectively makes no difference which network type is used. This is because the applications generally have very high cache hit rates that significantly reduce network traffic. Nevertheless, we believe for applications that exhibit a lot of network activity (multimedia, transaction processing applications) the network type will have a significant performance impact.

Chapter Summary

This chapter presented a detailed comparative performance study of different low-dimensional direct and hierarchical-ring networks. It was shown that the hierarchical-ring network performs better than 2-dimensional direct networks at low (below saturation) request rates either when the workloads exhibit high locality in memory access pattern or for large cache line sizes. Hierarchical-ring networks performance is the same or slightly better than 2-dimensional direct networks for small system sizes of up to 25 processors (the crossover point) even when there is poor locality in the memory access patterns. However, 2-dimensional direct networks scale well for larger system sizes. This is because the bisection bandwidth of direct networks grows with system size allowing them to scale to a larger number of nodes than in the hierarchical-ring networks.

Chapter 5
Topology and Bisection Bandwidth

This chapter considers topology issues. For a given number of processors there are not many ways to configure a 2-dimensional direct network, so studying topology issues of such networks is relatively uninteresting. We assume square topologies for 2-dimensional direct networks since they are currently popular [4, 55, 57], although there have been a few studies on hexagonal [12] topologies. Given this fact, the rest of this chapter focuses mainly on topology issues of hierarchical-ring networks and the impact of topology on performance of such networks.

Multiprocessors based on a single ring are limited to a small number of processors because the diameter of the network grows with the number of processors, and because of the constant bisection bandwidth. A hierarchical-ring network can accommodate a larger number of processors by interconnecting multiple rings in a hierarchical fashion. A major advantage of the hierarchical-ring topology is that it can be used to exploit the spatial locality of memory accesses often exhibited in parallel programs. As we will demonstrate in this chapter, this spatial locality property of hierarchical rings is critical to size scalability.

There are several ways we can build a hierarchical-ring network given a number of processors. Feasible configurations or topologies range from a tall, lean network to a short and wide network. However, only a few of these topologies tend to possess a high throughput, low latency combination, which should be the goal of any topology. In this chapter we describe a bottom-up approach in finding such topologies and discuss the effect that the bisection bandwidth has on the performance of such networks.

5.1 Deriving Optimal Hierarchical-ring Topologies

In this section we derive good high-performance hierarchical-ring topologies using flit-level simulations. We assume wormhole switching with 2 virtual channels per physical channel to avoid deadlock,[1] and choose optimal buffer sizes (as determined by our simulations) for both NIC ring

[1] Deadlock-free minimal routing for hierarchical rings using virtual channels is described in Chapter 7.

buffers and IRI buffers. However, it should be noted that the hierarchical-ring topologies we derive is independent of the switching technique or buffer sizes assumed. We use a bottom-up approach and start from the lowest level in the hierarchy and work up one level at a time. At the lowest level, we derive the maximum number of processors that can be sustained at high throughput and low latency and then fix that configuration. At higher levels, we derive the maximum number of next lower level rings of the previously set configuration that still gives high throughput and low latency.

We use the $T_{uniform}$ and T_{loc} workloads to simulate poor and high spatial locality in memory accesses, respectively. Given the fact that memory accesses of real applications tend to lie somewhere between these two extremes [79], the topology derived using the $T_{uniform}$ workload tends to be conservative, while the topology derived from the T_{loc} workload will be optimistic.

Our procedure is as follows. We start with a single ring, $L1$, and determine the maximum number of processors, $n_{L1,uniform}$ and $n_{L1,loc}$ that can be sustained with high throughput and low latency for the two workloads. The result consists of two optimal 1-level topologies for the two workload models. We then plot the maximum achievable throughput for both topologies for most memory access patterns that lie between $T_{uniform}$ and T_{loc}. From the throughput curves we evaluate the trade-off between the two topologies to determine a good $L1$ topology.

We then add the next level, $L2$, to the hierarchy and determine the appropriate number of $L1$ rings, $n_{L2,uniform}$ and $n_{L2,loc}$, that gives us high throughput, low latency topologies. We again evaluate the trade-off between these topologies and choose an $L2$ topology based on the maximum achievable throughput for different memory access patterns.

In the next step, we proceed to derive the number of $L2$ rings in a $L3$ ring for a high throughput, low latency 3-level hierarchy. As we go to higher levels of hierarchy, we find that the topology we obtain under the T_{loc} workload is the same as the topology we obtain under the $T_{uniform}$ workload. This signifies that even high locality in memory accesses cannot compensate for the constant bisection bandwidth constraints.

5.1.1 Single Rings

Here, we will show that a single ring can reasonably sustain a total of 8 processor-memory modules across most memory access patterns, and as we increase the cache line size, the effect of locality in the memory access pattern on system performance becomes less significant.

Figure 5.1a presents the throughput-latency curves for single ring topologies when subjected to the $T_{uniform}$ workload, while Figure 5.1b presents the throughput-latency curves for single rings when subjected to the T_{loc} workload. For the case with no locality, $n_{L1,uniform} = 4$ gives us a low latency configuration with high throughput compared to that of 8 and 16 processor

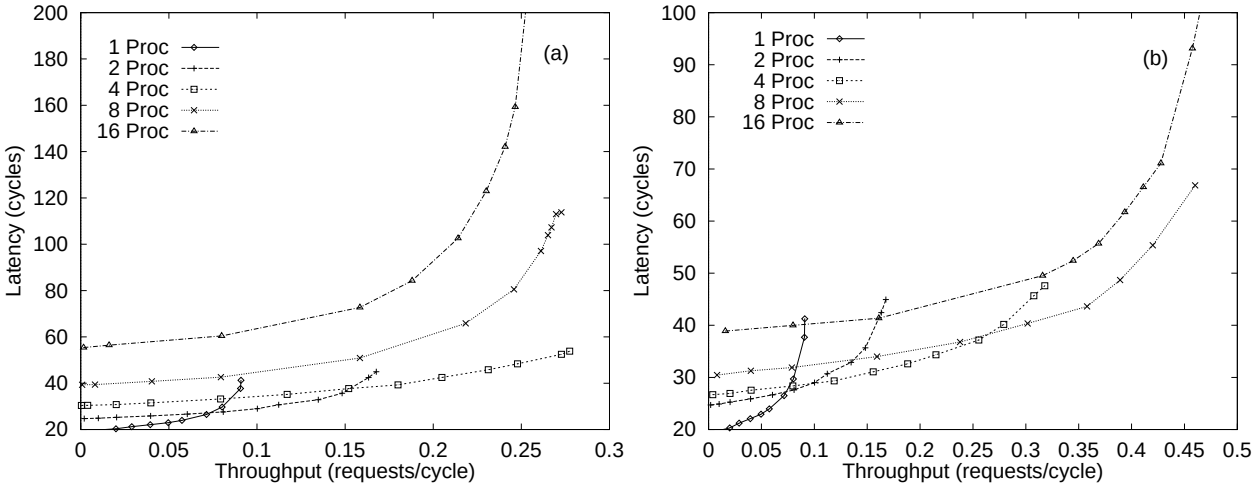

Figure 5.1: Throughput-latency curves for single ring topologies with 32B cache lines for (a) $T_{uniform}$ and (b) T_{loc} workloads.

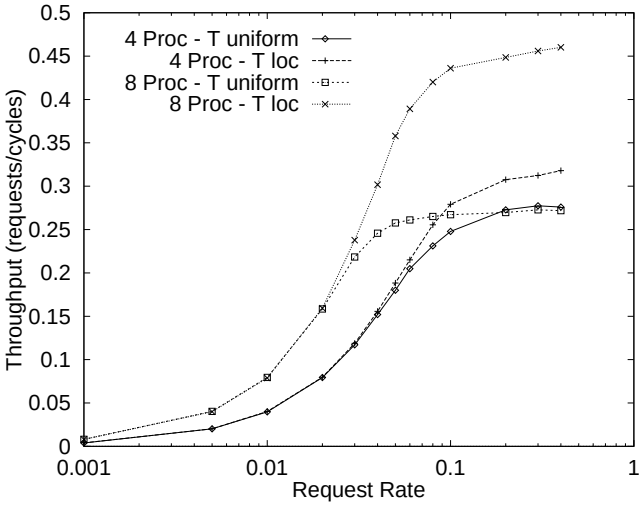

Figure 5.2: Throughput as a function of request rate for single ring 4 and 8 processor systems running the $T_{uniform}$ and T_{loc} workloads.

systems; initially when the number of processors is less than 4, performance is throughput limited and when we add more processors, throughput increases to a point after which performance becomes latency limited.

For the T_{loc} workload (Figure 5.1b), the maximum achievable throughput for the 8 processor ring is much higher than for the 4 processor ring; therefore, we choose $n_{L1,loc} = 8$, although the 4 processor configuration exhibits lower latency at lower request rates. For both workload models, however, the 16 processor configuration is clearly not desirable, as it exhibits higher latency when compared to the 8 processor topology.

Figure 5.2 presents the throughput versus request rate curves when subjected to the $T_{uniform}$

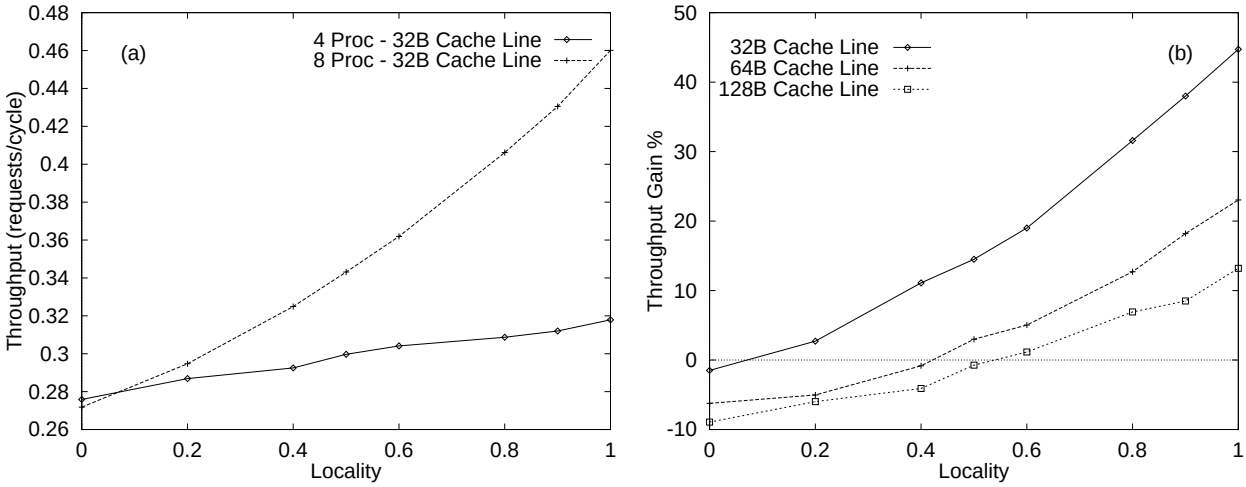

Figure 5.3: Maximum achievable throughput for 4 processor and 8 processor single ring systems. Figure (b) presents the throughput gain in percent of using an 8 processor system as opposed to a 4 processor system.

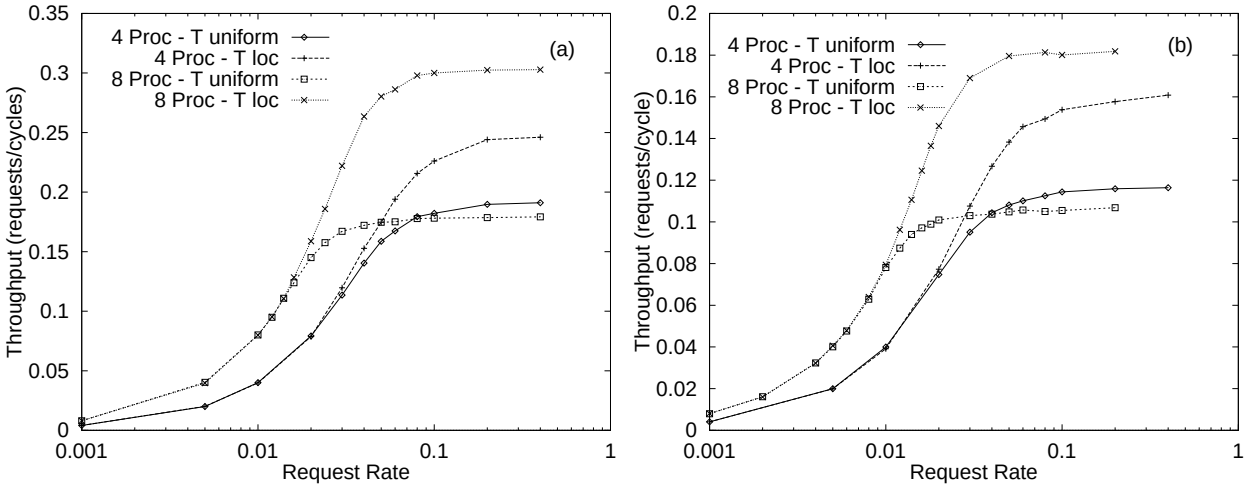

Figure 5.4: Throughput as a function of request rate for single ring 4 and 8 processor, (a) 64B and (b) 128B cache line systems running the $T_{uniform}$ and T_{loc} workloads.

and T_{loc} workloads for the two topologies $n_{L1,uniform}$ and $n_{L1,loc}$. For $n_{L1,loc}$, the maximum achievable throughput is 65% higher when there is high locality in the memory accesses than when there is poor locality. For $n_{L1,uniform}$ the difference is only 15%.

Given a workload model $P = (P_1, P_2, P_3)$, where P_1, P_2, and P_3 are the probabilities of memory accesses in the three clusters, as defined in Chapter 3, we then define the following equation for the locality in the memory access pattern,

$$locality = P_1 + (1 - P_1)P_2 \tag{5.1}$$

For $T_{uniform}$, $P_1 = \frac{1}{n}$ and $P_2 = 4 \cdot \frac{1}{n}$, where n is the total number of processors in the system,

and for T_{loc}, $P_1 = 0.5$ and $P_2 = 0.8$. Substituting these values in Equation 5.1, *locality* varies from $locality_{uniform} = \frac{1}{n} + (1 - \frac{1}{n})\frac{4}{n}$ for the $T_{uniform}$ workload to $locality_{loc} = 0.9$ for the T_{loc} workload. We normalize *locality* values that lie between $locality_{uniform}$ and $locality_{loc}$ to lie between 0 and 1 with the following equation,

$$locality = \frac{locality - locality_{uniform}}{locality_{loc} - locality_{uniform}} \tag{5.2}$$

Figure 5.3a presents the maximum achievable throughput (for locality in memory access patterns that lies between $T_{uniform}$ and T_{loc}) for the two topologies. The locality scale is normalized from 0 (that corresponds to $T_{uniform}$) to 1 (that corresponds to T_{loc}). It is clear that the 8 processor topology results in higher throughput for most memory access patterns.

Figure 5.3b presents the maximum throughput gain in percent by using a 8 processor topology ($n_{L1,loc}$) as opposed to a 4 processor topology ($n_{L1,uniform}$) for different degrees of locality in memory accesses. It is obvious that there is a positive throughput gain (as high as 45%) for most memory access patterns by using the 8 processor topology. The trend is similar for larger cache line sizes, where there is still a gain in the maximum achievable throughput when using a 8 processor topology although it is much less for 64 and 128-byte cache line systems than for the 32-byte cache line system.[2] Figure 5.4 presents the throughput versus request rate for single ring 4 and 8 processor topologies for systems with 64-byte and 128 byte cache lines.

The main conclusions in this section can be summarized as follows:

1. a total of 8 processor-memory modules can be reasonably sustained in a single ring across most memory access patterns, and

2. as we increase the cache line size, the effect of locality in the memory access pattern on system performance becomes less significant.

5.1.2 Two-level Rings

In this section we will show that a total of 5 local rings can be reasonably sustained in a two-level hierarchical-ring topology for most memory access patterns and that the effect of locality in memory accesses on system performance is independent of cache line size for systems of this size. To do so, we add a second level ring, $L2$, and determine how many $L1$ local rings a two level hierarchy can sustain. The $L2$ *global ring* connects a number of $L1$ *local rings*, each containing the maximum number of processor-memory modules ($n_{L1} = 8$), as determined in the previous section. Figure 5.5 presents the throughput-latency curves for 2-level hierarchical

[2]The ring size $n_{L1,uniform}$ and $n_{L1,loc}$ remain the same at 4 and 8 nodes for 64 and 128-byte cache line sizes.

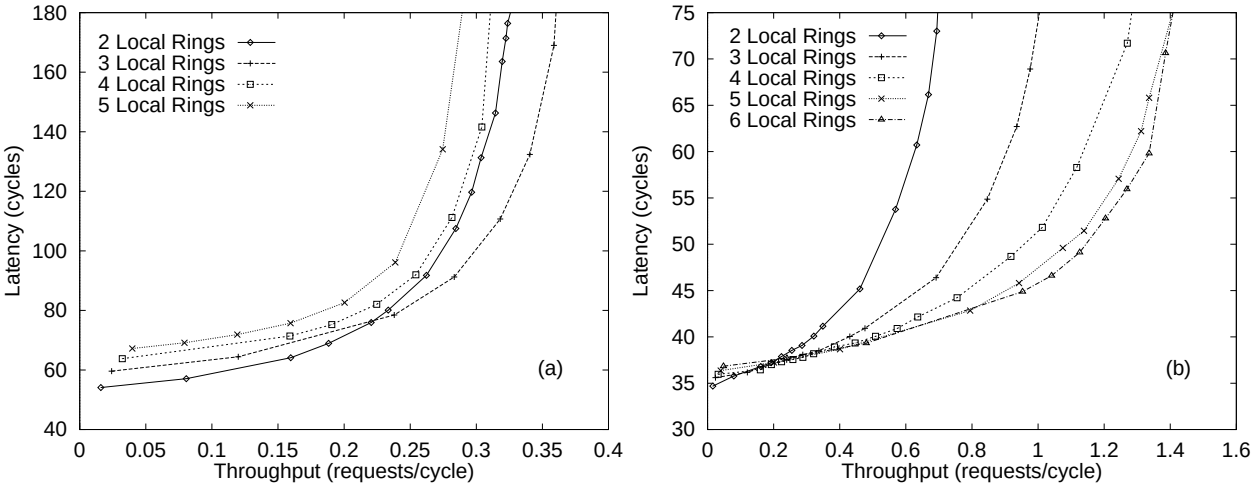

Figure 5.5: Throughput-latency curves for two level ring topologies with 32B cache lines for the (a) $T_{uniform}$ and (b) T_{loc} workloads.

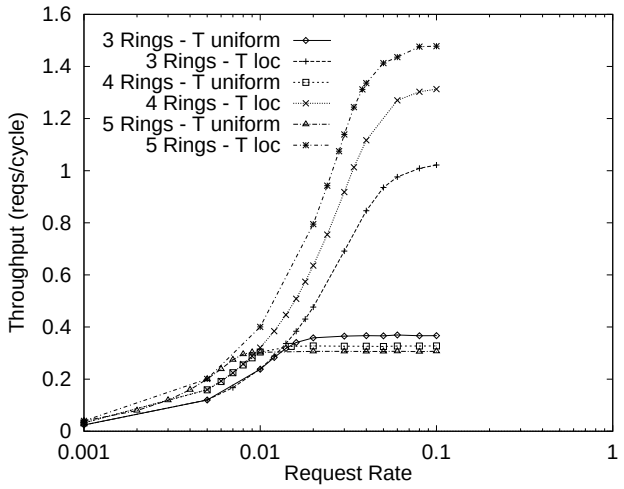

Figure 5.6: Throughput as a function of request rate for two level ring topologies with 32B cache lines running $T_{uniform}$ and T_{loc} workloads

rings with 32-byte cache lines. With the $T_{uniform}$ workload, a global ring can sustain only 3 ($n_{L2,uniform} = 3$) local rings; any increase in the number of local rings decreases the maximum achievable throughput of the network. However, with the T_{loc} workload that has high locality, we can increase the number of local rings to 5 ($n_{L2,loc} = 5$). In the latter case, when the number of local rings is further increased, there is no significant increase in the maximum achievable throughput.

One major difference between single ring and two-level ring topologies is the effect of locality on the maximum achievable throughput. For T_{loc}, the maximum achievable throughput with $n_{L2,loc}$ (8×5 topology) is about 500% higher than for $T_{uniform}$. This is shown in Figure 5.6,

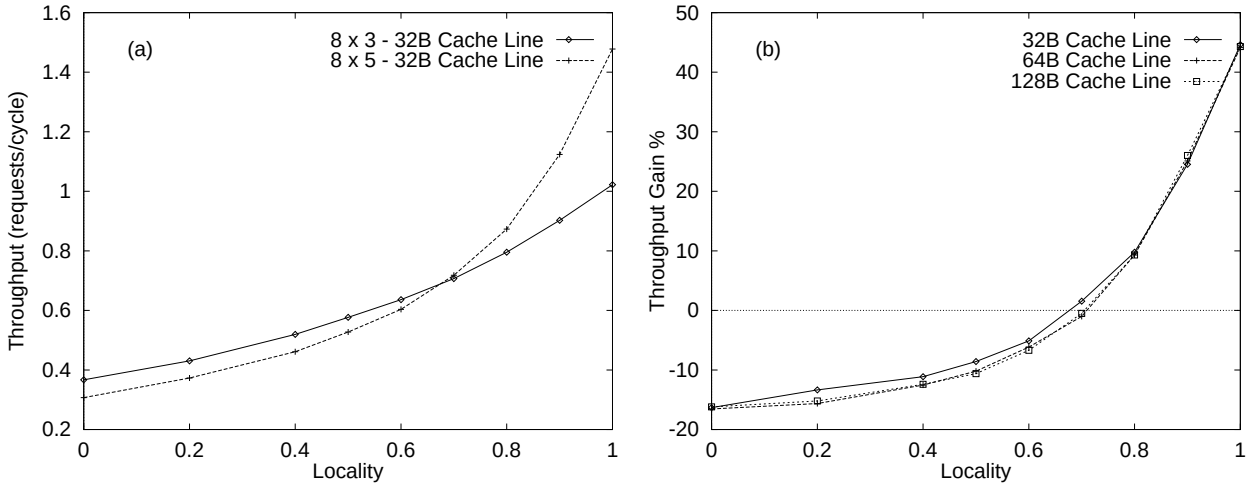

Figure 5.7: Maximum achievable throughput for the 8×3 and 8×5 topologies. Figure (b) presents the throughput gain in percent of using a 8×5 topology as opposed to an 8×3 topology.

which plots the throughput as a function of the request rate for two-level ring topologies. The difference in throughput is highest for $n_{L2,loc}$.

Figure 5.7a plots the maximum achievable throughput as a function of normalized locality (for memory access patterns that lie between $T_{uniform}$ and T_{loc}). Figure 5.7b presents the throughput gain in percent when using an 8×5 topology as opposed to an 8×3 topology. We see that by using the 8×5 topology there is a throughput loss for most memory access patterns; however, this loss is small and decreases as locality is increased. The throughput gain starts to grow at a higher rate when $locality > 0.6$, resulting in a 45% throughput gain when $locality = 1$. It should be noted it is possible for real applications to have more locality than $locality = 1$, resulting in an even higher throughput gain. Since the throughput gain by using an 8×5 topology is much higher at higher locality levels than the throughput loss at lower locality levels, we can reasonably assume that the number of local rings a second-level global ring can sustain is 5. An important observation is that unlike the single ring case, the effect of locality on the throughput gain remains independent of the cache line size.

Figure 5.8 presents the global ring utilization for the 2-level rings plotted as a function of the number of local rings. Two curves are shown for each workload model: one for a low request rate and one for a high request rate. It is clear that when subjected to a low request rate, the global ring utilization increases linearly irrespective of the nature of workload. This is not true for high request rates, where the global ring utilization rises much faster when local rings are added, resulting in an early saturation. For the $T_{uniform}$ workload, the global ring saturates at the point where 3 local rings are connected, whereas for T_{loc} it takes 5 local rings to saturate.

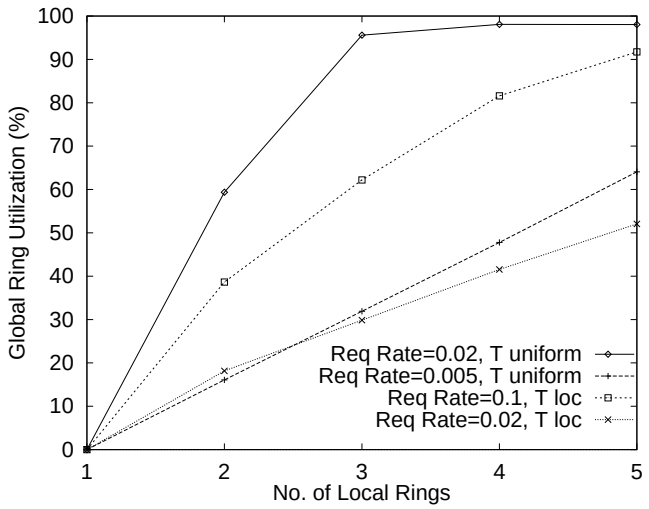

Figure 5.8: Global ring utilization for 2-level rings with 32-byte cache lines. Two curves are shown for the $T_{uniform}$ and T_{loc} workloads at both high and low request rates.

The trend is similar for 64 and 128-byte cache line sizes (but not shown).

The main conclusions from this section can be summarized as follows:

1. the global ring of a two level hierarchy can reasonably sustain 5 local rings for most memory access patterns, and

2. the effect of locality on system throughput is independent of the cache line size used.

5.1.3 Three-level Rings

We next introduce a third level to the hierarchy and proceed to determine how many $L2$ rings can be sustained. Each $L2$ ring now consists of a second level ring connected to 5 $L1$ rings of 8 nodes each, for a total of 40 nodes. We refer to the third level ring as the global ring.

Figure 5.9 presents the throughput-latency curves for 3-level hierarchical rings with 32-byte cache lines. For $T_{uniform}$, the trend is similar to what we observed for 2-level rings, namely that a maximum of 3 $L2$ rings ($n_{L3,uniform} = 3$) can be sustained by a global ring. However, for T_{loc}, we are only able to sustain 3 $L2$ rings ($n_{L3,loc} = 3$). The constant bisection bandwidth constraint of the hierarchical-ring network offsets the benefits of high locality in the memory accesses. Thus, even good locality (where in this case 90% of all requests lie within a 4 neighbor cluster) saturates the global ring fairly easily.

The effect of bisection bandwidth on performance is shown in Figure 5.10, which plots the global and local rings utilization against the number of $L2$ rings in a 3-level ring hierarchy. The trend is similar to that of the 2-level ring hierarchy, although the rate at which the global ring utilization rises is higher. Also, in the 3-level ring hierarchy, the global ring saturates when

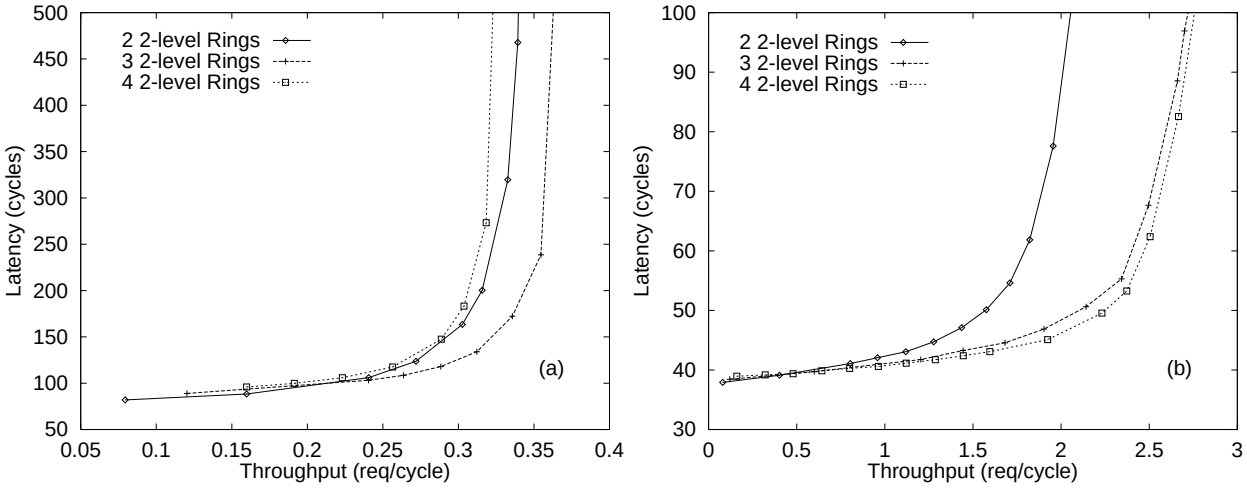

Figure 5.9: Throughput-latency curves for three level rings with 32B cache lines for the (a) $T_{uniform}$ and (b) T_{loc} workloads.

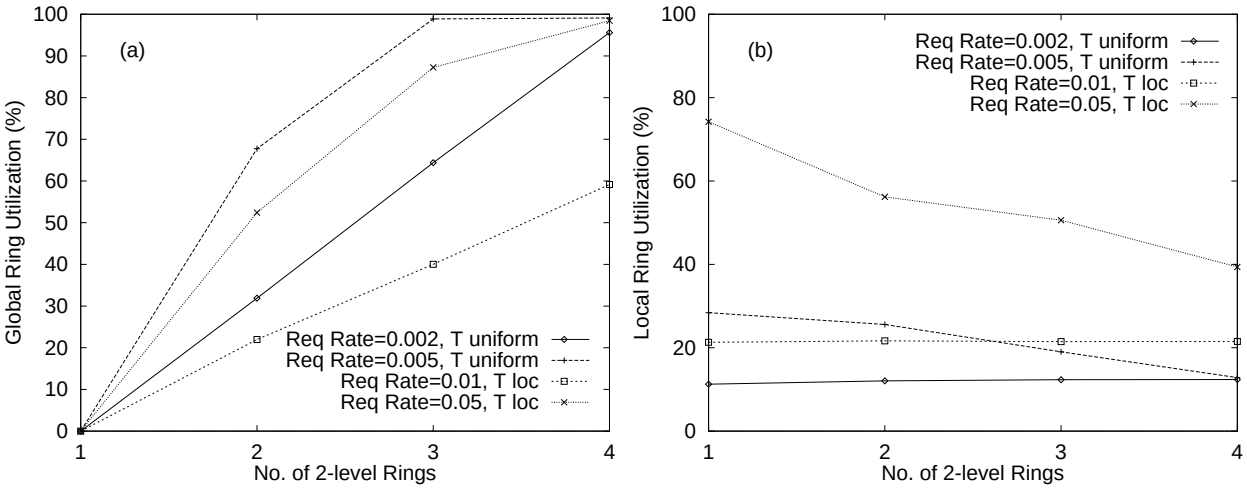

Figure 5.10: (a) Global and (b) local ring utilization for three-level hierarchical rings with 32B cache lines. Curves are shown for $T_{uniform}$ and T_{loc} workloads and for high and low request rates.

$n_{L3} = 3$ irrespective of the degree of locality. The fact that the local ring utilization is low at the point where the global ring saturates, as shown in Figure 5.10b, further strengthens our argument that the performance of the network is bisection bandwidth limited.

5.1.4 Verification

To verify that the topologies derived in the previous sections result in high throughput, low latency configurations, we compare three different topologies of a 64 processor, 32-byte cache line system. The three topologies include:

- a $8 \times 4 \times 2$ topology derived from our results,

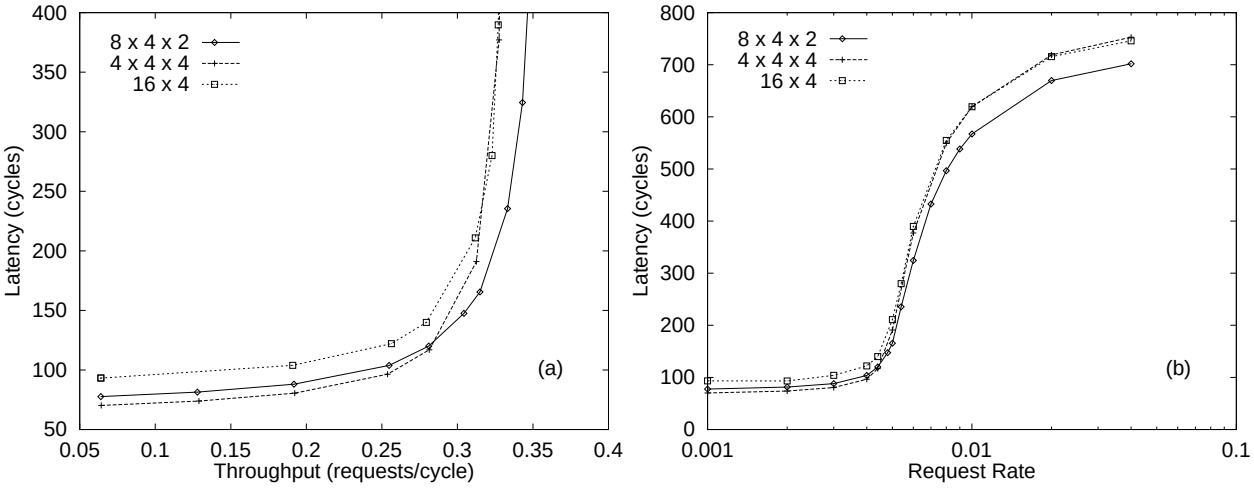

Figure 5.11: Comparing the "optimal" topology with two other topologies for a 64 processor, 32-byte cache line system with the $T_{uniform}$ workload. (a) Throughput-latency and (b) latency as a function of request rate curves are shown.

- a balanced $4 \times 4 \times 4$ hierarchy, and
- a 16×4 with large local rings.

Figure 5.11a presents the throughput-latency curves for these configurations. It is clear that even though at low request rates the balanced $4 \times 4 \times 4$ topology has lower latency, the $8 \times 4 \times 2$ hierarchy achieves highest throughput. This is because, at low request rates, in the absence of contention, the network with a lower diameter has lower latency. However, the advantage of lower diameter begins to disappear as contention sets in at higher request rates, as shown in Figure 5.11b.

To further verify this, we compare the execution time of real applications in Figure 5.12 for the three topologies. The applications are from the SPLASH-2 suite and consist of FFT, LU, Radix, Raytrace and Ocean. We observe that for these applications there is no significant difference in the execution time. However, from Figure 5.13, which plots the average memory access latency we see that a balanced network topology ($4 \times 4 \times 4$) has the lowest average latency, followed by the $8 \times 4 \times 2$ topology. The 16×4 topology performs worst. These results confirm our findings obtained from our synthetic workload simulations at low request rates.

5.2 Effect of Critical Parameters on Performance

In this section we develop a simple analytical model to study the effect of certain critical parameters on the performance of hierarchical-ring topologies. The analytical model is semi-empirical in that it uses some input parameters derived from simulations. This semi-empirical

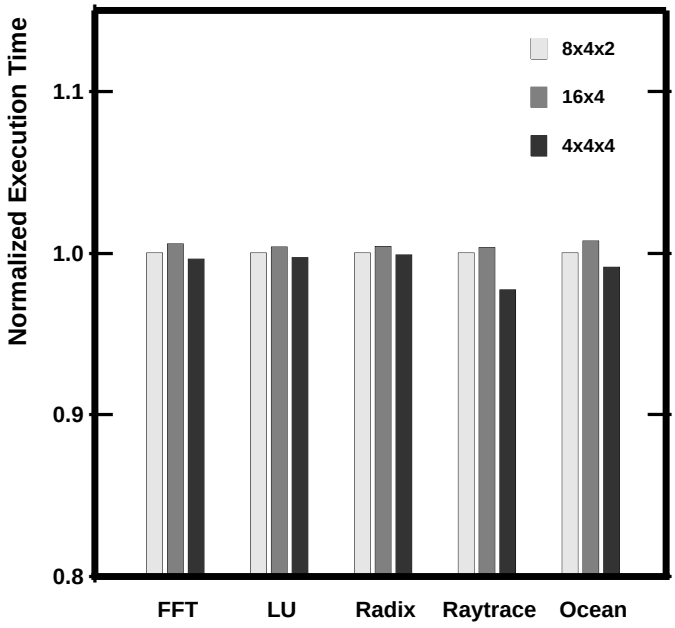

Figure 5.12: Normalized execution time of five SPLASH applications for three different topologies of 64 processor, 32-byte cache line systems: (1) a 3-level $8 \times 4 \times 2$ system, (2) a 2-level 16×4 system, and (3) a 3-level $4 \times 4 \times 4$ system. The execution time is normalized to the 3-level $8 \times 4 \times 2$ system.

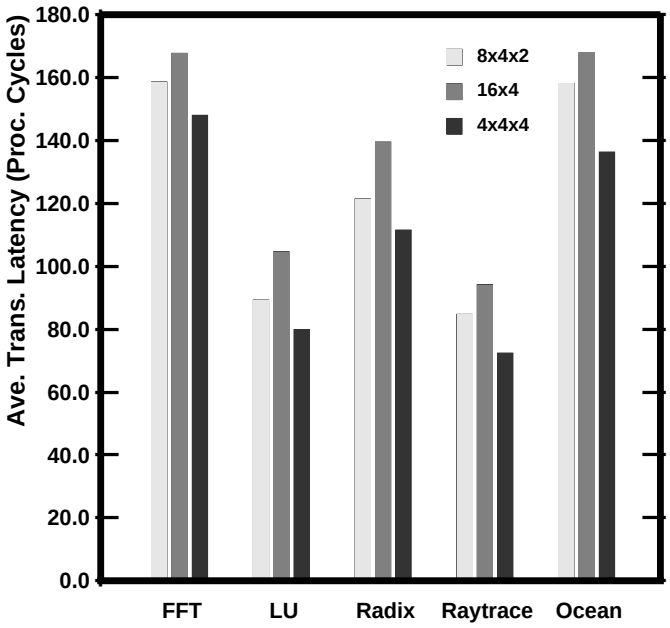

Figure 5.13: Average memory access latency (after L2 cache miss) of the five SPLASH applications under three different 64 processor, 32-byte cache line system topologies. Since these applications have low (network) request rates, the latency of transactions is sensitive to the diameter of the particular topology.

model allows us to save much simulation time and is useful for determining which part of the design space should be simulated for more accurate predictions. The parameters we consider in our model include router speed, channel width and the number of processors per node. In particular, we define the following parameters:

λ	= cache miss rate or the processor request rate.
λ_{max}	= processor request rate beyond which the network goes into saturation.
λ_{lm}	= fraction of λ to the local memory.
λ_{lr}	= fraction of $\lambda(1-\lambda_{lm})$ to a processor on the local ring.
λ_{gr1}	= fraction of $\lambda(1-\lambda_{lm})(1-\lambda_{lr})$ that stays with in the 2-level ring hierarchy.
S_{proc}	= processor speed in cycles/second.
S_{nic}	= NIC speed in cycles/second.
S_{iri}	= second-level IRI router speed in cycles/second.
S_{glb_iri}	= third-level IRI router speed in cycles/second.
n_{Li}	= number of next lower level ring connected to an ith level ring, when $i=1$, n_{L1} = number of nodes in a local ring. when $i=2$, n_{L2} = number of local rings connected to a second level ring. when $i=3$, n_{L3} = number of 2-level rings connected to a third level ring.
W	= Channel width.
L_{trans}	= average length of a memory transaction (request and response) in bits.

5.2.1 Single Rings

In the first step, we develop the model for a single ring and extend it to include a second and a third level. Let us assume for simplicity a single processor per node. The traffic, m in bits/sec, injected by a node into the ring depends on the cache miss rate (the processor request rate), λ, the fraction of requests that go to local memory, λ_{lm}, the average length of a transaction, L_{trans}, and the processor speed, S_{proc}:

$$m = \lambda \cdot (1-\lambda_{lm}) \cdot L_{trans} \cdot S_{proc} \tag{5.3}$$

Assuming the $T_{uniform}$ workload, the average load at any point in the ring will be $m \cdot \frac{n_{L1}}{2}$, since a packet typically (on average) traverses half the ring. We refer this as the *bisection load*. For the bisection load to be less than or equal to the bisection bandwidth it is necessary that:

$$m \cdot \frac{n_{L1}}{2} \leq 2 \cdot W \cdot S_{nic} \tag{5.4}$$

Substituting for m from Equation 5.3 we have:

$$\lambda \cdot (1-\lambda_{lm}) \cdot L_{trans} \cdot S_{proc} \cdot \frac{n_{L1}}{2} \leq 2 \cdot W \cdot S_{nic} \tag{5.5}$$

which can be rewritten as:

$$\lambda \cdot (1-\lambda_{lm}) \cdot \frac{S_{proc}}{S_{nic}} \cdot \frac{L_{trans}}{W} \cdot n_{L1} \leq 4 \tag{5.6}$$

If we define S_{ratio} as the ratio of processor and NIC router speeds, $\frac{S_{proc}}{S_{nic}}$, and n_{phits} (number of physical transfer units) as the ratio of the average length of a transaction to the channel width, $\frac{L_{trans}}{W}$, Equation 5.6 can be rewritten as:

$$\lambda \cdot (1 - \lambda_{lm}) \cdot S_{ratio} \cdot n_{phits} \cdot n_{L1} \leq 4 \tag{5.7}$$

For the $T_{uniform}$ workload, $\lambda_{lm} = \frac{1}{n_{L1}}$, so equation 5.7 is reduced to:

$$\lambda \cdot (n_{L1} - 1) \cdot S_{ratio} \cdot n_{phits} \leq 4 \tag{5.8}$$

Therefore, the maximum processor request rate in a single ring, given the number the nodes for which the network does not saturate, can be derived from the following:

$$\lambda_{max(1-level)} = 4 \cdot \frac{1}{n_{phits}} \cdot \frac{1}{n_{L1} - 1} \cdot \frac{1}{S_{ratio}} \tag{5.9}$$

In other words, to keep a single ring network below saturation, a processor's cache miss rate should be at most the value defined in Equation 5.9. It should be noted that this value is inversely proportional to the average length of a transaction (n_{phits}), the ratio of processor and NIC router speeds (S_{ratio}), and the number of nodes in a ring.

5.2.2 Additional Ring Levels

For two levels of rings, the equivalent of equation 5.4 is:

$$m_{L2} \cdot \frac{n_{L2}}{2} \leq 2 \cdot W \cdot S_{iri} \tag{5.10}$$

where m_{L2} is the request rate from a local ring into the global ring, n_{L2} is the total number of local rings, and S_{iri} is the inter-ring interface router speed. m_{L2} can be defined in the same way as in Equation 5.3:

$$m_{L2} = n_{L1}[\cdot(\lambda \cdot (1 - \lambda_{lm}) \cdot L_{trans} \cdot S_{proc} \cdot \frac{1}{S_{ratio}})] \cdot (1 - \lambda_{lr}) \tag{5.11}$$

Substituting $\frac{S_{proc}}{S_{nic}}$ for S_{ratio} and expanding equation 5.10 using equation 5.11 we obtain:

$$\lambda \cdot (1 - \lambda_{lm}) \cdot (1 - \lambda_{lr}) \cdot n_{phits} \cdot \frac{S_{nic}}{S_{iri}} \cdot n_{L2} \cdot n_{L1} \leq 4 \tag{5.12}$$

If we define S_{lcl_ratio} as the ratio of the NIC and IRI router speeds, $\frac{S_{nic}}{S_{iri}}$, and substitute $\frac{1}{n_{L2}}$ for λ_{lr} and $\frac{1}{n_{L1}}$ for λ_{lm} for the $T_{uniform}$ workload, we can obtain the maximum processor request

rate in a two level ring system from:

$$\lambda_{max(2-level)} = 4 \cdot \frac{1}{n_{L1} - 1} \cdot \frac{1}{n_{L2} - 1} \cdot \frac{1}{n_{phits}} \cdot \frac{1}{S_{lcl\_ratio}} \tag{5.13}$$

For 3-level rings, we can proceed similarly and derive the following equation for the maximum processor request rate:

$$\lambda_{max(3-level)} = 4 \cdot \frac{1}{1 - \lambda_{lm}} \cdot \frac{1}{1 - \lambda_{lr}} \cdot \frac{1}{1 - \lambda_{gr}} \cdot \frac{1}{n_{phits}} \cdot \frac{1}{S_{glb\_ratio}} \cdot \frac{1}{n_{L1}} \cdot \frac{1}{n_{L2}} \cdot \frac{1}{n_{L3}} \tag{5.14}$$

where $(1 - \lambda_{gr})$ is the fraction of packets in the second level that travels to the third level, $S_{glb\_ratio} = \frac{S_{iri}}{S_{glb\_iri}}$ is the ratio of second-level IRI and the third-level global IRI router speeds, and n_{L3} is the number of second-level rings connected to the third-level global ring.

5.2.3 Assessment of the Model

To assess the accuracy of our model, we compare λ_{max} obtained from equations 5.9, 5.13 and 5.14 to the simulation results obtained in Section 4.2. Since $S_{ratio} = 2$, n_{phits}= 4, 6, and 10 for 32, 64 and 128-byte cache lines, respectively, and $n_{L1} = 8$, $n_{L2} = 5$ and $n_{L3} = 3$ in our simulations, we substitute these values to obtain λ_{max} predicted by the model. Table 5.1 presents the maximum processor request rates, as obtained from our simulations and from the model. The model generally follows the maximum processor request rates obtained from our simulations, but generally overestimates the values. The values are overestimated, because network contention is not captured in our model. Network contention can be accounted for by multiplying the model output with a contention factor (see Table 5.1). The contention factor is computed as the ratio of the average of maximum processor request rates (for all cache line sizes) obtained from the simulation to that obtained from the model. We observe that this contention factor decreases by a factor of 2 as we increase the number of levels in the hierarchy.

An interesting property of (contention-free) maximum processor request rates is that they decrease by a factor of two for every increase in the number of levels in the hierarchy. Dividing equations 5.14 and 5.13, the ratio of the maximum processor request rates for a 3-level and a 2-level hierarchy is given by,

$$\frac{\lambda_{max(3-level)}}{\lambda_{max(2-level)}} = \frac{1}{1 - \lambda_{gr}} \cdot \frac{1}{S_{glb\_ratio}} \cdot S_{lcl\_ratio} \cdot \frac{1}{n_{L3}} \tag{5.15}$$

Substituting $\lambda_{gr} = \frac{1}{n_{L3}}$, $S_{glb\_ratio} = S_{lcl\_ratio} = 1$ and $n_{L3} = 3$ from our simulation results, we obtain

$$\frac{\lambda_{max(3-level)}}{\lambda_{max(2-level)}} = 0.5 \tag{5.16}$$

Hierarchy	Cache Line Size	Max Processor Request Rate			Contention Factor
		Simulation	Model	Model (adjusted)	
Single-ring	32B	0.07	0.0714	0.063	0.88
	64B	0.04	0.0476	0.042	
	128B	0.02	0.0286	0.025	
2-level	32B	0.02	0.0357	0.0157	0.44
	64B	0.008	0.02381	0.011	
	128B	0.005	0.01429	0.0063	

Table 5.1: Maximum processor request rates for single and 2-level rings obtained from the model and simulation for 32, 64 and 128-byte cache lines.

Similarly, the ratio of the maximum processor request rates for a 2-level hierarchy and a single-ring is given by dividing equations 5.13 and 5.11,

$$\frac{\lambda_{max(2-level)}}{\lambda_{max(1-level)}} = \frac{1}{1-\lambda lr} \cdot \frac{1}{S_{lcl\_ratio}} \cdot S_{ratio} \cdot \frac{1}{n_{L2}} \tag{5.17}$$

Substituting $\lambda_{lr} = \frac{1}{n_{L2}}$, $S_{lcl\_ratio} = 1$, $S_{ratio} = 2$, and $n_{L2} = 5$, we obtain,

$$\frac{\lambda_{max(2-level)}}{\lambda_{max(single-ring)}} = 0.5 \tag{5.18}$$

We can, for example, use this property to obtain the number of lower level rings a global ring can sustain for a level higher than 3. For example, in a 4-level hierarchical-ring network, we know that the maximum processor request rate $\lambda_{max(4-level)}$ will be half of that in a 3-level hierarchy. Therefore,

$$\frac{\lambda_{max(4-level)}}{\lambda_{max(3-level)}} = \frac{1}{1-\lambda_{gr1}} \cdot \frac{1}{S_{glb1\_ratio}} \cdot S_{glb\_ratio} \cdot \frac{1}{n_{L4}} = 0.5 \tag{5.19}$$

Substituting $\lambda_{gr1} = \frac{1}{n_{L4}}$, $S_{glb1\_ratio} = S_{glb\_ratio} = 1$, and solving for n_{L4}, we obtain,

$$n_{L4} = \frac{\lambda_{max(3-level)}}{\lambda_{max(4-level)}} + 1 = 3 \tag{5.20}$$

Therefore, we can sustain up to 3 $L3$ rings in a 4-level hierarchy.

5.2.4 Effect of Router Speeds on Performance

As shown earlier, the performance and scalability of hierarchical rings are clearly limited by their constant bisection bandwidth. By increasing the bandwidth of the global ring (and thus

the bisection bandwidth), we can connect additional lower level rings without worsening the average memory access latency. Targeting just the global ring is effective, because the utilization of the lower level rings is low, especially when the global ring is saturated. The bandwidth of the global ring can be increased either by increasing the width of the ring or the speed of the ring. We explore the option of clocking the global ring at a speed higher than that of local and intermediate rings.

For 2-level rings, we use Equation 5.13 to obtain n_{L2}, the maximum number of local rings connected to a global ring,

$$n_{L2} = 4 \cdot \frac{1}{\lambda_{max(2-level)}} \cdot \frac{1}{n_{L1} - 1} \cdot \frac{1}{n_{phits}} \cdot \frac{1}{S_{iri\_ratio}} + 1 \tag{5.21}$$

If the global ring is twice as fast as the local rings then $S_{iri\_ratio} = \frac{S_{nic}}{S_{iri}} = 0.5$. Dividing equation 5.21 by itself, we obtain:

$$\frac{n_{L2(Siri\_ratio=0.5)} - 1}{n_{L2(Siri\_ratio=1)} - 1} = 2 \tag{5.22}$$

Substituting $n_{L2(iri\_ratio=1)} = 5$ from our simulation results and solving for $n_{L2(iri\_ratio=0.5)}$, we obtain:

$$n_{L2(iri\_ratio=0.5)} = 9 \tag{5.23}$$

From this we conclude that a 2-level hierarchical ring can sustain up to 9 local rings when the global ring is twice as fast as the local rings. For 3-level rings, equation 5.22 becomes,

$$\frac{n_{L3(glb\_ratio=0.5)} - 1}{n_{L3(glb\_ratio=1)} - 1} = 2 \tag{5.24}$$

Since $n_{L3(glb\_ratio=1)} = 3$ and $n_{L3(glb\_ratio=0.5)} = 5$, the global ring in a 3-level hierarchy can sustain up to 5 second-level rings when it is clocked at twice the speed of local rings.

5.2.5 Effect of Channel Width on Performance

The channel width affects the number of physical transfer units that must be transferred per packet. Our basic assumption has been a 128-bit (16-byte) wide ring channels, which requires transfers of 2, 4, or 8 phits for messages containing a 32, 64 and 128-byte cache line (excluding header phits), respectively. Any reduction in channel width increases the number of phits required and therefore reduces the maximum processor request rate for the same network configuration. To determine the performance impact of a reduction in channel width by a factor

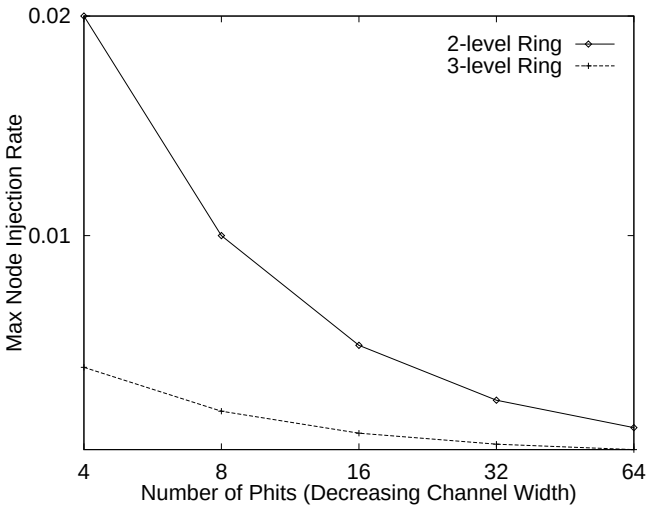

Figure 5.14: Effect of decreasing channel width on maximum processor request rate. The graph is shown for 2-level and 3-level rings with 32-byte cache lines.

of 2, we divide the two equations derived from 5.21 by setting n_{phits} in the numerator to 4 and n_{phits} in the denominator to 8 and obtain:

$$\lambda_{max(phits=8)} = 0.5 \cdot \lambda_{max(phits=4)} \tag{5.25}$$

Thus the maximum processor request rate is reduced by a factor of 2, when the channel width is reduced by a factor of 2. Figure 5.14 shows the impact of decreasing channel width on the maximum processor request rate.

Chapter Summary

This chapter presented techniques to derive high-performance topologies for hierarchical-ring networks. Our overall goal was to maximize system throughput. From this point on, in this dissertation, when dealing with hierarchical-ring systems, we assume the following topologies: up to 8 processors on a level-1 ring, a maximum of 5 level-1 rings in a 2-level hierarchy, and a maximum of 3 level-2 rings in a 3-level hierarchy. For mesh and torus-connected systems, we assume 2-dimensional square topologies.

We also showed in this chapter that single rings and 2-level hierarchical-ring topologies are more sensitive to locality in memory accesses, whereas higher level hierarchical-ring topologies are less sensitive. We presented some semi-empirical analytical models to explore design spaces not considered in our simulations.

Chapter 6
Switching, Buffer Management, and Flow-control

This chapter presents switching and buffer management issues in hierarchical-ring and direct networks. In particular, we will show how different switching techniques affect system performance as does the network buffer size. For hierarchical-ring networks, we consider wormhole, buffered wormhole, virtual cut-through, and cell switching, whereas for direct networks we consider wormhole and buffered-wormhole switching techniques. We consider blocking flow-control for wormhole switched networks and non-blocking flow-control for virtual cut-through and cell switched hierarchical-ring networks. We also propose and evaluate a blocking cell switching scheme in hierarchical-ring networks. We assume deterministic, deadlock-free routing with virtual channels in hierarchical-ring, tori, and bidirectional ring networks and dimension-ordered routing (with no virtual channels) in the mesh networks. Routing issues and in particular, deadlock-free routing are explained in Chapter 7.

One advantage of unidirectional rings, as pointed out in Chapter 3, is that the routers require fewer connections (due to the single dimension), allowing for a wider data path than in higher-dimensional direct networks under constant pin constraints, and thus shorter packet switching time. Another advantage is that the single network input link at the router allows larger ring buffers than in higher-dimensional networks under constant memory constraints.

Wormhole routers for distributed memory systems predominantly use single-flit buffers [53, 78, 85]. For shared-memory multiprocessors, we extended this to *buffered wormhole switching* [71], where the routers have buffers that are several flits large as opposed to just one flit. This definition of buffered wormhole switching is irrespective of a flit containing one or more phits. The buffered wormhole switching approach significantly improves system throughput and reduces latency, primarily because the number of links held by a blocked packet is vastly reduced. Also, since both long packets (containing cache lines) and short packets (containing acknowledgments or data-free requests) co-exist in a shared-memory multiprocessor network, even a buffer size of a few flits is effective in improving throughput, as it can buffer smaller packets in entirety.

6.1 Switching Techniques

Switching is the mechanism by which a router removes a packet from its input link and places it in an output link, thereby allocating channels and buffers to the packet as it travels through the network. We consider four switching techniques: wormhole (WH), buffered wormhole (BWH), virtual cut-through (VCT), and cell switching.

Wormhole and virtual cut-through are common switching techniques used in n-dimensional meshes and k-ary n-cubes [6, 55, 57, 80, 81], but we are aware of no previous work that considers these switching techniques in ring connected networks. Cell switching is different from either wormhole or virtual cut-through switching in that packets are divided into cells that are routed independently; it has been used in both single rings (typically under the name "slotted ring") and hierarchical-ring networks [92, 93].

6.1.1 Wormhole and Buffered-wormhole Switching

In wormhole switching (see Chapter 2), a packet is sent as a sequence of flits, with the header flit containing the routing information. A flit is the smallest unit a router can accept or refuse. Since there is no distinction between a flit and a phit in our study, a flit is transferred in a single clock cycle between any two neighboring nodes. As flits are forwarded, a packet may be spread over multiple links, and a packet is hence sometimes referred to as a *worm* in this context. Since only the head flit of a packet contains routing information, it is essential that the flits of a packet not be interleaved with flits of another packet. Packets are of variable size, with the size of the largest packet (and hence the longest worm) assumed to be large enough to hold a cache line.

A flit cannot move forward if the link it is to use next is busy (which can only happen for the header flit) or if the buffers of the next router in the path are full (which can happen for any of the flits). In either of these cases, the flit is blocked. Traditionally, wormhole routers are associated with single flit buffers on input links. As a result, a blocked flit will block the following flits, and when the entire packet becomes blocked, it typically spans multiple nodes, continuing to hold resources (buffers and links) in the nodes it spans. We will show later that having buffers larger than just a single flit can decrease the network latency and improve throughput. In this *buffered wormhole* approach, when a flit of a packet becomes blocked, the router continues to accept additional flits of the packet (if any) over the incoming link until the input buffers associated with the link become full; only at that point does the router stop accepting further flits.

6.1.2 Virtual Cut-through Switching

Virtual cut-through switching is similar to wormhole switching in that a packet is sent as a sequence of flits that may not be interleaved with other packets and the header flit containing routing information. A virtual cut-through node, however, will accept the header flit of a packet only if it has enough buffer space for the entire packet. Thus, when a packet is blocked, it is removed from the network and buffered inside a single node. This requires large ring buffer spaces in the routers. In a blocking virtual cut-through network, flow-control is almost always sender initiated in that a node keeps track of the free buffer space in directly connected neighboring nodes, and it does not forward a packet to a neighbor when there is not enough buffer space there. Sender initiated blocking is used, because the variable-sized packets require a node to know the size of the packet before the header can be sent.

Non-blocking virtual cut-through switching

We have adapted blocking virtual cut-through switching to realize non-blocking virtual cut-through switching for hierarchical-ring networks. A non-blocking virtual cut-through node drops a packet (instead of blocking) when it cannot forward it to a neighboring node. In hierarchical rings, packets are dropped only at the inter-ring interfaces (IRIs) when they need to change rings because in network interface controllers (NICs), priority for the output link assignment is given to transit packets, and in IRIs, priority is given to packets that do not change rings.

Dropped packets are recovered by using NACK packets and end-to-end time-outs. When a request packet is dropped, it is recovered by sending a negative acknowledgment (NACK) packet to the source node, and when a response packet is dropped it is recovered through time-out. When a request packet is sent, the source NIC keeps a copy of the request and starts a timer. If a response is received before the timer expires, then the timer is cleared and the copy is discarded. If the source NIC receives a NACK for its request, then it resends the request and resets the timer. If the timer expires, then it is assumed that either the response packet or that the NACK packet had to be dropped, and the request is resent with simultaneous resetting of the timer. Since the worst-case latency value for a memory request is non-deterministic, time-out values are normally chosen to be very large (two orders of magnitude greater than the average round trip time) since a smaller value will introduce duplicate request packets in the network.

6.1.3 Cell Switching

Cell switching is a variant of virtual cut-through switching, where a packet is divided into a number of equi-sized cells that are routed independently. In a single ring, cell switching is the same as the slotted ring. Cells are similar to flits except that each cell carries separate routing and sequencing information so that it can be routed independently. In hierarchical-ring networks, we assume the size of a cell to be the same as a phit that can be forwarded in a single clock cycle from one node to a neighboring node. The first cell (header cell) of a packet, similar to a header flit in wormhole switching, carries the full target memory address while the remaining cells (body cells) of the packet identify only the destination PM. The cells of a packet are assembled together at a destination node. The destination node discards a packet if it does not receive all the cells of the packet.

The advantage of cell switching is that, unlike virtual cut-through switching, they do not require large packet-sized ring buffers at the nodes. In fact, there is no requirement for ring buffers (with the exception of transreceivers) at NICs and at IRIs when the cell size is the same as the phit size, since an incoming transit cell can always be transmitted on the outgoing link without having to be buffered (given that priority is always given to ring packets).

A major disadvantage of cell switching is the overhead (in the data path) associated with carrying routing and sequencing information in body cells. For example, in a 128 processor system, 7 bits are needed to address each processor, and 7 more bits to identify the source node in order to distinguish between cells from different source nodes to the same destination node. This amounts to a total of 14 bits which translates into about 11% overhead if we assume a 128-bit cell size.[1]

6.2 Buffer Management in Hierarchical-ring networks

6.2.1 Wormhole Switched Hierarchical Rings

In this section, we study the performance impact of IRI buffer size on hierarchical-ring networks under wormhole switching. We vary only the up/down buffer sizes and assume that the size of NIC and IRI ring buffers (referred simply as ring buffers) are optimal (as determined later) and fixed.[2] This allows us to study the sensitivity of the network performance to IRI up/down buffer sizes alone.

[1] There is no need to identify cells of different packets of the same source node since they arrive in order to the destination node due to the minimal, deterministic routing used in hierarchical rings.

[2] By optimal buffer size, we mean the minimum ring buffer size required for high throughput and low latency networks.

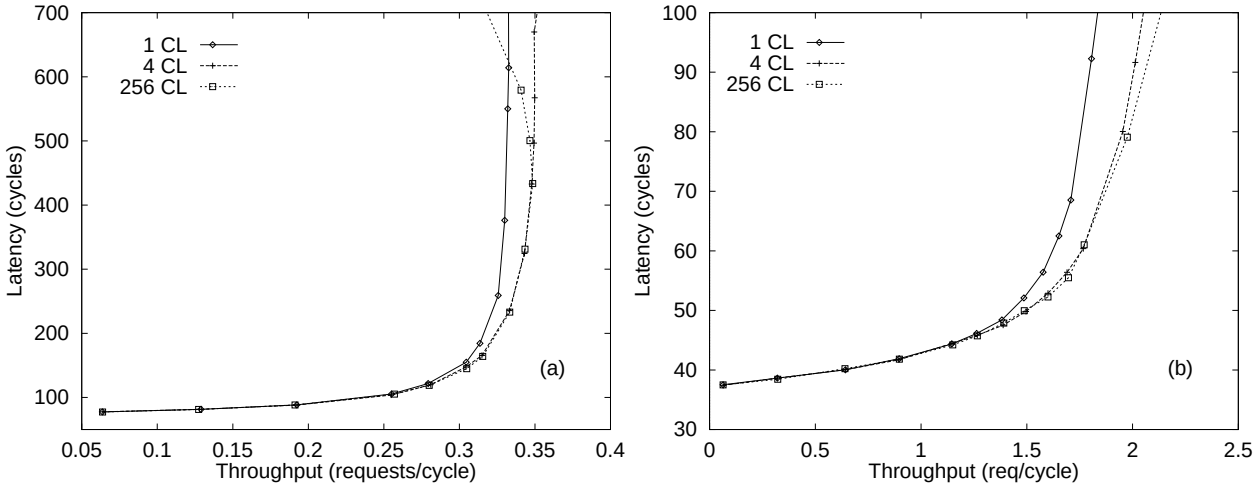

Figure 6.1: Throughput-latency curves for a 3-level, 64 processor $8 \times 4 \times 2$ hierarchical-ring system with 32-byte cache lines for the (a) $T_{uniform}$ and (b) T_{loc} workloads. The curves are shown for three different IRI buffer sizes.

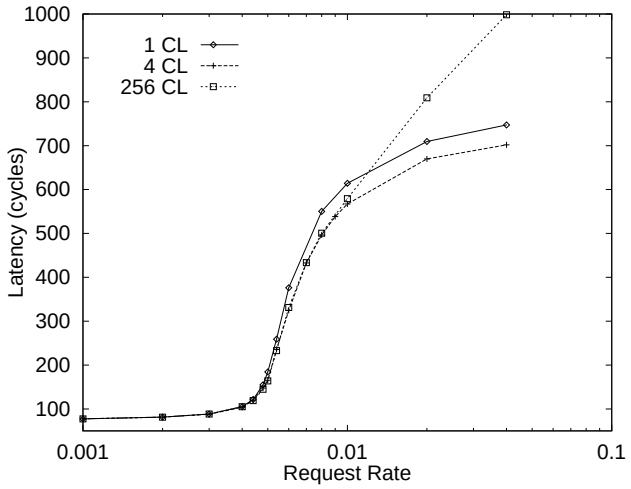

Figure 6.2: Latency as a function of request rate for a 3-level, 64 processor $8 \times 4 \times 2$ hierarchical-ring system with 32-byte cache lines for 3 different buffer sizes under $T_{uniform}$ workload.

Consider a 3-level, 64 processor $8 \times 4 \times 2$ hierarchical-ring system with 32-byte cache lines and a phit size of 128 bits. In such a system, a packet containing a cache line of 32 bytes (the read response and write request packet) is 3 flits long (1 flit for the header and 2 flits for the cache line). The other packets, namely the read request and write response packets, are 1-flit long. For the NIC and IRI, we chose a ring buffer size of 3 flits, which can hold the largest packet.

Figure 6.1 presents the throughput-latency curves for IRI buffer sizes of 3 flits (enough to hold a packet with a cache line), 12 flits (enough to hold 4 cache-line-sized packets), and 768 flits (to hold 256 cache-line-sized packets), which we shall henceforth refer to as 1 CL, 4 CL,

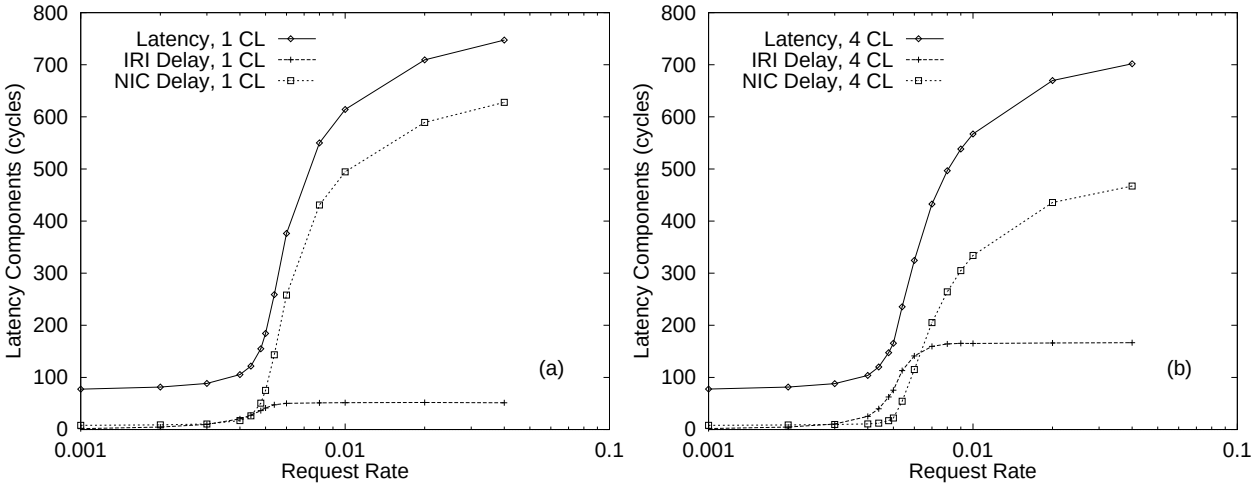

Figure 6.3: Latency components (NIC and IRI delays) as a function of request rate for a 3-level, 64 processor $8 \times 4 \times 2$ hierarchical-ring system with 32-byte cache lines and an IRI buffer size of (a) 1 CL and (b) 4 CL under $T_{uniform}$ workload.

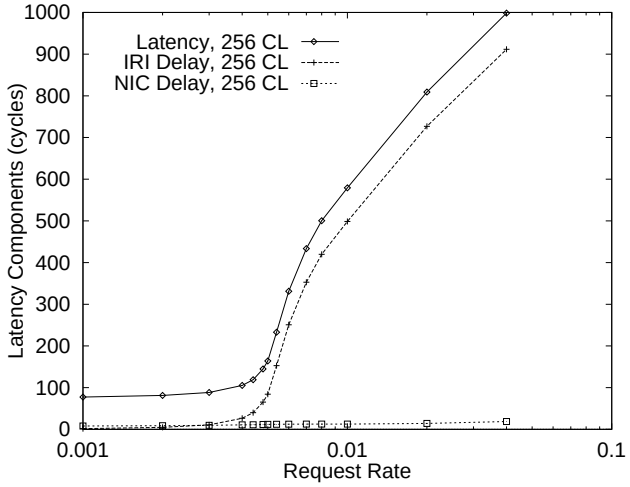

Figure 6.4: Latency components (NIC and IRI delays) as a function of request rate for a 3-level, 64 processor $8 \times 4 \times 2$ hierarchical-ring system with 32-byte cache lines and an IRI buffer size of 256 CL under $T_{uniform}$ workload.

and 256 CL-sized buffers. Figure 6.1a presents these curves for the $T_{uniform}$ workload, whereas Figure 6.1b presents the same for the T_{loc} workload. We observe that with wormhole switching, the maximum achievable throughput is good when the IRI buffer size is set to 1 CL. There is a small gain in throughput when the IRI buffer size is increased form 1 CL to 4 CL, with negligible gain in throughput thereafter with the T_{loc} workload, and a degradation in throughput with the $T_{uniform}$ workload. This is evident in Figure 6.1a, where we see a decrease in throughput at high request rates for the 256 CL IRI buffer sizes. The performance graphs for other system sizes and cache line sizes (64 and 128 bytes) are similar and are therefore not shown.

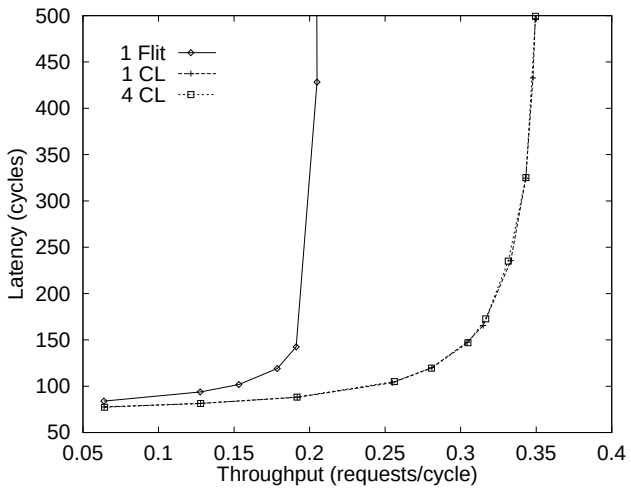

Figure 6.5: Throughput-latency curves for three different NIC ring buffer sizes under $T_{uniform}$ workload for a 3-level, 64 processor $8 \times 4 \times 2$ hierarchical-ring system with 32-byte cache lines.

Hence, with poor memory locality, large IRI buffers in wormhole switched hierarchical-ring networks hurt performance. To further analyze the behavior of large IRI buffer sizes under the $T_{uniform}$ workload, we present latency curves in Figure 6.2 for the three different IRI buffer sizes plotted as a function of request rate. We observe that under high request rates, the round-trip latency with 256 CL buffer sizes is much higher than with smaller buffer sizes. This is due to high IRI queuing delays that occur for the large buffer size under high request rates. The latency due to NIC and IRI queuing delays for different IRI buffer sizes is plotted in Figures 6.3 and 6.4. We see that for high request rates, NIC delays form a large fraction of the total latency for 1 CL buffer sizes. As we increase the IRI buffer size, the NIC delay decreases, but the IRI delay increases; the IRI delays form a larger fraction of the latency when a 256 CL IRI buffer size is used.

Next, we fix the IRI buffer size at 4 CL and proceed to analyze the performance impact of ring buffer sizes. Figure 6.5 plots the throughput-latency curves under the $T_{uniform}$ workload for three different ring buffer sizes, namely 1 flit, 1 CL, and 4 CL. There is a similar trend under the T_{loc} workload (not shown). We see a significant improvement in throughput when the ring buffer size is increased from 1 flit (traditional wormhole switching) to 1 CL (buffered wormhole switching). This is mainly because of the reduction in the number of links a blocked packet can hold. Additional increases in the ring buffer size do not further affect performance, because the single input/output link of a NIC effectively serializes the transmission of packets.

We conclude that the performance of wormhole switched hierarchical-ring networks is somewhat sensitive to the IRI buffer size, with a smaller size (of about 4 CL) resulting in high throughput. A higher IRI buffer size can hurt performance, especially under high request rates.

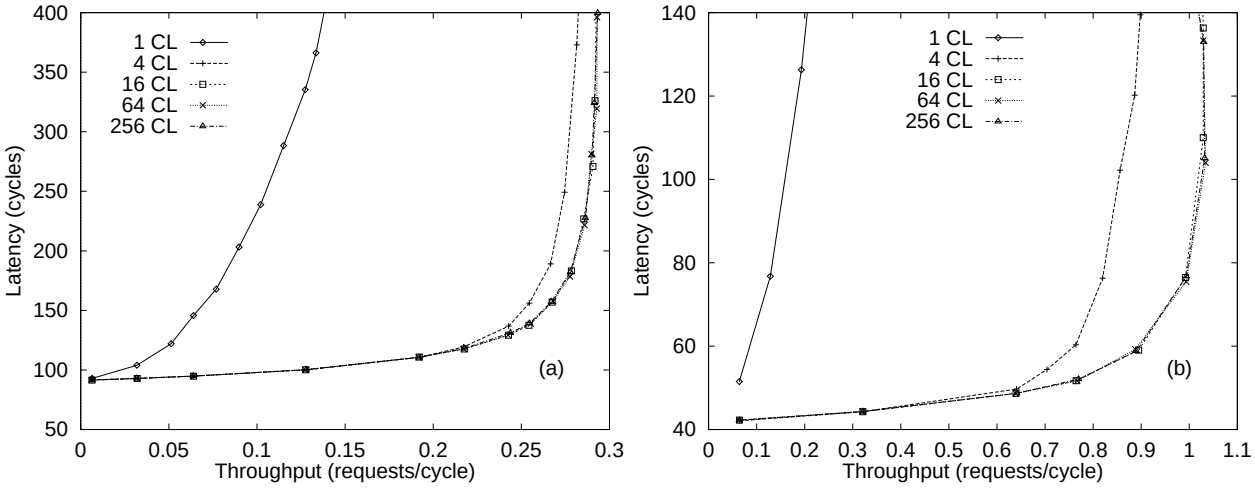

Figure 6.6: Throughput-latency curves for four different IRI buffer sizes under the (a) $T_{uniform}$ and (b) T_{loc} workloads for non-blocking virtual cut-through switched 3-level, 64 processor $8 \times 4 \times 2$ hierarchical-ring system with 32-byte cache lines.

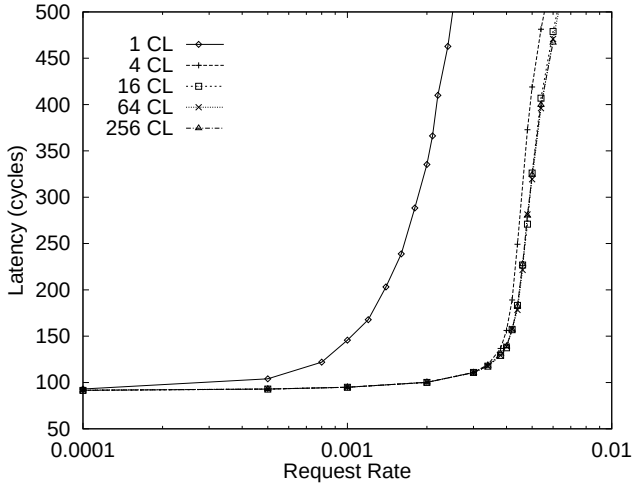

Figure 6.7: Latency as a function of request rate for four different IRI buffer sizes under the $T_{uniform}$ workload for non-blocking virtual cut-through switched 3-level, 64 processor $8 \times 4 \times 2$ hierarchical-ring system with 32-byte cache lines.

The network performance is not that much affected by ring buffer sizes greater than a single cache line size. There is a similar trend for larger cache line sizes (not shown). In a nutshell, buffered wormhole switching results in a significant performance improvement when compared to traditional wormhole switching with single flit buffers.

6.2.2 Virtual Cut-through Switched Hierarchical Rings

The performance of a virtual cut-through hierarchical-ring network with blocking flow control is similar to that of the buffered wormhole switched network considered in the previous section. In

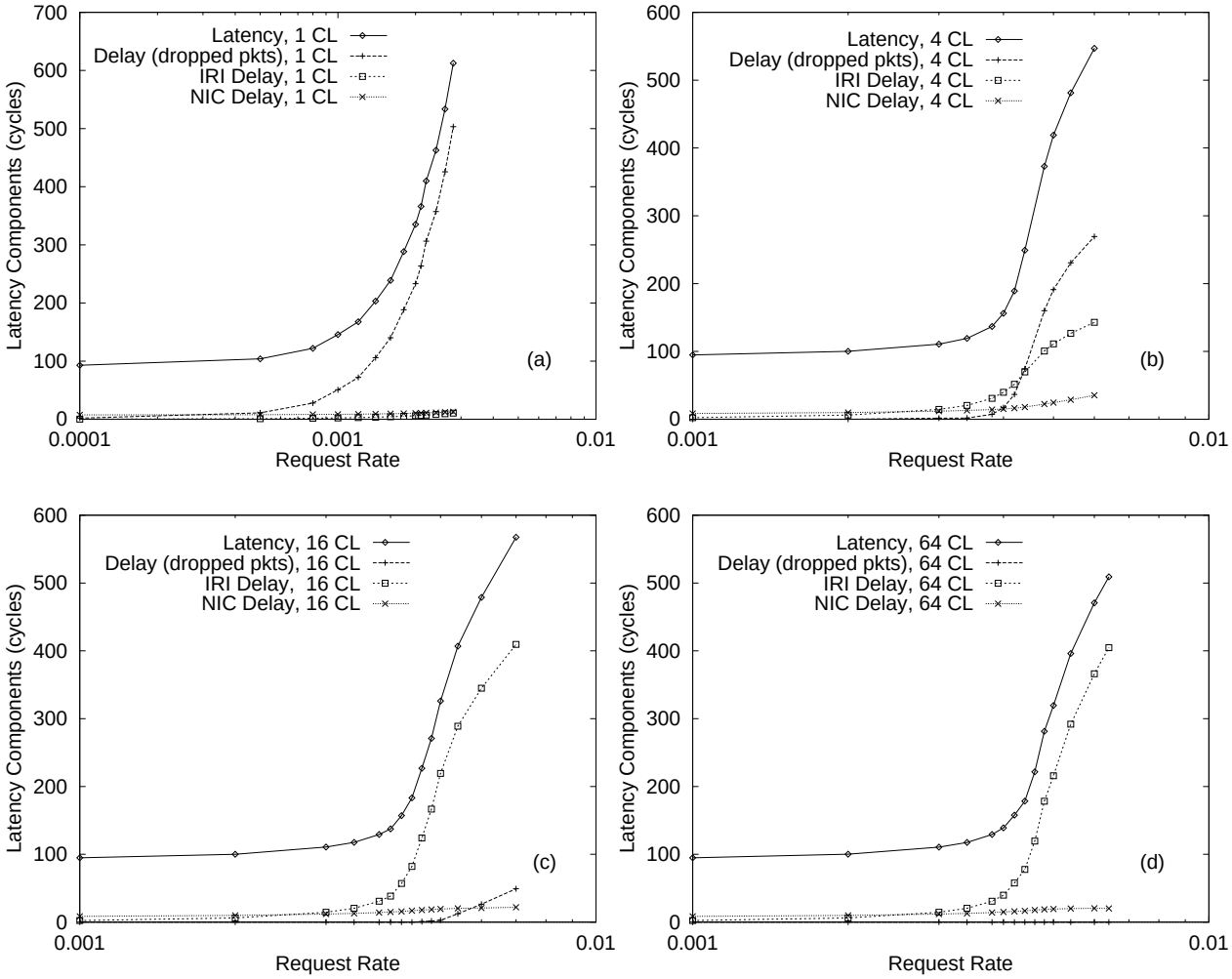

Figure 6.8: Latency components as a function request rate for a non-blocking virtual cut-through switched 3-level, 64 processor $8 \times 4 \times 2$ hierarchical-ring system with 32-byte cache lines for IRI buffer sizes of (a) 1 CL, (b) 4 CL, (c) 16 CL, and (d) 64 CL under $T_{uniform}$ workload.

wormhole switching, the optimal ring buffer size was one cache line deep, so a blocked packet can occupy at most only one link. For ring-based systems, therefore, there is no difference between when a blocked packet is buffered inside the NIC buffer (as in virtual cut-through switching) or buffered across two NIC buffers since all packet transmissions are serialized through a single output link. For this reason, we do not present performance graphs for blocking virtual cut-through switching in rings; instead, we focus on a non-blocking variant of virtual cut-through switching.

Figure 6.6 presents the throughput-latency curves for a 3-level, 64 processor $8 \times 4 \times 2$, non-blocking virtual cut-through switched hierarchical ring under $T_{uniform}$ and T_{loc} workloads. The time-out value was chosen to be large enough to avoid duplicate packets in the network.[3] We

[3] A time-out value of 5000 processor cycles was sufficient to prevent any duplicate packets in our simulations.

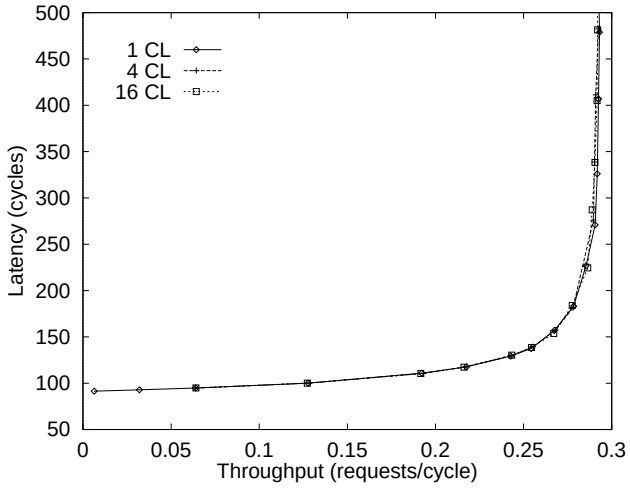

Figure 6.9: Throughput-latency curves for three different ring buffer sizes under $T_{uniform}$ workload with optimal IRI buffer size of 16 CL for a non-blocking VCT switched 3-level, 64 processor $8 \times 4 \times 2$, hierarchical-ring system with 32-byte cache lines.

vary the IRI buffer size, and keep the ring buffer sizes constant at 1 CL. The smallest IRI buffer size we consider is 1 CL, as this is the minimum required buffer size for virtual cut-through switching, and we go up to 256 CL. There is a significant increase in the maximum achievable throughput when we increase the buffer size from 1 CL to 4 CL, and there is a small further increase when we increase the buffer size to 16 CL but no increase beyond that. Figure 6.7 presents the corresponding latency curves under the $T_{uniform}$ workload. It can be seen that the IRI buffer size of 16 CL gives the lowest latency values for a wide range of request rates.

The performance sensitivity of non-blocking VCT networks to the IRI buffer size differs from that of blocking wormhole networks in the following ways: (1) in non-blocking VCT networks, we see an increase in the maximum achievable throughput as we increase the IRI buffer size, with a size of 16 CL being optimal, whereas in blocking wormhole ring networks a much smaller buffer size of 4 CL is found optimal, and (2) unlike in a wormhole network, a larger IRI buffer size in a non-blocking VCT network does not hurt performance.

Figure 6.8 presents the latency components as a function of the request rate for the four different IRI buffer sizes under the $T_{uniform}$ workload. In addition to the NIC delay and IRI delay, an important latency component in a non-blocking network is the fraction of round-trip time that is spent in time-outs and retransmissions when packets are dropped. This delay constitutes a major fraction of the latency for smaller IRI buffer sizes, since the IRIs drop a larger number of packets. However, this delay fraction decreases when the IRI buffer size is increased, eventually reducing to zero for 64 cache line IRI buffers. But the decreasing number of dropped packets is accompanied by an increase in IRI queuing delay.

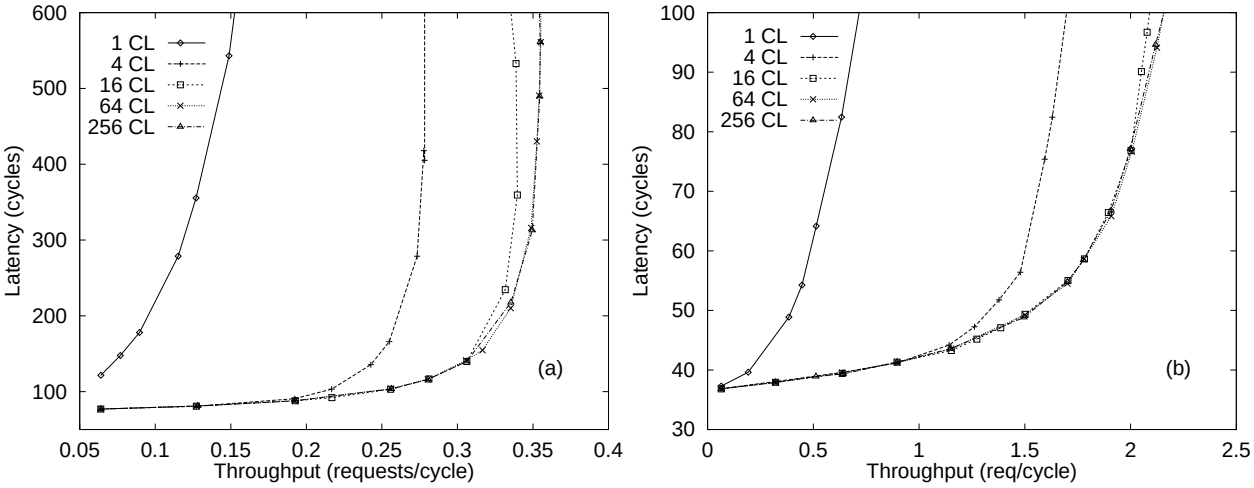

Figure 6.10: Throughput-latency curves for five different IRI buffer sizes under (a) $T_{uniform}$ and (b) T_{loc} workloads for a non-blocking cell switched 3-level, 64 processor $8\times4\times2$, hierarchical-ring system with 32-byte cache lines.

Figure 6.9 presents the throughput-latency curves for three different ring buffer sizes with the IRI buffer size kept at its optimal value of 16 CL. Similar to blocking wormhole networks, the performance is not affected by the size of the ring buffers. There is a similar trend under the T_{loc} workload (not shown). We also observe similar performance curves when we vary the IRI and NIC buffer sizes for a hierarchical-ring system with larger cache lines (64 and 128 bytes).

6.2.3 Cell Switched Hierarchical Rings

In this section, we consider non-blocking cell switched hierarchical-ring networks. Since we assume a cell size to be the same as the phit size, there is no requirement for ring buffers, as a cell arriving at a router can always be transmitted to the output link without being buffered.

Figure 6.10 depicts throughput-latency curves for a cell switched 3-level, 64 processor $8 \times 4 \times 2$ hierarchical-ring system. The curves are drawn for five different IRI buffer sizes. The behavior is similar to that of non-blocking virtual cut-through networks; however, the optimal IRI buffer size (64 CL) is larger than for the non-blocking virtual cut-through switched network. Figure 6.11 presents the corresponding latency curves as a function of request rate. It can be seen that the IRI buffer size of 64 CL gives the lowest latency values for a large range of request rates.

The latency components are shown in Figure 6.12 for four different IRI buffer sizes. Similar to virtual cut-through switching, the latency component due to time-outs and retransmission of dropped packets forms a large fraction of the latency for smaller buffer sizes and decreases with an increase in buffer size. However, as the IRI buffer size is increased, we see a rise in IRI

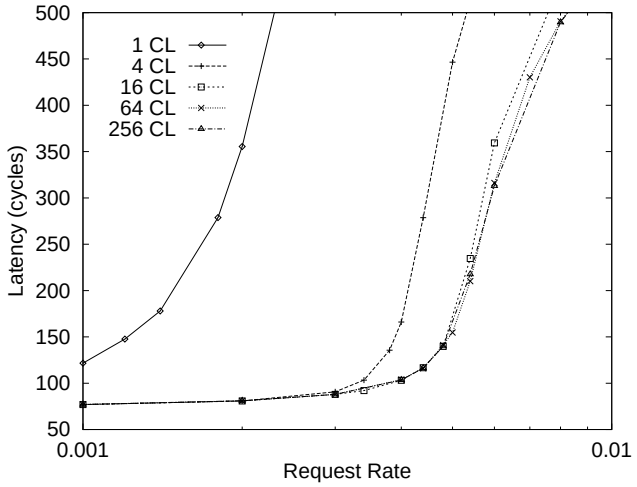

Figure 6.11: Latency as a function of request rate for four different IRI buffer sizes under $T_{uniform}$ workload for a non-blocking cell switched 3-level, 64 processor $8 \times 4 \times 2$, hierarchical-ring system with 32-byte cache lines.

queuing delay that forms a major fraction of the latency at larger buffer sizes.

6.3 Performance of Switching Techniques in Hierarchical-ring Networks

In this section, we compare the performance of various switching techniques, namely, (1) wormhole switching with 1 flit ring buffers, (2) buffered wormhole switching (with 1 CL sized ring buffers), (3) non-blocking virtual cut-through switching (with 1 CL sized ring buffers), and (4) cell switching. Table 6.1 compares the different characteristics and memory requirements of these switching schemes. The IRI buffer sizes in all these switching schemes are chosen to be optimal. Figure 6.13 presents the throughput-latency curves for the four different switching techniques in a 3-level, 64 processor $8 \times 4 \times 2$ hierarchical-ring system with 32-byte cache lines. It is obvious that traditional wormhole switching with 1-flit ring buffers results in poor performance when compared to the other switching techniques. This is because as the request rate increases, the probability of a packet being blocked increases, as does the likelihood of it blocking other packets, since a blocked packet spans multiple links. As a result, networks using wormhole switching with single flit buffers saturate from contention well before they exhaust their bandwidth. However, the advantages of single flit buffers include high-speed routers and complete isolation of nodes from in-transit packets [85]. Buffered wormhole switching (with single cache line ring buffers) results in a much improved performance, both in terms of average latency and throughput. This is because a blocked packet can hold at most one link. The disadvantage of buffered wormhole switching is that the ring buffer size becomes large for large

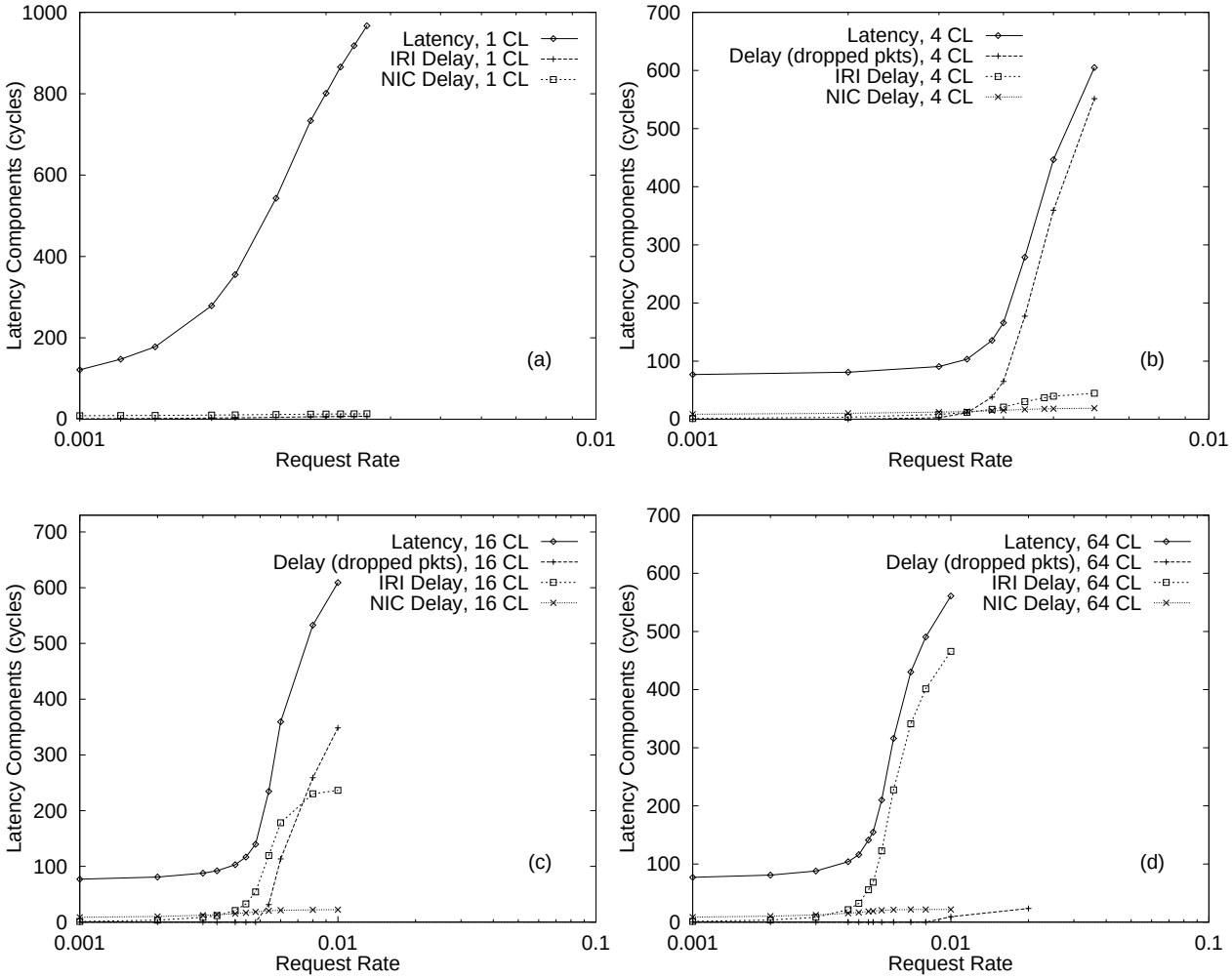

Figure 6.12: Latency components as a function of request rate for a non-blocking cell switched 3-level, 64 processor $8 \times 4 \times 2$, hierarchical-ring system with 32-byte cache lines for IRI buffer sizes of (a) 1 CL, (b) 4 CL, (c) 16 CL, and (d) 64 CL under $T_{uniform}$ workload.

cache lines, resulting in larger memory requirements.

Non-blocking virtual cut-through switching results in better performance than single flit wormhole switching, but performs poorly compared to buffered wormhole switching and suffers from the same disadvantages as buffered wormhole switching. Non-blocking cell switching results in a performance that is marginally better than buffered wormhole switching and as we will show, significantly better than blocking cell switching. However, we see that advantage only at high request rates. The other advantage of non-blocking cell switching includes small ring buffers. However, cell switching requires an overhead in the data path for carrying the source and destination node identification for each cell. This overhead can become a significant percent of the total data path for larger system sizes and/or for systems with smaller data path. On the other hand, all blocking switching techniques require virtual channels for deadlock free

	Blocking			Non-blocking	
	WH	Buffered WH	WH-Cell	VCT	Cell
Ring Buffer Size	1 Flit	1 CL	1 Flit	1 CL	0
IRI Buffer Size	4 CL	4 CL	4 CL	16 CL	64 CL
Virtual Channels	2	2	2	1	1

Table 6.1: Comparison of different switching and flow-control techniques in hierarchical-ring networks.

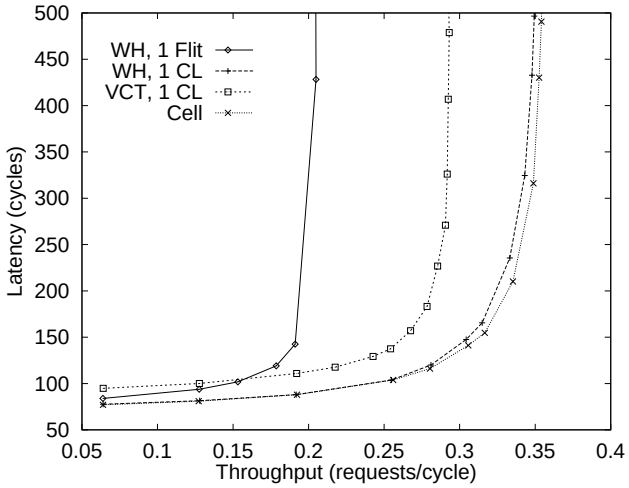

Figure 6.13: Throughput-latency curves for different switching techniques with blocking and non-blocking flow control under $T_{uniform}$ workload for a 3-level, 64 processor $8\times4\times2$ hierarchical-ring system with 32-byte cache lines.

routing. The performance of different switching techniques is similar in larger cache line sized (64 and 128 bytes) hierarchical-ring systems and are not shown.

6.4 Blocking Cell Switching in Hierarchical-ring Networks

An interesting alternative to single-flit wormhole switching is the blocking cell switching. As established above, the main disadvantage of single-flit wormhole switching is its poor throughput under high request rates due to the blocking nature of worms. On the other hand, non-blocking cell switching with the size of a cell being the same as the *phit*, cells do not block holding links. The blocking cell switching combines wormhole and cell switching where a worm is divided and sent as a sequence of cells with single-flit ring buffers. Each cell carries enough information to be routed independently. The size of each cell can vary, with the minimum being the size of the data path. The header cell normally carries the full target memory address, and the following cells carry only the source and destination node identification. The extra routing information

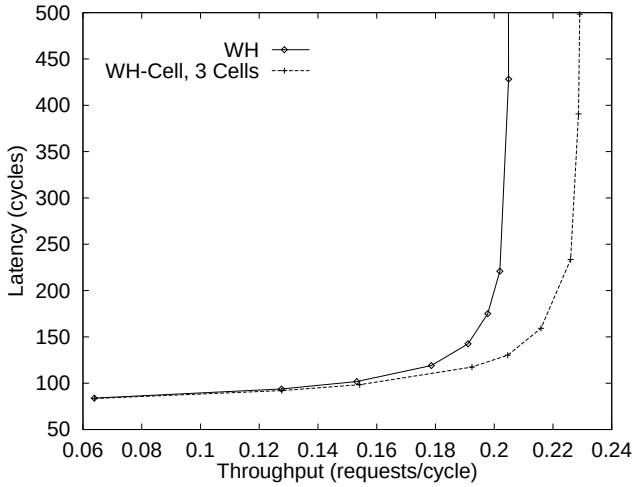

Figure 6.14: Throughput-latency curves for blocking cell switching with single flit buffers under $T_{uniform}$ workload for a 3-level, 64 processor $8 \times 4 \times 2$, hierarchical-ring system with 32-byte cache lines. A cache line is broken and sent as 3 cells instead of a single large worm. For comparison purposes, latency-throughput curve for wormhole switching with single flit buffers is also shown.

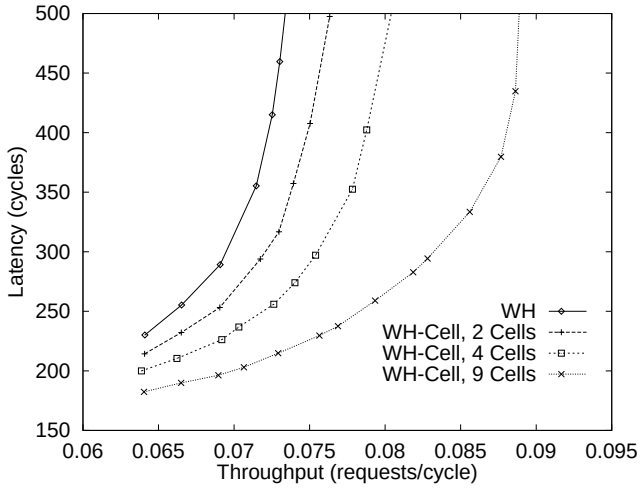

Figure 6.15: Throughput-latency curves for blocking cell switching under $T_{uniform}$ workload for a 3-level, 64 processor $8 \times 4 \times 2$, hierarchical-ring system with 128-byte cache lines. A cache line is sent as 2, 3, and 9 cells instead of a single large worm. For comparison purposes, throughput-latency curve for single-flit wormhole switching is also shown.

requires extra wires, similar to cell switching. However, since it is blocking, it eliminates the need for time-outs. It differs from wormhole switching in that a blocked cell typically occupies fewer links (depending on the size of a cell). When the cell size is the same as the phit size, a blocked cell may hold no links.

Figure 6.14 presents the throughput-latency curves for blocking cell switching with single-flit ring buffers for a 3-level, 64 processor $8 \times 4 \times 2$, hierarchical-ring system with 32-byte cache lines.

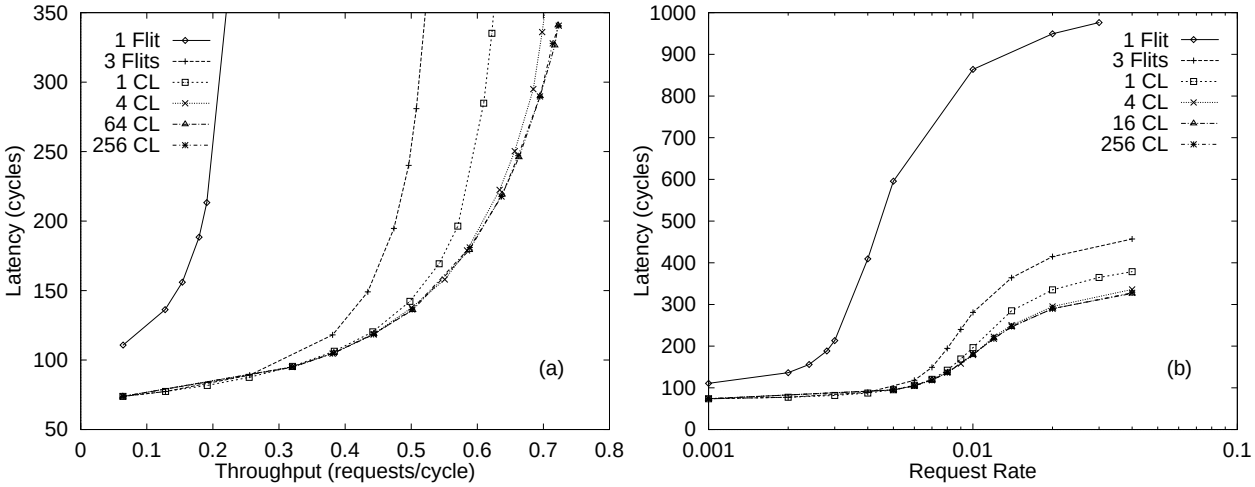

Figure 6.16: Performance impact of NIC buffer sizes on a 64 processor 8×8, 2-dimensional mesh connected system with 32-byte cache lines under the $T_{uniform}$ workload. (a) Throughput-latency and (b) latency versus request rate curves are shown.

Assuming a cell size same as the phit size (128 bits), blocking cell switching with 3 cells for a cache line transfer achieves a higher throughput than sending a cache line as a single worm. Figure 6.15 presents the throughput-latency curves for the same system, but with 128-byte cache lines. This time we vary the number of cells per packet from 2 to 9.

It should be noted that blocking cell switching should be considered only as an alternative to single-flit wormhole switching as it still performs poorly when compared to either buffered wormhole or non-blocking cell switching.

6.5 Buffer Management in Direct Networks

6.5.1 Wormhole Switched Mesh Networks

In this section, we present the performance impact of NIC input buffer sizes on 2-dimensional wormhole switched mesh networks. We vary the NIC buffer size from 1 flit (traditional wormhole) to 256 CL. We choose a phit size of 32 bits and a cache line size of 32 bytes. As a result, the largest packet is 12 flits long (containing a cache line and a 4-flit header), while the smallest packet is 4 flits long.

Figure 6.16a presents the throughput-latency graphs for a 64 processor 8×8, mesh-connected network under the $T_{uniform}$ workload. In wormhole switching with single-flit buffers, the network saturates far before the full network bandwidth is exhausted. An increase in the NIC buffer size from a 1 flit to 3 flits results in a much improved performance with the maximum achievable throughput increasing by more than 100%. This is because the maximum number of links a

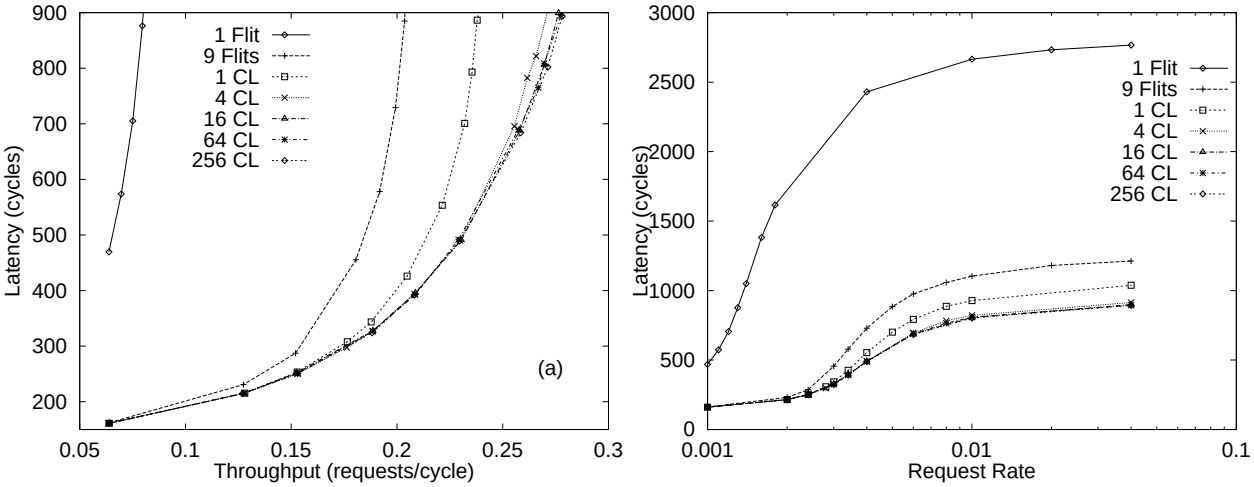

Figure 6.17: Performance impact of NIC buffer sizes on a 64 processor 8×8, 2-dimensional mesh connected system with 128-byte cache lines under the $T_{uniform}$ workload. (a) Throughput-latency and (b) latency versus request rate curves are shown.

blocked packet can occupy is greatly reduced; with 3-flit NIC buffers a packet can occupy at most 3 links. An one CL NIC buffer size (12 flits) results in an another 20% improvement in the maximum achievable throughput. However, further increases in NIC buffer size result in diminishing returns, with only about a 15% increase in the maximum achievable throughput for a 256 CL NIC buffer size. A 3-flit NIC buffer size (one-fourth of a CL) is therefore a good compromise considering buffer space requirements. The latency curves presented in Figure 6.16b further strengthen our case for 3-flit NIC buffers.

We observe a similar trend for larger cache lines. Figure 6.17 depicts the throughput-latency and latency curves for systems with 128-byte cache lines, where a packet can be as large as 36 flits. In this case, we consider buffer sizes of 1 flit, 9 flits, and 1 CL (36 flits) to 256 CL. We can see from the performance curves that 9-flit buffers (one-fourth of a CL) are a good compromise considering buffer space requirements.

For the T_{loc} workload, the trend is very similar except that the increase in the maximum achievable throughput with the increase in NIC buffer size is relatively small (not shown).

The total buffer space requirements for mesh routers with different NIC buffer sizes are presented in Table 6.2. We conclude that while a single flit NIC buffer is a poor choice in terms of performance, a multiple cache line size buffer results in large buffer space requirements. A NIC buffer size that lies between a single flit and a single cache line is a good compromise between performance and buffer space requirement. We choose flit buffers one-fourth the size of a cache line for the purpose of requiring the same amount of memory per router as hierarchical-ring NICs with 1 CL ring buffers (see Table 4.1).

	VCs per Physical Channel	Cache line size	NIC memory requirements					
			1-flit	1/4 CL	1/2 CL	1 CL	4 CL	16 CL
Hierarchical Rings	2	32B	32B	-	-	96B	384B	1.5KB
		64B	32B	-	-	160B	640B	2.5KB
		128B	32B	-	-	288B	1.125KB	4.5KB
Bidirectional Rings	2	32B	32B	-	96B	192B	768B	3KB
		64B	32B	-	160KB	320KB	1.25KB	5KB
		128B	32B	-	288B	576B	2.25KB	9KB
Meshes & Tori	2	32B	32B	96B	-	384B	1.5KB	6KB
		64B	32B	160B	-	640B	2.5KB	10KB
		128B	32B	288B	-	1.125KB	4.5KB	18KB
Meshes	1	32B	16B	48B	-	192B	768B	3KB
		64B	16B	80B	-	320B	1.25KB	5KB
		128B	16B	144B	-	576B	2.25KB	9KB

Table 6.2: A comparison of NIC buffer memory requirements.

6.5.2 Wormhole Switched Tori Networks

Torus-connected systems exhibit similar characteristics as mesh-connected systems with respect to NIC buffer sizes. Figure 6.18 presents the throughput-latency and latency versus request rate curves for a 64 processor 8×8, torus system with 32-byte cache lines for different NIC buffer sizes. The curves are similar to their mesh counterparts, with a 3-flit NIC buffer size (one-fourth of a CL) being a good compromise between performance and buffer space requirements. However, the maximum achievable throughput in the torus is about 40% higher than in a comparable mesh (for 3-flit buffers). The improved throughput is partly due to the reduced network diameter and partly due to the existence of two virtual channels per physical channel. Even though the virtual channels are used for deadlock free routing, they result in reduced contention for output links in routers, thus resulting in better link utilization.

6.5.3 Wormhole Switched Bidirectional Rings

We consider wormhole switched bidirectional rings with two virtual channels per physical channel. Under the assumption of constant pin constraints, we choose a phit size of 64 bits for bidirectional rings, which is twice the size of 32 bits considered for mesh/torus networks.[4] The throughput-latency curves for a 64 processor bidirectional ring show a trend similar to the other systems for different NIC buffer sizes as illustrated in Figure 6.19. We see a 3-flit buffer size

[4]This follows from the fact that a bidirectional network has 2 input and 2 output links, as opposed to 4 each in a torus/mesh network.

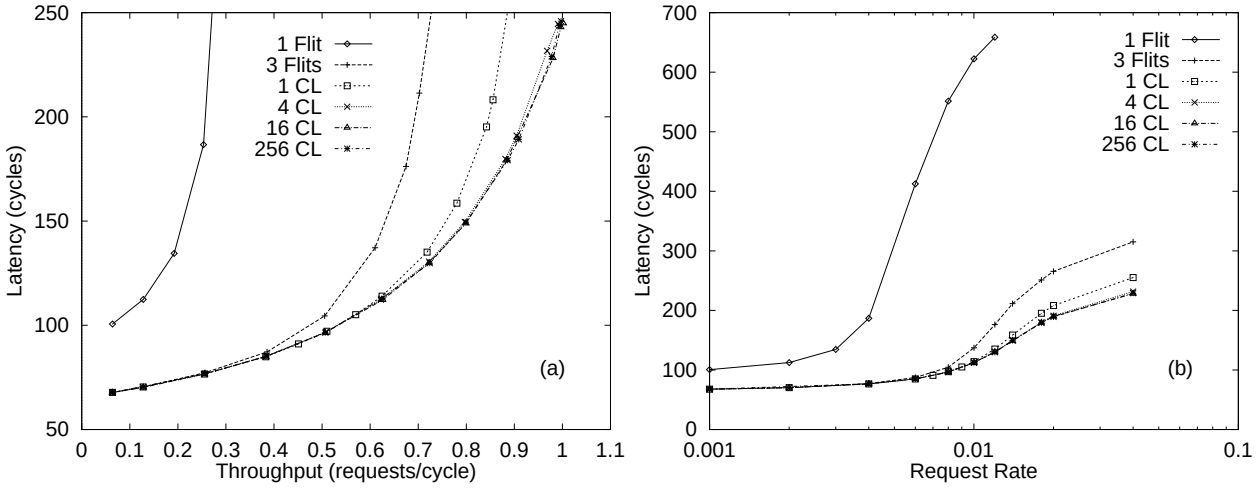

Figure 6.18: Performance impact of NIC buffer sizes on a 64 processor 8×8, 2-dimensional torus system with 32-byte cache lines under $T_{uniform}$ workload. (a) Throughput-latency and (b) latency versus request rate curves are shown.

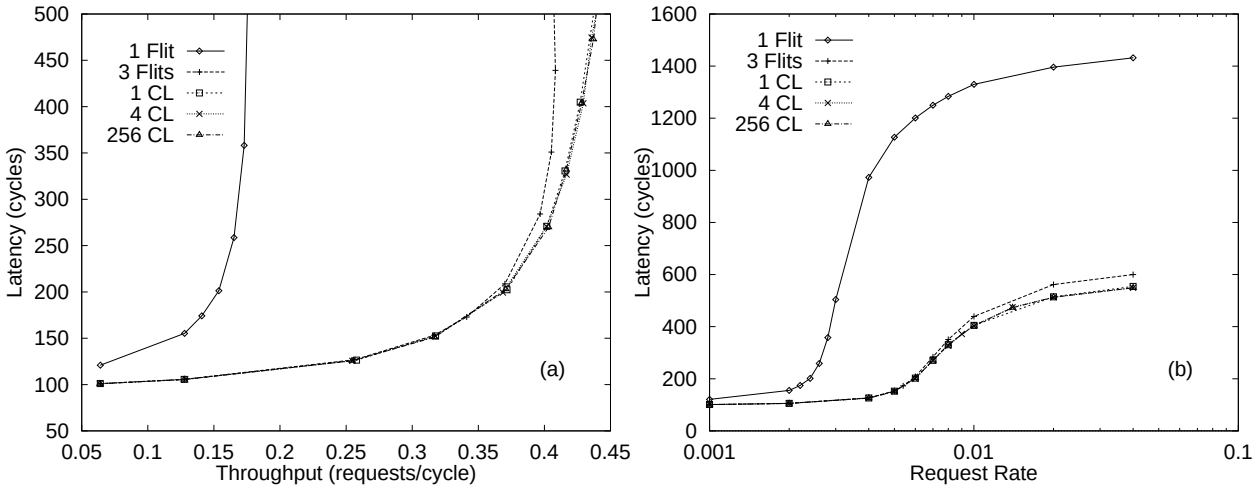

Figure 6.19: Performance impact of NIC buffer sizes on a 64 processor bidirectional ring system with 32-byte cache lines under $T_{uniform}$ workload. (a) Throughput-latency and (b) latency versus request rate curves are shown.

(one-half of a CL) as a good trade-off between performance and buffer space requirement.

Chapter Summary

In this chapter, we presented the results of a performance analysis of different cut-through switching techniques and the impact of buffer size on hierarchical-ring, mesh, torus, and bidirectional-ring connected systems. These results have been derived under the assumption that there is no distinction between a flit and a phit. We summarize the main conclusions from this chapter as follows:

- In hierarchical-ring networks, buffered wormhole switching and non-blocking cell switching perform better than other switching techniques considered. While the buffered wormhole switching requires smaller IRI buffers when compared to cell switching, it requires cache line sized ring buffers and two virtual channels per physical channel for deadlock free routing.

- In direct networks, buffered wormhole switching is preferred due to its ability to improve system throughput significantly without large NIC buffer space requirements.

- A blocking cell switching is an attractive alternative to the blocking wormhole switching with single-flit buffers.

Chapter 7
Routing

Deadlock free routing techniques in 2-dimensional direct networks have been studied thoroughly [18, 24, 25, 26] and therefore we only briefly present them here. For 2-dimensional meshes, dimension-ordered routing is a minimal and deterministic routing algorithm. It routes a packet along the lowest dimension first for as far it must go, before routing it on the next higher dimension. Dimension-ordered routing guarantees deadlock freedom in 2-dimensional meshes by enforcing a strict monotonic order on the dimensions traversed. For 2-dimensional tori and bidirectional rings, dimension-ordered routing is still minimal and deterministic, but not deadlock free. In this case, a deadlock would involve wraparound channels within a given dimension. Such deadlock cycles in a single dimension can be broken by splitting each physical channel along a cycle into two virtual channels and restricting the assignment of packets to these channels during routing (see Chapter 2).

In this chapter, we focus on deadlock free deterministic routing in hierarchical-ring connected networks. A wormhole switched hierarchical-ring network is susceptible to deadlock as any wormhole switched direct networks. Because multiple rings are connected by a hierarchy, there are several ways deadlock can occur. The deadlock could involve single rings and/or multiple rings. We present in this chapter a novel routing technique that prevents deadlock in wormhole switched hierarchical-ring networks.

7.1 Deterministic Routing in Hierarchical-ring Networks

Figure 7.1 presents the two possible ways deadlock can occur in a two-level hierarchical-ring network. The deadlock can occur in individual rings (including the global ring). These cycles are labeled from 1 to 4 in the figure. The other way deadlock can arise is from cyclic dependencies of channel resources that span the hierarchy. One such deadlock cycle (label 5) is shown in the figure. Preventing such multiple deadlock cycles in a hierarchy of rings makes the design of a deadlock free routing algorithm non-trivial. Figure 7.3 presents the channel dependency graph

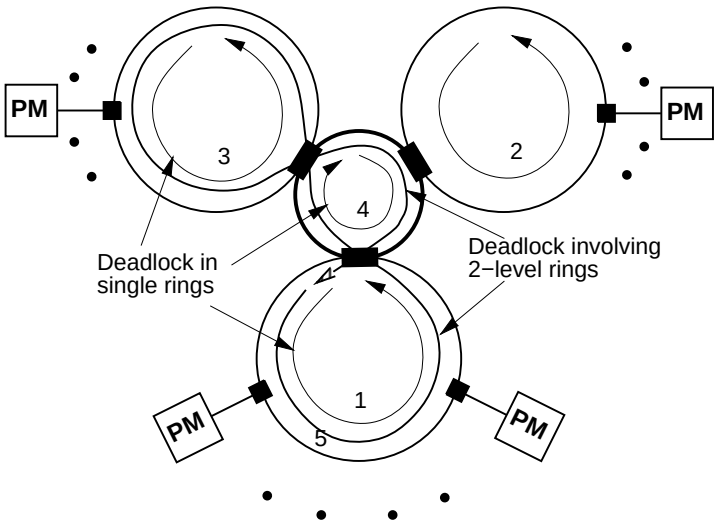

Figure 7.1: Deadlock cycles in a 2-level wormhole switched hierarchical-ring network.

for a 2-level hierarchical-ring network that clearly shows the different deadlock cycles.

Dally and Seitz proposed a necessary and sufficient condition for minimal deterministic routing in wormhole switched networks to be deadlock-free [18]. They state that a deterministic routing function is deadlock free *if and only if* there are no cycles in the channel dependency graph (see Chapter 2). For a single unidirectional ring, we can break such cycles by splitting each physical channel along a cycle into a group of virtual channels. We number the nodes and corresponding output channels of a single ring as $n_0, n_1, ..., n_m$ and $c_0, c_1, ..., c_m$, respectively, and pick channel c_0 as the dividing channel. We then split each channel into high virtual channels, $c_{10}, c_{11}, .., c_{1m}$, and low virtual channels, $c_{00}, c_{01}, .., c_{0m}$. If the routing algorithm is such that the packets at a node numbered less than their destination node are routed on the high channels, and packets at a node numbered greater than their destination node are routed on the low channels, then such an algorithm is proven to be deadlock-free.

To extend the deadlock-free routing algorithm from a single ring to a hierarchy of rings, we again split each physical channel into low and high virtual channels. We number the network nodes (NICs) as $nic_0, nic_1, ..., nic_m$, where m is the total number of nodes, starting from local ring 0. The IRIs are numbered separately as $iri_0, iri_1, ..., iri_n$ starting from the IRI associated with local ring 0. Because the deadlock cycles can involve both single and multiple rings, we pick a dividing channel in each local and each higher-level ring. This is shown in Figure 7.2 for a 2-level hierarchy. The dividing channel in a local ring is used to break deadlock cycles arising from the traffic confined to the local ring, while the dividing channel at the higher level ring is used to break cycles that span the hierarchy.

For packets whose source and destination nodes lie in a same local ring, we use the local

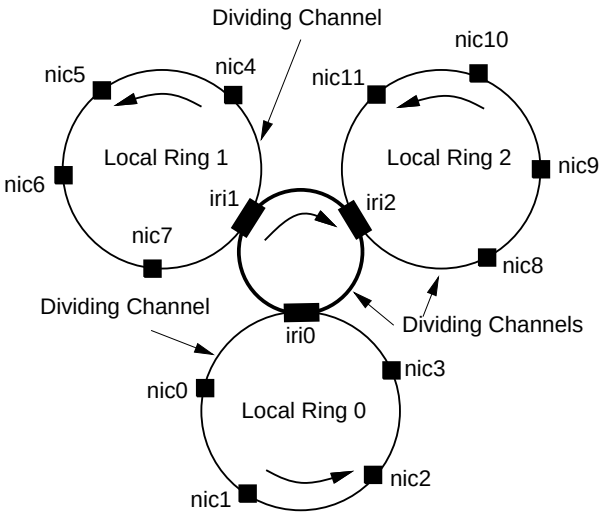

Figure 7.2: Dividing channels and node numbers for deadlock free routing in a 2-level wormhole switched hierarchical-ring network.

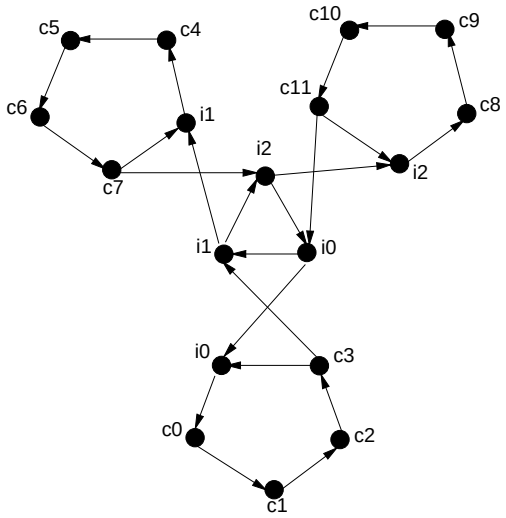

Figure 7.3: Channel dependency graph for a 2-level wormhole switched hierarchical-ring network with deterministic, minimal routing. The graph shows various deadlock cycles in such a routing algorithm.

dividing channel to break deadlock cycles in exactly the same way as in a single ring. For packets, whose source and destination nodes lie in different rings, the dividing channel of the highest level ring the packet must travel to is used to break the deadlock cycle. A packet arriving at a NIC in the same ring as the source node (source ring) of the packet is routed on a high virtual channel if the destination node number is higher than the source node number; otherwise it is routed on the low virtual channel. A packet arriving at an IRI, is routed on the high virtual channel if the destination IRI (the IRI that connects to the destination

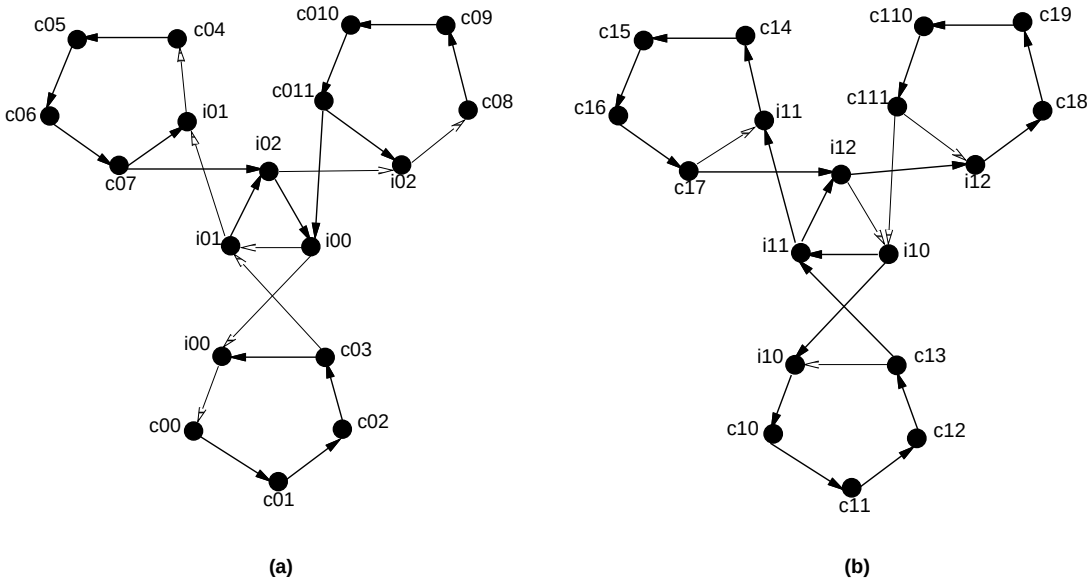

Figure 7.4: Channel dependency graphs for a 2-level wormhole switched hierarchical-ring network with deadlock-free routing using virtual channels. Channel dependency graphs are presented for (a) low and (b) high virtual channels.

ring of the packets) is higher than the source IRI; otherwise it is routed on the low virtual channel. Figure 7.4 shows the channel dependency graph of the routing algorithm for low and high virtual channels. The channels with lighter shades are restricted by our routing algorithm. Since neither of the graphs has cycles, we conclude that the routing algorithm prevents deadlock in wormhole switched hierarchical-ring networks.

Chapter Summary

A wormhole switched hierarchical-ring network is susceptible to deadlock. We presented in this chapter a deadlock free routing technique for wormhole switched hierarchical-ring network that extends Dally and Seitz's deadlock free routing in single rings.

Chapter 8
Prioritized Direct Networks: Design and Performance

Adding priorities to direct interconnection networks (of shared-memory multiprocessor systems) can lead to a number of advantages. It can reduce average latencies and improve system throughput. It can be used to support multiple classes of traffic, such as regular, best-effort traffic and multimedia, time-constrained traffic. It can lead to much lower variance in latency and hence improved system predictability, which is important for (soft) real-time systems.

Adding priorities to direct networks is surprisingly simple, and involves three main components: (*i*) priority-based link arbitration, (*ii*) priority inheritance, and (*iii*) dynamic virtual channels. With priority-based link arbitration, if two or more packets compete for the same idle link, the link will be assigned to the higher priority packet (as opposed to assigning in a round-robin or in a FIFO manner). There are many ways to assign priority to packets. For instance, we can assign priority to a packet based on its age, transaction type, or size.

This simple priority scheme can, however, result in priority inversion, where a lower priority packet may block a higher priority packet that may come behind it in a queue. With priority inheritance [70], a blocking lower priority packet at the head of a queue temporarily inherits the priority of the higher-priority packet behind it. This allows the lower priority packet to obtain the desired link sooner, thereby reducing the queuing delays for higher priority packets. Though priority inheritance was originally introduced to prevent priority inversions of real-time tasks in operating systems, we are unaware of any work that has applied priority inheritance to multiprocessor networks.

Finally, we introduce the concept of *dynamic virtual channels* that allows the allocation of virtual channels dynamically [75]. We allocate new virtual channel buffers for high priority packets that would otherwise unnecessarily be blocked.

In this chapter, we show how a connectionless wormhole switched two-dimensional mesh connected shared-memory multiprocessor network can be extended to support priorities of network packets, and we analyze its performance. Through flit-level simulations, we show that such prioritized networks can significantly reduce latency and improve system throughput, how

they can support multiple classes of traffic, and how they can improve system predictability.

Although prioritized networks have been studied before [52, 78] they have only been considered for implementing real-time multiprocessor networks and not for improving system throughput. Our work is also different from this work in that we do not establish virtual connections between end-to-end nodes but use a connectionless network that do not require a connection set-up to reserve link bandwidth and buffer space for routing time-constrained traffic. In other related earlier work [77], Rexford et. al. propose virtual networks for routing different classes of traffic. Our approach is different in that we use demand driven dynamic virtual channels as opposed to static virtual channels and employ priority inheritance. By dynamically allocating virtual channel buffers from a common pool, we utilize buffer resources more efficiently (when compared to utilization of static virtual channel buffers) and by allowing the number of virtual channels in a physical channel to vary (depending on demand), we improve system throughput and reduce average transaction latencies when compared to the case with static virtual channels.

8.1 The Problem

In this section, we illustrate one of the applications of prioritized networks, namely to support two classes of traffic. Multiprocessor systems are increasingly being used for multimedia applications, while still serving as data and computation engines. In the backplane networks of such systems, a variety of traffic types will co-exist, ranging from traffic of parallel computations, which we refer to as *best-effort traffic*, to the traffic for multimedia audio and video communications, which we refer to as *time-constrained traffic*. These two types of traffic have quite different traffic characteristics and performance requirements. Time-constrained traffic often requires a bound on worst-case latency, while a good average-case behavior will suffice for the best-effort traffic arising from parallel computations. Bounds on worst-case latency could be provided if the network is connection oriented and resources can be reserved in advance during a connection set-up phase. In addition to the overhead of setting-up a connection, each reservation decreases the available link bandwidth and buffer resources for regular traffic. Since reservation based schemes are conservative, they reserve more network resources than required, often far more than needed on average.

A connectionless network, in contrast, allows better utilization of network resources among several classes of traffic. Although a connectionless network may not be able to guarantee bounds on worst-case latency, we will show it can be effective in significantly reducing worst-

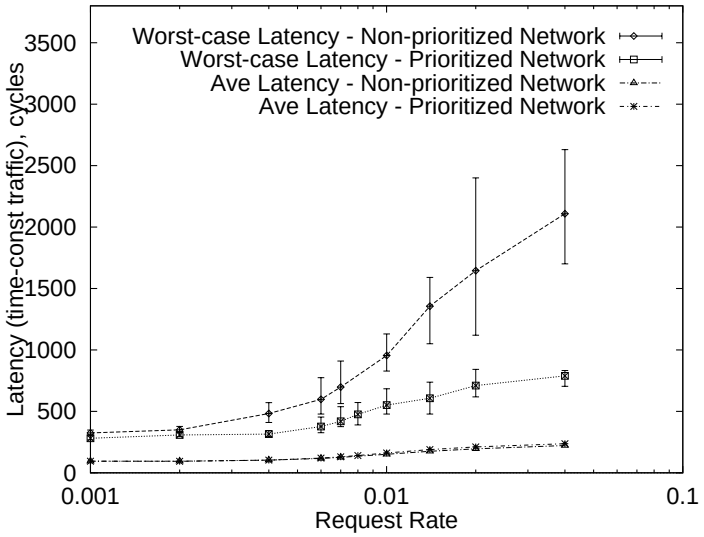

Figure 8.1: Worst-case and average communication latencies for time-constrained traffic in a 2D 8×8 mesh-connected multiprocessor network. Worst-case latency is shown both for round-robin link arbitration and with dynamic virtual channels. The errorbars show the variances on these values.

case latency bounds and therefore can be used in conjunction with jitter[1] control techniques at the end nodes [48]. As such, priority based interconnection networks are suitable for at least a subclass of multimedia applications, and they certainly can be used to improve the behavior of interactive applications [11].

Wormhole routed networks with round-robin link arbitration are used in many of todays multiprocessor routers, and they deliver good average performance. However, worst-case communication latency can be very high and unpredictable as the network load increases. Figure 8.1 illustrates this. Assuming a workload described in a later section (containing time-constrained and best-effort traffic), the bottom curve plots the average communication latency of time-constrained requests as a function of load rate of best-effort requests for a 2-dimensional 8×8 mesh network. The top curve plots the worst-case latency of the time-constrained traffic for the same workload. We used batch-mean analysis method [59] where the average latency is computed as the grand average of all batch averages and the worst-case latency is computed as the average of all batch worst-case latencies. The errorbars show the variances on these values. For the worst-case latency, the top end of the errorbars represents the global worst-case (over all batches), while the bottom end represents the global best (over all batches) of the worst-cases.

It is apparent that the worst-case latencies and their variance increase significantly as the load increases. The curve in the middle plots the worst-case latencies of time-constrained traffic

[1]Unpredictability in the worst-case latency of time-constrained traffic leads to *delay jitter* defined as the variance in latency encountered during individual transactions.

for the same workload, but for a network that uses the techniques proposed in this Chapter. It is clear from this curve that the techniques are effective in reducing worst-case latency and its variance without the need for bandwidth reservation. While our goal is to reduce the worst-case latency of time-constrained packets, we wish to do so without unnecessarily penalizing best-effort traffic. By routing time-constrained traffic mainly through dynamically assigned channels, we reserve a set of primary virtual channels for best-effort traffic. This prevents performance deterioration of best-effort traffic even when there is a moderately high level of real-time traffic.

8.2 Static Virtual Channels

A network with virtual channels organizes the flit buffers associated with each physical channel into several virtual channels. Virtual channels increase physical channel utilization, and thus network throughput, because any blocked packet that spans several nodes occupies only one virtual channel, and can be bypassed using any of the other virtual channels associated with a physical channel. The virtual channel assignment is made at the packet level, while the physical channel is allocated at the flit level. The virtual channels associated with a physical channel arbitrate for physical channel bandwidth on a flit-by-flit basis. With *static virtual channels*, the number of virtual channels per physical channel remains constant, whereas with *dynamic virtual channels* (which we describe in the next section), they vary over time. Figure 8.2a shows a mesh NIC with two virtual channels per physical channel; the number of virtual channels in this case is static and will not vary. Figure 8.2b shows a NIC with dynamic virtual channels; the number of virtual channels per physical channel will vary over time, with the minimum number per physical channel being 1.

At the receiving side of a node, the routing algorithm first assigns an incoming packet to an output physical channel and then to a virtual channel. When virtual channels are used for deadlock free routing, then the choice of virtual channel is dictated by the routing algorithm; Otherwise, another allocation scheme is used or any free virtual channel associated with the physical channel is chosen.

Hardware support for static virtual channels require status registers at the transmitting and the receiving side of a node [19, 22]. The transmitting node contains a status register for each virtual channel on the corresponding receiving node. The status register normally includes a bit to indicate whether the virtual channel is active or idle and a count of the number of free virtual channel buffers. The active/idle bit is used to prevent interleaving of the flits of different packets. The receiving node contains a status register for each virtual channel that contains

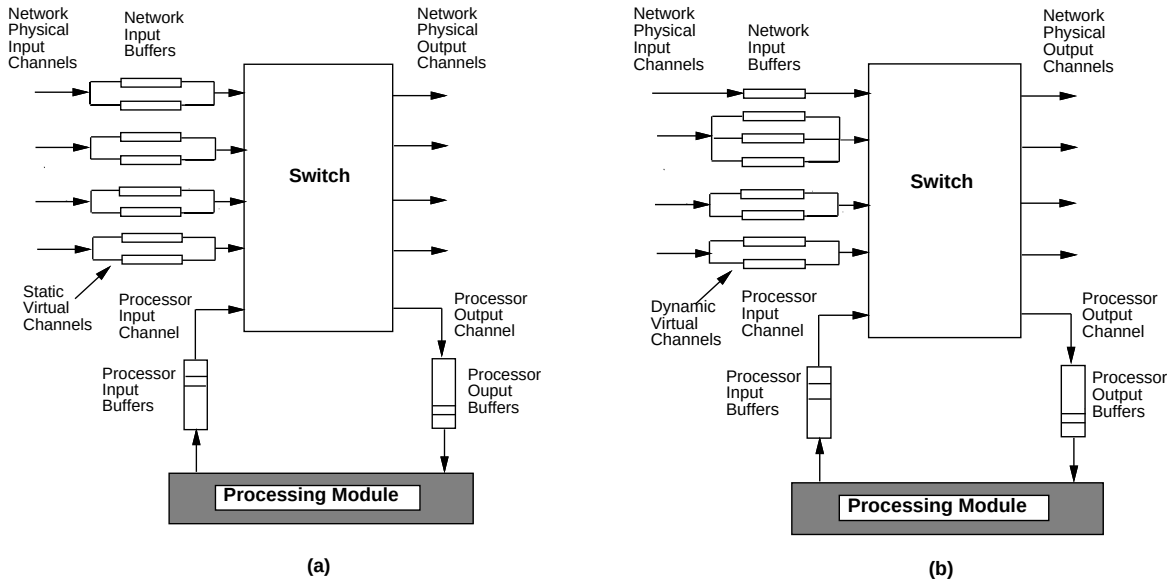

Figure 8.2: Mesh Network Interface Controller with (a) static and (b) dynamic virtual channels.

information such as the state of the channel and optionally, input and output virtual channel pointers. The status register storage requirements per physical channel is given by the following equation,

$$S_{pc} = N(\log(B_{vc}) + 1) + N \tag{8.1}$$

where the first and second term represent the storage requirement at the transmitting and receiving side of a node, respectively. B_{vc} is the number of flit buffers per virtual channel, and N is the number of virtual channels per physical channel. For $N = 4$, and $B_{vc} = 4$ flits, the status buffer storage requirement is 16 bits.

Adding virtual channels requires a few additional wires in the physical channel to identify the virtual channel for each transmitted packet in the forward direction and to indicate the availability of buffers to the transmitting node in the reverse direction. The virtual channel buffer counter at the transmitting side is incremented each time a flit is transmitted to the neighboring node and decremented when the neighboring node signals that it has forwarded a flit and thus freed up buffer storage by back propagating a *freed* signal along with the virtual channel identification. The extra channel width overhead for supporting virtual channels in a network with 32-bit phits with 4 static virtual channels per physical channel is: 2 bits to transmit the virtual channel id in the forward path, 2 bits to transmit the virtual channel id in the reverse path, and a *freed* line.

8.3 Dynamic Virtual Channels

In this section, we propose dynamic virtual channels. Dynamic virtual channels are similar to static virtual channels in that they are multiplexed over a single physical channel and each of these dynamic channels have independent FIFO buffers of the same size. However, unlike static virtual channels, where a fixed number of virtual channels are multiplexed over a physical channel, virtual channels, in this case, are allocated dynamically from a common pool. In our case, a new virtual channel is allocated dynamically, if possible, for a high priority packet that would otherwise unnecessarily block. The number of dynamic channels allocated per physical channel is thus flexible and varies depending on the contention for the physical channel. Routers using this dynamic virtual channel allocation therefore prevent head-of-line blocking effectively. In head-of-line blocking, a packet waiting for a blocked link is itself blocking another packet behind it whose target output link is free.

We assume that the total number of virtual channels that can be allocated in a NIC is constant. Initially there is one virtual channel per physical channel, which we refer to as VC-0. A virtual channel is allocated for a packet by the control logic at the transmitting side of a link, which transmits the dynamic virtual channel number along with the packet (similar to the static virtual channel allocation case). At the receiving side, if the dynamic channel for an arriving packet does not already exist, then it is allocated to the physical channel of the incoming packet. A dynamic channel, once allocated, is released when it contains no more data.

Dynamic virtual channels can be implemented with a simple extension to the hardware used to support static virtual channels. At the receiving side of a node, when a packet arrives, it is buffered in the specified virtual channel buffer (if the virtual channel has been allocated to the physical channel). When the specified virtual channel does not exist, it will be allocated from a common buffer pool. In the rare case where there are no free common pool buffers,[2] the incoming packet cannot be assigned the specified virtual channel and the packet (header flit) is dropped and a *drop* signal (back to the transmitting node) is asserted. The transmitting node then retransmits the header flit when the drop signal is deasserted. This requires the transmitting node to keep a copy of the header flit when a dynamic virtual channel is requested so that it can later be retransmitted if necessary. This has no performance impact on the system, as it is equivalent to blocking a flit for an extra cycle.

Figure 8.3 presents the hardware required for implementing dynamic virtual channel flow control for one physical channel between a transmitting and a receiving node. Similar to the

[2]This can happen when two or more arriving packets at different physical channels require new virtual channels at the same time and only a subset of requests can be granted.

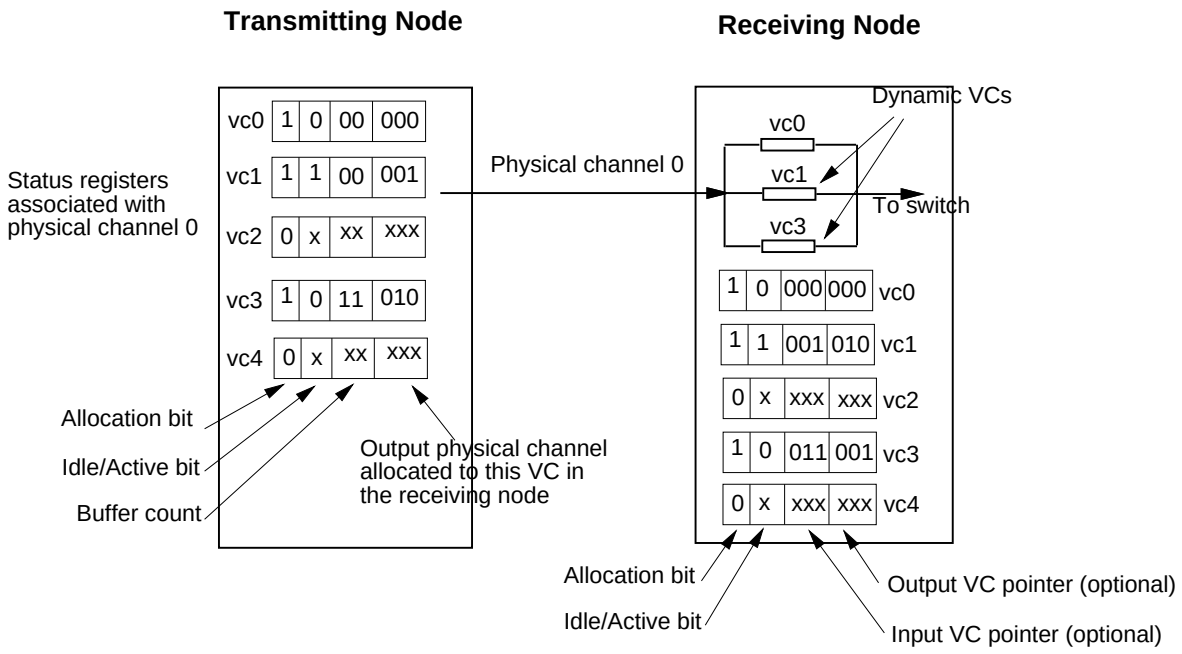

Figure 8.3: Hardware support for dynamic virtual channel flow control is illustrated for one physical channel between a transmitting and a receiving node.

static virtual channel case, the transmitting node contains a status register for each virtual channel on the receiving side. The number of such status registers is equal to the maximum number of possible dynamic virtual channels. The status register contains an allocation bit to identify whether the virtual channel has been allocated to a physical channel and, if allocated, a bit to indicate whether it is idle or active, and a count of the number of free buffers associated with that virtual channel. In addition to the above, to avoid head-of-line blocking, 3 bits are required to store the physical channel number assigned to the packet at the head of virtual channel buffer in the receiving node. The receiving side contains a status register for each virtual channel; the register contains an allocation bit and an idle/active bit. The status register storage requirement per physical channel is therefore given by,

$$S_{pc} = N_{max}(\log(B_{vc}) + 2 + 3) + 2 \cdot N_{max} \tag{8.2}$$

where the first and second term represents the storage requirement at the transmitting and receiving side of a node, respectively, B_{vc} is the number of flit buffers per dynamic virtual channel, and N_{max} is the number of maximum virtual channels that can be assigned. For $N_{max} = 4$, and $B_{vc} = 4$ flits, the status buffer storage requirement is 36 bits.

With respect to channel width overhead, similar to the static virtual channel case, we need to identify the virtual channel number both in the forward direction that is transmitted along with the packet and in the reverse direction that is transmitted along with the *freed* signal.

In addition, an extra wire is required for the *drop* signal that is asserted when a header flit is dropped.

8.4 Prioritized Direct Networks

In our implementation of a prioritized direct network, initially one virtual channel, VC-0, is statically assigned to each physical channel. In addition to the VC-0s, there is a pool of virtual channels that are allocated to physical channels dynamically. Low-priority packets may only use VC-0s, but high priority packets, may use VC-0 or, if they would otherwise block, a dynamically assigned channel. We use a three step process to allocate output links. Output links are first allocated to packets buffered in dynamic virtual channels excluding VC-0s, as all these packets have high priority. Among competing higher priority requests, we allocate the output link to the oldest packet. Second, we assign output links to high priority packets, if any, at the head of the processor input queue. Finally, lower priority packets in the VC-0s and at the processor input queues are assigned output links in that order. Since we have independent virtual channels for time-constrained traffic in the network, we need to apply priority inheritance only at processor input queues, as that is the only place where priority inversion can occur.

In this section, we show how effective priority networks are in reducing latency, improving throughput, and improving system predictability by reducing worst-case latency. We do this using detailed flit-level simulations of a 2-dimensional mesh-connected network, extended with priority-based link arbitration, priority inheritance, and dynamic virtual channels. Although our evaluations are for two priority levels, a high and a low priority level, it could be extended to multiple priority levels. We also show that dynamic virtual channels can be used to support multiple classes of traffic. For this purpose, we consider two traffic classes, namely best-effort traffic and time-constrained traffic. We show that the priority network is effective in reducing the worst-case communication latency of time-constrained traffic, while not penalizing best-effort traffic.

8.4.1 Priority Traffic for Traditional Applications

Even with no time-constrained traffic, it can make sense to assign priorities to different classes of packets if it benefits that class or the traffic overall. For example, in a shared-memory multiprocessor, one can consider giving a higher priority to large packets containing data or to shorter packets containing requests or acknowledgments. Large packets consume more network resources (e.g., links and buffers) than short packets, and when a large packet is blocked in the network, it will unnecessarily block other packets, thereby reducing system throughput.

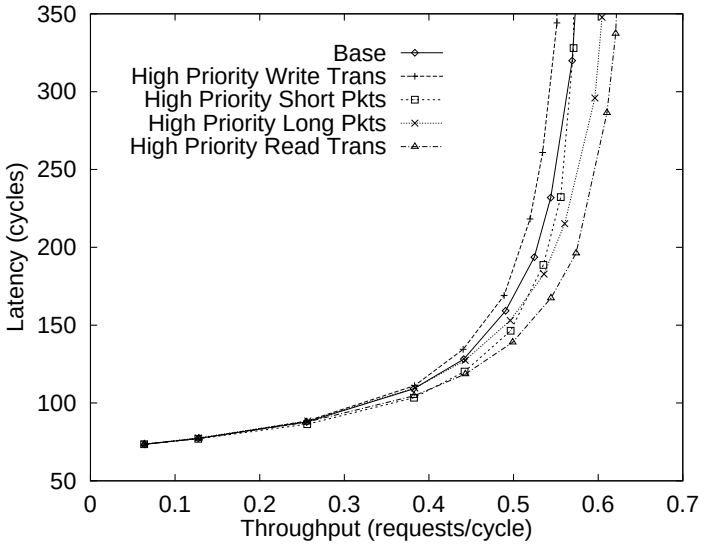

Figure 8.4: Throughput versus latency curves for a 64 processor 8×8 wormhole switched prioritized network. Curves are drawn for the base case for non-prioritized network with no dynamic channels (for comparison purposes) and cases where higher priority is given based on packet size (longer or shorter) and transaction type (read or write).

By giving priority to large packets, they will be removed from the network sooner, thereby reducing the number of packets they can block. On the other hand, by giving priority to short packets, we prevent them from being unnecessarily blocked by large packets. It is also possible to prioritize packets according to transaction type i.e., read and write transactions.

Figure 8.4 shows how a prioritized network can improve system performance. It presents the throughput-latency curves for five different cases: the base case is for non-prioritized networks with no dynamic or static channels; for the other curves priority is given to large packets, short packets, read transactions, and write transactions. This is for a 64 processor 8×8, mesh-connected system with 32-byte cache lines and $T_{uniform}$ workload. For the prioritized networks, we assume a virtual channel buffer size of 3 flits and that the maximum number of dynamic virtual channels is 4. For the base case, the network input buffer is twice the size (6 flits) compared to the prioritized network; under the assumption of equal memory resources. As can be seen from the figure, the highest throughput is achieved when read transactions (i.e., read request and read response packets) are given high priority. Also, giving priority to large packets results in better performance than giving priority to short packets, but giving priority for write transactions results in worse performance than in the base case. A possible explanation for this is that since the number of read transactions is far more than the number of write transactions, giving priority to read transactions will result in a higher dynamic channel buffer utilization when compared to high-priority write transactions.

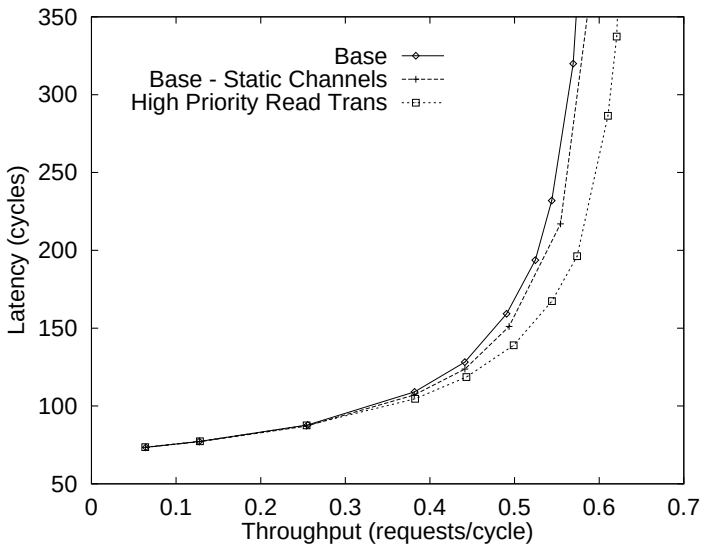

Figure 8.5: Throughput versus latency curves for a 64 processor 8×8 wormhole switched network. Curves are drawn for the base case for non-prioritized network with no dynamic or static virtual channels (for comparison purposes), for the non-prioritized network with two virtual channels per physical channel, and for the prioritized network with high priority read transactions.

To show that a prioritized network with dynamic virtual channels performs better than a non-prioritized network with the same number of static virtual channels, Figure 8.5 compares the throughput-latency curves for the prioritized network with high priority read transactions with a non-prioritized network with two static virtual channels per physical channel (requiring the same memory resources (for buffers) as a prioritized network with four dynamic virtual channels). In the latter case, we use static virtual channels to improve system throughput by preventing head-of-line blocking.

To measure the impact of prioritized networks on the predictability of the system, we plot the worst-case latency in Figure 8.6 for both the non-prioritized network and for the prioritized variant with high priority for read transactions. For comparison purposes we also present the average latency curves. We used the batch-mean analysis method, where the average latency is computed as the grand average of all batch averages and the worst-case latency is computed as the average of all batch worst-case latencies. The errorbars on the worst-case latencies show the global worst-case over all batches (the top end) and the global best case over all batches (the bottom end). It is obvious from the plot that although the average latency is small, the worst-case latency can be as high as a factor 50 higher. The unpredictability in the worst-case latency values shown by the length of the errorbars leads to jitter or unpredictable variance in individual transaction latency. The prioritized network substantially reduces the average worst-case latency and the variance by more than a factor of 2, thereby improving the predictability

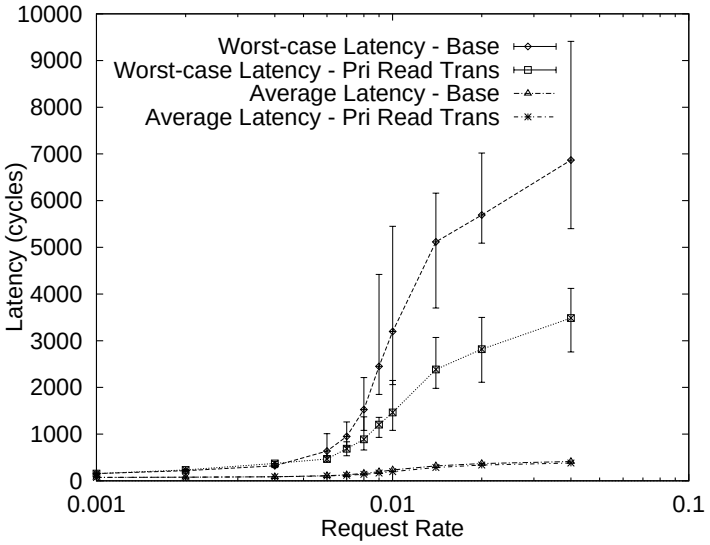

Figure 8.6: Worst-case latency versus request rate for a 64 processor 8×8 wormhole switched prioritized network. Curves are drawn for the base case for non-prioritized network with no dynamic channels and for the case where read transaction is given higher priority. The worst-case latency is the average over all batches with the absolute maximum and minimum values are shown as errorbars.

of the system.

When priority is given to short packets as opposed to read transactions, then there is no reduction either in the average worst-case latency or in their variance. In fact we see a higher worst-case latency in this case than the base case, as shown in Figure 8.7. We can conclude that although dynamic channels can improve system predictability it is sensitive to the priority scheme used.

8.4.2 Time-constrained Traffic

Two classes of uniform traffic pattern

Here, we consider a mix of two classes of traffic: (*i*) *best-effort* traffic with uniformly distributed destinations and an exponentially distributed inter-arrival time between requests, and (*ii*) *time-constrained* traffic with destinations uniformly distributed, but with a fixed inter-arrival time between requests, as seen in multiprocessor video servers [61]. In our simulations, a processor is allowed to have 2 outstanding best-effort requests and 2 outstanding time-constrained requests for a total of 4 outstanding requests, before it is required to block for a reply.[3] For best-effort traffic, we assume 32-byte cache lines are being transferred. The batch termination criterion

[3]In this model, the time-constrained and best-effort requests are interleaved and can be assumed to be equivalent to having a main processor and a co-processor with the former issuing best-effort requests while the latter issuing time-constrained requests independent of each other.

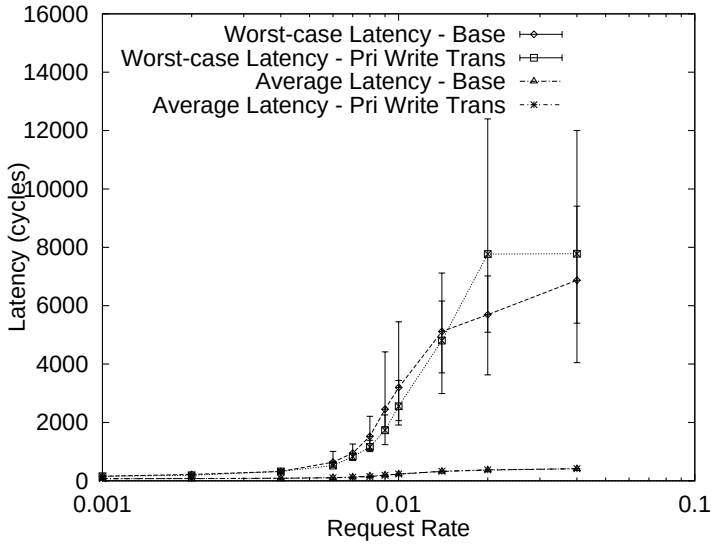

Figure 8.7: Worst-case latency versus request rate for a 64 processor 8×8 wormhole switched prioritized network. Curves are drawn for the base case for non-prioritized network with no dynamic channels and for the case where shorter packets are given priority.

is that all processors have to complete both a minimum number of best-effort requests and a minimum number of time-constrained requests.

Our measures of performance are the worst-case and the average round-trip access latency for time-constrained and best-effort requests, respectively. We assume a total of 4 dynamic channels and that the network input buffer size is 3 flits. In all our experiments we vary the request rate of best-effort traffic and measure the worst-case latency of time-constrained requests and the average latency of best-effort requests. The inter-arrival time of time-constrained requests remains constant at 1 in 1000 processor cycles. A 2-dimensional 64 processor 8×8, wormhole switched mesh-connected system was simulated for this experiment.

Figure 8.8a presents the average over all batches of worst-case communication latency of time-constrained requests with errorbars indicating the absolute maximum and minimum values over all batches. There are two such curves, the top one is for the base case of a non-prioritized network with no dynamic channels, whereas the bottom one is for a prioritized network with 4 dynamic channels per node. The latter gives priority to time-constrained traffic. It is clear that a prioritized network is effective in reducing the worst latency of time-constrained requests by more than 50% and in reducing the variance in the latency. In particular, the prioritized network is very effective in reducing the absolute maximum worst-case latency of time-constrained requests, thereby improving the predictability of the network.

Figure 8.8b presents the average latency of the best-effort requests as a function of best-effort request rate. The graph shows that even though we give priority to time-constrained

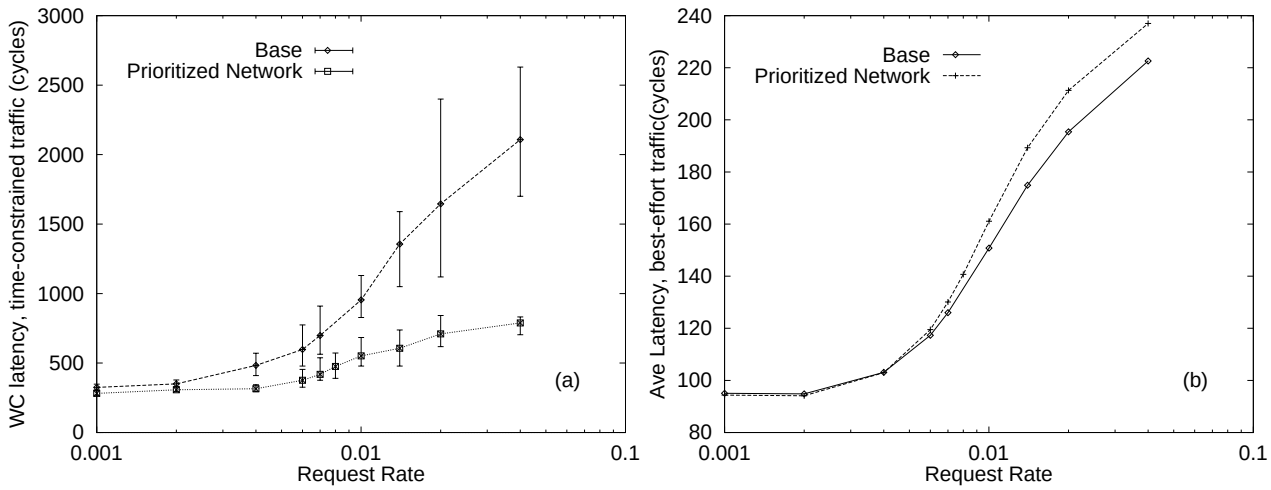

Figure 8.8: (a) Worst-case latency of time-constrained requests and (b) average latency of best-effort requests as a function of best-effort request rate for an 8×8 64 processor wormhole switched prioritized network. Curves are drawn for the base case of a non-prioritized network with no dynamic channels and for the case with dynamic channels, where time-constrained packets are given priority.

requests, the average latency of best-effort requests does not get significantly worse. This is mainly because now there is now less contention for VC-0s that are used to route best-effort traffic.

Non-uniform best-effort traffic pattern

Here, we consider non-uniform best-effort traffic and study how it affects the worst-case latency of time-constrained requests. We assume a *bit-complement* best-effort traffic pattern. The bit-complement permutation requires a source node (x, y) to communicate with the destination node $(X_{max} - x, Y_{max} - y)$, where $X_{max}(Y_{max})$ is the number of nodes in a row (column). Consequently, all packets must eventually cross both the middle row and middle column, congesting the center of a 2-dimensional mesh network. For time-constrained requests, as before, we assume uniformly distributed destinations with a constant inter-arrival time.

Figure 8.9a presents the worst-case latency for time-constrained requests for the same network considered in the previous section. The bit-complement traffic pattern significantly affects the worst-case latency of time-constrained requests in a non-prioritized network as shown with the upper curve. An important observation is that the worst-case latency grows rapidly with an increase in best-effort traffic. We observe from the lower curve that a prioritized network can again be very effective in reducing the worst-case latency of time-constrained requests by an order of magnitude. Another benefit of the prioritized network with a non-uniform best-effort memory access pattern, is a significant reduction in the average latency of best-effort requests at

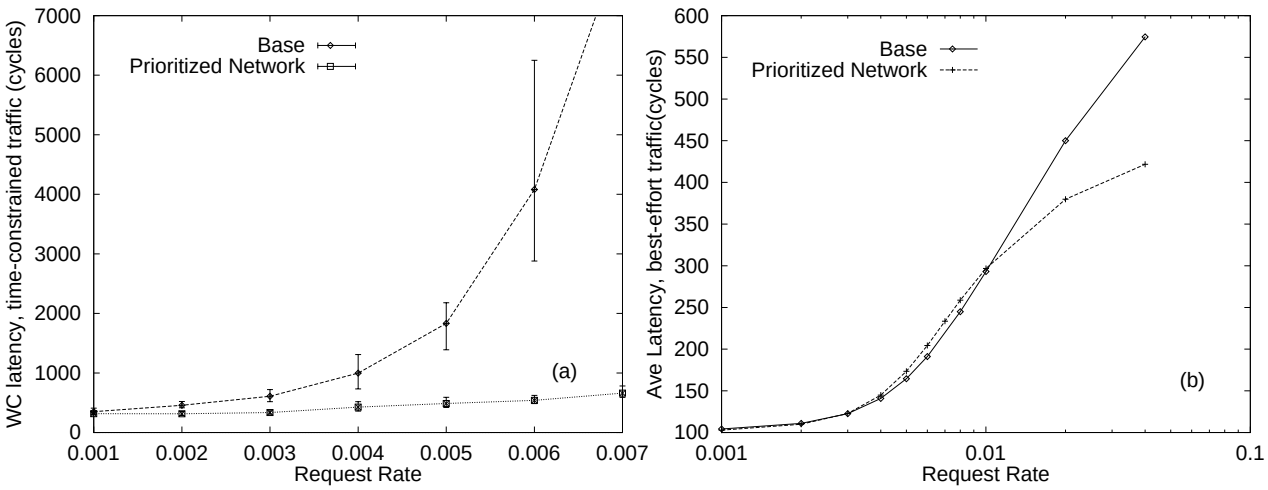

Figure 8.9: (a) Worst-case latency of time-constrained requests and (b) average latency of best-effort requests as a function of best-effort request rate for an 8×8 64 processor wormhole switched prioritized network. A non-uniform bit complement memory access pattern is used for best-effort requests. Curves are drawn for the base case of a non-prioritized network with no dynamic channels and for the case with dynamic channels, where time-constrained packets are given priority.

high request rates when compared to the non-prioritized network. This is shown in Figure 8.9b where there is a 25% reduction in the average latency of best-effort requests at a high request rate of 0.04 when a prioritized network is used.

Chapter Summary

In this chapter, we proposed and evaluated prioritized connectionless shared-memory multiprocessor networks. In our implementation of prioritized networks, we used three main components namely priority-based link arbitration, priority inheritance, and dynamic virtual channels. It was shown that a prioritized network can significantly reduce average transaction latencies and improve system throughput when running traditional parallel applications. It was also shown how a prioritized network could be used to reduce the worst-case latencies of time-constrained traffic when it co-exists with best-effort traffic. One of the key aspects of the prioritized network is that they do not increase the average latency of best-effort traffic while they improve that of time-constrained traffic, independent of the best-effort traffic pattern.

Chapter 9
Conclusion

This dissertation concentrated on performance issues in the design of high-performance shared-memory multiprocessor networks. In particular we studied low-dimensional direct and hybrid hierarchical-ring networks. Two-dimensional direct networks are currently popular in research and commercial environments, whereas hierarchical-ring networks present an interesting alternative to direct networks from a performance and practicality point of view. This dissertation makes the following contributions:

- *Comprehensive performance study of shared-memory multiprocessor interconnection networks*: We believe this study constitutes the first comprehensive performance study of low-dimensional direct and hierarchical-ring interconnection networks for shared-memory systems.

- *Comparative performance study*: We presented a detailed comparative performance evaluation of hierarchical-ring, 2D mesh, 2D torus, and bidirectional ring networks under wormhole switching using both synthetic workload and program-driven simulations. Under constant pin and memory constraints, it was shown that hierarchical-ring networks perform better than 2-dimensional direct networks for system sizes of up to 64 processors at low request rates either when there is high locality in the memory access pattern or for large cache line sizes. However, 2-dimensional direct networks scale well to large system sizes and perform better at high request rates.

- *Topology*: We derived in Chapter 5 several 'optimal' topologies for different sized hierarchical-ring systems using a bottom-up approach. This is important because hierarchical-ring networks are highly configurable.

- *Switching techniques*: We studied the performance of various cut-through switching techniques for hierarchical-ring networks under both blocking and non-blocking flow-control policies. These included wormhole, virtual cut-through, and cell switching techniques.

While non-blocking cell switching is a good choice for hierarchical-ring networks, it has several disadvantages. It requires large buffers to minimize the number of packets dropped and large non-deterministic timeout values to trigger retransmission of dropped packets. On the other hand, blocking buffered wormhole switching requires virtual channels to prevent deadlock although it uses less buffer space.

- *Buffer management*: We studied wormhole switching in low-dimensional direct networks namely the 2D mesh, 2D torus, and the bidirectional rings under blocking flow-control and under the assumption that there is no distinction between a flit and a phit. We showed that buffered wormhole switching in direct networks results in a good trade-off between performance and NIC buffer space requirements. We also studied buffer management issues and the impact of router buffer sizes on system performance and found that the optimal router buffer sizes are sensitive to the system cache line size.

- *Routing*: For wormhole switched hierarchical-ring network, we proposed a deadlock free deterministic routing technique that uses a virtual channel approach.

- *Dynamic virtual channels*: We proposed dynamic virtual channel flow-control for 2-dimensional direct networks, and proposed and evaluated priority networks using dynamic virtual channels, priority based link arbitration, and priority inheritance. We showed that such priority networks can be used to improve system throughput and support multiple classes of traffic.

9.1 Future Work

There are plenty of directions in which the work described here can be extended:

- **Bisection bandwidth**: The performance and scalability of hierarchical-ring networks are clearly limited by their constant bisection bandwidth. In Chapter 5, we showed through an empirical model that increasing the bandwidth of the global ring (and thus the bisection bandwidth) allows the network to support more processors. Targeting just the global ring is effective, because the utilization of the lower level rings is low, especially when the global ring is saturated. The bandwidth of the global ring can be increased either by increasing the width of the ring or the speed of the ring. It will be interesting to study in detail the performance impact of such an increase in bisection bandwidth.

- **Hybrid Flow-control**: Hybrid flow-control combines blocking and non-blocking flow-control, and may be able to exploit the advantages of both. In a blocking network, a

blocked packet can span multiple nodes, thereby occupying link resources and preventing other packets from using those links. We showed that blocking networks suffer from early saturation before they exhaust their full bandwidth (if small buffers are used), and therefore perform poorly under high request rates. On the other hand, non-blocking networks drop packets when they cannot be buffered at a node; negative acknowledgments and time-outs are used to recover dropped packets. It is non-trivial to find a good time-out value in non-blocking networks. A larger than required time-out value results in higher latency while a small value results in duplicate packets. Also, to reduce the number of dropped packets, especially under high request rates, it is necessary to use large buffers in routers, as shown in Chapter 6, further increasing the time-out value. Another disadvantage of non-blocking networks is that the hardware cache consistency mechanism cannot tolerate the loss of invalidation packets. In hybrid flow-control, the decision to drop or block a packet can be taken on-line depending on the nature of traffic. One possibility is to drop a packet (at the head of a buffer) only if it has been blocked more than a certain time, thus freeing up buffer space. When to drop a packet can also depend on the type of the packet. For example, a request packet might be dropped sooner than a response packet, and negative acknowledgments and invalidation packets might never be dropped. It would be interesting to study the performance of hybrid flow-control under various such policies.

- **Effect of Adaptive Routing on Shared-memory Networks**: There have been extensive studies of adaptive routing protocols on 2-dimensional direct networks in the context of distributed memory multiprocessors. Almost all those studies used wormhole switching with single flit buffers and constant packet sizes. It is not clear how adaptive routing will be effective in shared-memory networks under buffered wormhole switching and variable sized packets. It would also be interesting to study the performance impact of adaptive routing with static and dynamic virtual channels.

9.2 Impact of this Research and Applicability to Industry

We believe that the results of this research will influence the design of shared-memory multiprocessor networks, in particular the design of hierarchical rings, 2D meshes and tori. An interconnection network with low latency and high throughput allows more work to be accomplished in a given period of time. We expect the market for high performance parallel processors to rise in the near future, especially as parallel processing provides a cost effective way to improve computer system performance. Recently, we have seen small scale parallel systems become popular for the same reason. These systems will become much more popular

since multimedia applications, visualizations, and 3D modeling are all becoming more computationally intensive; hence, using multiple off-the-shelf processors to exploit parallelism in those applications is becoming very attractive. As these small scale systems become more common place, they will naturally be extended to larger sizes. For these systems, the results of our research will be relevant, as high performance routers are an essential component in any high performance parallel system.

Bibliography

[1] S.V. Adve, V.S. Adve, M.D. Hill, and M.K. Vernon, "Comparison of hardware and software cache coherence schemes," *Proc. Intl. Symp. on Computer Architecture*, pp. 298-308, May 1991.

[2] V.K. Adve and M.K. Vernon, "Performance analysis of mesh interconnection networks with deterministic routing," *IEEE Trans. on Parallel and Distributed Systems*, Vol. 5, No. 3, pp. 225-246, March 1994.

[3] A. Agarwal, "Limits on interconnection network performance," *IEEE Trans. on Parallel and Distributed Systems*, Vol. 2, No. 4, pp. 398-412, April 1991.

[4] A. Agarwal, R. Bianchini, D. Chaiken, K. L. Johnson, D. Kranz, J. Kubiatowicz, B. Lim, K. Mackenzie, and D. Yeung, "The MIT Alewife machine: Architecture and performance," *Proc. Intl. Symp. on Computer Architecture*, pp. 2-13, 1993.

[5] L.A. Barroso and M. Dubois, "Performance evaluation of the slotted ring multiprocessor," *IEEE Trans. on Computers*, vol.44, no.7, pp. 878-890, July 1995.

[6] R. Berrendorf *et. al.*, "Intel Paragon XP/S - Architecture, software environment and performance," *http://www.kfa-juelich.de/zam/CompServ/services/sco.paragon.html*

[7] R. V. Boppana and S. Chalasani, "A framework for designing deadlock-free wormhole routing algorithms," *IEEE Trans. on Parallel and Distributed Systems*, Vol. 7, No. 2, pp. 169-183, February 1996.

[8] G. A. Boughton, "Arctic routing chip," *Proc. Intl. Workshop on Parallel Computer Routing and Communication,* pp.311-317, May 1994.

[9] URL: http://www.cs.washington.edu/research/projects/lis/chaos/www/bnf.html

[10] T. Callahan and S. C. Goldstein, "NIFDY: A low overhead, high throughput network interface," *Proc. Intl. Symp. on Computer Architecture*, pp. 230-242, May 1995.

[11] J. Chapin, "A fresh look at memory hierarchy management," *Proc. HOT OS-VI*, 1997.

[12] M. S. Chen, K. G. Shin, and D. D. Kandlur, "Addressing and routing issues in hexagonal mesh multiprocessors," *IEEE Trans. on Computers*, Vol. 39, No. 1, pp. 10-18, January 1990.

[13] A. A. Chien, "A cost and speed model for k-ary n-cube wormhole routers," in *Proc. Hot Interconnects '93*, pp 3.1.1-3.1.7, August 1993.

[14] A. A. Chien and J. H. Kim, "Planar-adaptive routing: Low-cost adaptive networks for multiprocessors," *Proc. Intl. Symp. on Computer Architecture*, pp. 268-277, 1992.

[15] DRAM Synchronous DRAM DIMM 10ns, 100MHz, 3.3V Ram Chips, *Corsair Microsystems*, URL: http://www.nf-ny.com/nfny/sdram.html

[16] Cray Research Inc. "Cray Origin2000 system overview", URL: http://www.cray.com/products/systems/origin2000/

[17] W. J. Dally and C. L. Seitz, "The torus routing chip," *Journal of Distributed Computing*, Vol. 1, No. 3, pp 187-196, March 1986.

[18] W. J. Dally and C. L. Seitz, "Deadlock-free message routing in multiprocessor interconnection networks," *IEEE Trans. on Computers*, Vol. C-36, No. 5, pp. 547-553, May 1987.

[19] W. J. Dally, "Virtual-channel flow control," *IEEE Trans. on Parallel and Distributed Systems*, vol. 3, no. 2, pp. 194-205, March 1992.

[20] W. J. Dally, "Performance analysis of k-ary n-cube interconnection networks," *IEEE Trans. on Computers*, Vol. 39, No. 6, pp. 775-785, June 1990.

[21] W. J. Dally and H. Aoki, "Adaptive routing using virtual channels," *IEEE Trans. on Parallel and Distributed Systems*, Vol. 4, No. 4, pp. 466-475, April 1993.

[22] W. J. Dally, L. R. Dennison, D. Harris, K. Kan, and T. Xanthopoulos, "Architecture and implementation of the reliable router," *Proc. of Hot Interconnects II*, August 1994.

[23] W. J. Dally *et. al.*, "The message-driven processor: A multicomputer processing node with efficient mechanisms," *IEEE Micro*, April 1992.

[24] J. Duato, P. Lopez, and S. Yalamanchilli, "Deadlock- and livelock-free routing protocols for wave switching," *Proc. Intl. Parallel Processing Symposium,* June 1997.

[25] J. Duato, "A new theory of deadlock-free adaptive routing in wormhole networks," *IEEE Trans. on Parallel and Distributed Systems*, Vol. 4, No. 12, pp. 1320-1331, December 1993.

[26] J. Duato, "A necessary and sufficient condition for deadlock-free routing in cut-through and store-and-forward networks," *IEEE Trans. on Parallel and Distributed Systems*, Vol. 7, No. 8, pp. 841-854, August 1996.

[27] J. Duato, "Performance evaluation of adaptive routing algorithms for k-ary n-cubes," *Proc. Intl. Workshop on Parallel Computer Routing and Communication,* pp. 44-59, May 1994.

[28] T. H. Dunigan, "Multi-ring performance of the Kendall Square multiprocessor," *Oak Ridge National Laboratory Report TM-12331*, October 1994.

[29] K.I. Farkas, Z. Vranesic, and M. Stumm, "Scalable cache consistency for hierarchically structured multiprocessors," *The Journal of Supercomputing*, Vol. 8, No. 4, pp. 345-369, 1992.

[30] R. A. Finkel and M. H. Solomon, "Processor interconnection strategies," *IEEE Trans. on Computers*, Vol. C-29, No. 5, pp. 360-371, May 1980.

[31] R. M. Fujimoto, "VLSI communication components for multicomputer networks," *Ph.D. Dissertation*, Technical Report No. UCB/CSD 83/136, University of California, Berkeley, California, 1983.

[32] M. Gerla and L. Kleinrock, "Flow Control: A comparative survey," *IEEE Trans. on Communications*, Vol. COM-28, No. 4, pp. 553-574, April 1980.

[33] K. Gharachorloo, A. Gupta, and J. Hennessy, "Hiding memory latency using dynamic scheduling in shared-memory multiprocessors," *Proc. Intl. Symp. on Computer Architecture*, pp. 22-35, May 1992.

[34] C. J. Glass and L. M. Ni, "The Turn model for adaptive routing," *Proc. Intl. Symp. on Computer Architecture*, pp. 278-287, 1992.

[35] I. S. Gopal, "Prevention of store-and-forward deadlock in computer networks," *IEEE Trans. on Communications*, Vol. 33, pp. 1258-1264, December 1985.

[36] A. Gupta, J. Hennessy, K. Gharachorloo, T. Mowry, and W. D. Weber, "Comparative evaluation of latency reducing and tolerating techniques," in *Proc. Intl. Symp. on Computer Architecture*, pp. 254-265, May 1991.

[37] D.B. Gustavson, "SCI and related standards projects," *IEEE Micro*, Vol. 12, No. 1, pp. 10-22, Jan 1992.

[38] V.C. Hamacher and H. Jiang, "Comparison of mesh and hierarchical networks for multiprocessor," *Proc. Intl. Conf. on Parallel Processing*, Vol. I, pages 67-71, August 1994.

[39] V.C. Hamacher and H. Jiang, "Performance and configuration of hierarchical ring networks for multiprocessors," *Proc. Intl. Conf. on Parallel Processing*, Vol. I, August 1997.

[40] K. A. Harzallah and K. C. Sevcik, "Hot spot analysis in large scale shared-memory multiprocessors," *Proc. Supercomputing '93*, November 1993.

[41] R. A. Hexel, *A quantitative performance evaluation of SCI memory hierarchies*, Ph.D. Dissertation, University of Edinburgh, 1994.

[42] M. Holliday and M. Stumm, "Performance evaluation of hierarchical ring-based shared memory multiprocessors", *IEEE Trans. on Computers*, Vol. 43, No. 11, pp. 52-67, Jan 1994.

[43] A. Hooper and R.C. Williamson, "Design and use of an integrated cambridge ring", *IEEE Journal on Selected Areas of Communication*, 1(5), pp. 775-784, 1983.

[44] D. E. Huber, W. Steinlin, and P. J. Wild, "SILK: An implementation of a buffer insertion ring," *IEEE Journal on Selected Areas in Communications*, Vol. SAC-1, No. 5, pages 766-774, November 1983.

[45] *Intel iPSC860 system overview*, Intel Scalable Systems Division, Intel Corporation.

[46] S. S. Isloor and T. A. Marsland, "The deadlock problem: An overview," *IEEE Computer*, Vol. 13, No. 9, pp. 58-78, June 1979.

[47] M. Jaseemuddin, *Bidirectional ring: An interconnection network for shared-memory multiprocessor systems*, Ph.D. Dissertation, Department of Electrical and Computer Engineering, University of Toronto, September 1997.

[48] S. Jha and M. Fry, "Continuous media playback and jitter control," *Proc. Intl. Conf. on Multimedia Computing and Systems*, pp. 245-252, June 1996.

[49] A. E. Joel, "Circuit switching: Unique architecture and applications," *IEEE Computer*, Vol. 12, No. 6, pp. 10-22, June 1979.

[50] P. Kermani and L. Kleinrock, "Dynamic flow-control in store and forward computer networks," *IEEE Trans. on Communications*, Vol. COM-27, Feburary 1979.

[51] P. Kermani and L. Kleinrock, "Virtual cut-through: A new computer communication switching technique," *Computer Networks*, Vol. 3, No. 4, pp. 267-286, September 1979.

[52] J. H. Kim and A. A. Chien, "Rotating combined queuing (RCQ): Bandwidth and latency guarantees in low-cost, high-performance networks," *Proc. Intl. Symp. on Computer Architecture*, pp. 226-236, May 1996.

[53] S. Konstantinidou and L. Snyder, "Chaos router: Architecture and performance," *Proc. Intl. Symp. on Computer Architecture*, pp. 212-221, May 1991

[54] A. Kumar and L. N. Bhuyan, "Evalauting virtual channels for cache-coherent shared-memory multiprocessors," *Proc. Intl. Conf. on Supercomputing*, May 1996.

[55] J. Kuskin, D. Ofelt, M. Heinrich, J. Heinlein, R. Simoni, K. Gharachorloo, J. Chapin, D. Nakahira, J. Baxter, M. Horowitz, A. Gupta, M. Rosenblum, and J. Hennessy, "The Stanford FLASH multiprocessor," *Proc. Intl. Symp. on Computer Architecture*, pp. 302-313, April 1994.

[56] J. Laudon and D. Lenoski, "The SGI Origin: A ccNUMA highly scalable server," *Proc. Intl. Symp. on Computer Architecture*, pp. 241-251, June 1997.

[57] D. Lenoski, J. Laudon, T. Joe, D. Nakahira, L. Stevens, A. Gupta, and J. Hennessy, "The DASH prototype: Logic overhead and performance," *IEEE Trans. on Parallel and Distributed Systems*, Vol. 4, No. 1, pp. 41-61, Jan 1993.

[58] D. H. Linder and J. C. Harden, "An adaptive and fault-tolerant wormhole routing strategy for k-ary n-cubes," *IEEE Trans. on Computers*, Vol. 40, No. 1, pp. 2-12, January 1991.

[59] M.H. MacDougall, *Simulating Computer Systems: Techniques and Tools*, MIT Press, 1987.

[60] P. M. Merlin and P. J. Schweitzer, "Deadlock avoidance - store and forward deadlock," *IEEE Trans. on Communications*, Vol. COM-28, No. 3, pp. 345-354, March 1980.

[61] A. L. Narasimha Reddy, "Scheduling and data distribution in a multiprocessor video server," *Proc. Intl. Conf. on Multimedia Computing and Systems*, pp. 256-263, May 1995.

[62] J. Y. Ngai and C. L. Seitz, "A framework for adaptive routing in multicomputer networks," *Proc. Intl. Symp. on Computer Architecture*, pp. 6-14, March 1991.

[63] L. M. Ni and P. K. McKinley, "A survey of wormhole routing techniques in direct networks," *IEEE Computer*, pp. 62-76, February 1993.

[64] Noakes, Michael D, Deborah A, and W. J. Dally, "The J-machine multicomputer: An architectural evaluation," *Proc. Intl. Symp. on Computer Architecture*, 1993.

[65] S. F. Nugent, "The iPSC/2 direct-connect technology," *Proc. ACM Conf. Hypercube Concurrent Computers and Applications*, pp. 51-60, 1988.

[66] H. Oi and N. Ranganathan, "Performance analysis of the bidirectional ring-based multiprocessor," *Proc. Intl. Conf. on Computer Design*, October 1997.

[67] G. Pfister and A. Norton, "Hot-spot contention and combining in multistage interconnect networks," *IEEE Trans. on Computers*, Vol. C-32, No. 10, pp. 943-948, 1995.

[68] G. F. Pfister, W. C. Brantley, D. A. George, S. L. Harvey, et. al., "The IBM research parallel processor prototype (RP3): Introduction and architecture," *Proc. Intl. Conf. on Parallel Processing*, pp. 764-771, August 1985.

[69] F. P. Preparata and J. Vuillemin, "The Cube Connected Cycles: A versatile network for parallel computation," *Communications of the ACM*, Vol. 24, No. 5, pp. 300-309, May 1981.

[70] R. Rajkumar, "Synchronization in real-time systems: A priority inheritance approach," *Kluwer Academic Publishers*, ISBN 0-7923-9211-6 pp. 15-58, 1991.

[71] G. Ravindran and M. Stumm, "Hierarchical ring topologies and the effect of their bisection bandwidth constraints," in *Proc. Intl. Conf. on Parallel Processing*, pp. I/51-55, August 1995.

[72] G. Ravindran and M. Stumm, "A comparison of blocking and non-blocking packet switching techniques in hierarchical ring networks," in *IEICE Trans. on Information and Systems*, Vol. E79-D, No. 8, pp. 1130-1138, August 1996.

[73] G. Ravindran and M. Stumm, "A performance comparison of hierarchical ring- and mesh-connected multiprocessor networks," *Proc. Intl. Symp. on High Performance Computer Architecture*, pp. 58-71, February 1997.

[74] G. Ravindran and M. Stumm, "Issues in the design of direct multiprocessor networks," *Technical Report*, CSRI, University of Toronto, November 1997.
Also available at http://www.eecg.toronto.edu/gravin/pub/wh.ps.gz

[75] G. Ravindran and M. Stumm, "Prioritized Direct Multiprocessor Networks: Design and Performance," *Submitted for Publication.*
Also available at http://www.eecg.toronto.edu/gravin/pub/pri.ps.gz

[76] D. A. Reed and R. M. Fujimoto, *Multicomputer networks: Message-based parallel processing*, ISBN: 0-262-18129-0 The MIT Press, Cambridge, Massachusetts 1987.

[77] J. Rexford and K. G. Shin, "Support for multiple classes of traffic in multicomputer routers," *Proc. Intl. Workshop on Parallel Computer Routing and Communication*, pp. 117-129, May 1994.

[78] J. Rexford, J. Hall and K. G. Shin, "A router architecture for real-time point-to-point networks," *Proc. Intl. Symp. on Computer Architecture*, pp. 237-246, May 1996.

[79] E. Rothberg, J. P. Singh, and A. Gupta, "Working sets, cache sizes and node granularity for large-scale multiprocessors," *Proc. Intl. Symp. on Computer Architecture*, pp. 14-25, May 1993.

[80] S. Scott and G. Thorson, "Optimized routing in the Cray T3D," *Proc. Intl. Workshop on Parallel Computer Routing and Communication*, pp. 280-294, May 1994.

[81] S. L. Scott and G. M. Thorson, "The Cray T3E network: Adaptive routing in a high performance 3D torus," *Proc. Hot Interconnects IV*, August 1996.

[82] S. Scott, J. R. Goodman, and M. K. Vernon, "Performance of SCI ring," *Proc. Intl. Symp. on Computer Architecture*, pp. 403-414, 1992.

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

[84] URL: http://www.sequent.com/

[85] K.G. Shin and S.W. Daniel, "Analysis and implementation of hybrid switching," *IEEE Trans. on Computers*, Vol. 45, No. 6, pp. 684-292, June 1996.

[86] J. P. Singh, W. Weber, and A. Gupta, "SPLASH: Stanford Parallel Applications for Shared-memory," *Computer Architecture News*, Vol. 20, No.1, pp. 5-44, 1992.

[87] URL: http://www-flash.stanford.edu:80/apps/SPLASH/

[88] P. Stenstrom, "A survey of cache coherence schemes for multiprocessors," *IEEE Computer*, Vol. 23, No. 6, pp. 12-24, June 1990.

[89] Anjan K. V. and T. M. Pinkston, "An efficient, fully adaptive deadlock recovery scheme: DISHA," *Proc. Intl. Symp. on Computer Architecture*, June 1995.

[90] J. E. Veenstra and R. J. Fowler, "MINT: A front end for efficient simulation of shared-memory multiprocessors," *Proc. Intl. Workshop on Modelling, Analysis, and Simulation of Computer and Telecommunication Systems (MASCOTS)* pp. 201-207, January 1994

[91] Z. G. Vranesic, V. C. Hamacher, A. K. Sanwalka, and S. G. Zaky, "A hybrid token/insertion ring LAN," Proc. INFOCOM, Vol. 1, pp. 211-220, April 1991

[92] Z. G. Vranesic et al., "The NUMAchine multiprocessor", *Technical Report*, CSRI-TR-324, CSRI, University of Toronto, 1995.

[93] Z. G. Vranesic, M. Stumm, D. Lewis, and R. White, "Hector: A hierarchically structured shared-memory multiprocessor," *IEEE Computer*, Vol. 24, No. 1, pp. 72-78, January 1991.

[94] S. Woo, M. Ohara, E. Torrie, J. P. Singh and A. Gupta. "The SPLASH-2 programs: Characterization and Methodological considerations," *Proc. Intl. Symp. on Computer Architecture*, pp. 24-36, June 1995.

[95] X. Zhang and Y. Yan, "Comparative modelling and evaluation of CC-NUMA and COMA on hierarchical-ring architectures," *IEEE Trans. on Parallel and Distributed Systems*, Vol. 6, pp. 1316-1331, December 1995.