AN ABSTRACT OF THE THESIS OF

Jangmin Choi for the degree of Master of Science in Computer Engineering presented on June, 28, 1995. Title:

Simulation Studies of Routing Algorithms for Multicomputer Systems

Redacted for Privacy

Abstract approved: _____

Ben Lee

Efficient routing of messages is critical to the performance of multicomputers. Many adaptive routing algorithms have been proposed to improve the network efficiency; however, they can make only short-sighted decisions to choose a channel for message routing because of limited information about network condition. The short-sighted decisions may cause unnecessary waits at the next clock cycle if the adaptive routing algorithm chooses a channel which has high probability of message block at the next clock cycle.

In this thesis, look-ahead routing scheme is introduced to provide better performance for conventional adaptive routing algorithms. The look-ahead scheme as an adaptive routing algorithm makes longer-sighted decision to avoid possible blocks at the next clock cycle.

This thesis also simulates the XY, the West-First, and the Double-Y channel routing algorithms, which are deterministic, partially-adaptive and fully-adaptive, respectively, in a 16 × 16 mesh network. Their performances are compared and analyzed. The simulation includes the examination of look-ahead effect for the West-First and the Double-Y channel routing algorithms.

Simulation Studies of Routing Algorithms for Multicomputer Systems

by

Jangmin Choi

A THESIS

submitted to

Oregon State University

in partial fulfillment of
the requirements for the
degree of

Master of Science

Completed June, 28, 1995
Commencement June 1996

Master of Science thesis of Jangmin Choi presented on June 28, 1995

APPROVED:

Redacted for Privacy

Major Professor, representing Electrical and Computer Engineering

Redacted for Privacy

Head of Department of Electrical and Computer Engineering

Redacted for Privacy

Dean of Graduate School

I understand that my thesis will become part of the permanent collection of Oregon State University libraries. My signature below authorizes release of my thesis to any reader upon request.

Redacted for Privacy

Jangmin Choi, Author

ACKNOWLEDGEMENT

First, I am grateful to my wife Eunhyoung for her unconditional support, prayers and patience. My special thanks go to Dr. Ben Lee who was my major advisor. His guidance and advices have been extremely valuable to my studies at Oregon State University which was in a foreign environment. Also, special thanks to the committee members, Dr. James H. Herzog, Dr. Donald L. Amort, and Dr. Goran N. Jovanovic. My heart is filled with thanks to my friend James Hogan who volunteered to edit this thesis. I will never forget his kindness in helping me. And lastly, I would like to thank Republic of Korea Air Force which has financially supported me.

This thesis is dedicated to my wife and lovely two boys - Daehan and Sonul.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

SIMULATION STUDIES OF ROUTING ALGORITHMS FOR MULTICOMPUTER SYSTEMS

1. INTRODUCTION

The goal of modern computer architecture design is to achieve one *teraflops** of computing power [35,2]. It does not appear that the conventional computers or super-computers working in SISD or SIMD† models can reach this goal. This is because their performance is limited by the available semiconductor technology. For example, although VLSI technology supports more than 3 million transistors on a single chip at a clock speed of more than 100 MHz, the speed of Dynamic Random Access Memory (DRAM) has not been able to keep up. Therefore, the performance of any computer with a single Central Processing Unit (CPU) is ultimately limited by the contention problem on the memory bus. This problem, known as the Von Neumann bottleneck, is caused by slow memory response time to fast memory re-

---

*10^{12} floating point operations per second.

†Flynn proposed a simple model to categorize various computer architectures based on the number of parallel instruction and data streams [34].

- Single instruction stream, single data stream (SISD)
- Single instruction stream, multiple data stream (SIMD)
- Multiple instruction stream, single data stream (MISD)
- Multiple instruction stream, multiple data stream (MIMD)

Some architectures are in hybrid nature.

quests from the CPU. In order to avoid this bottleneck, computers must use multiple CPUs and multiple memory modules in parallel.

Exploiting parallelism has been extensively studied in the last decade. Parallel computers or parallel processing does more than break the bottleneck problem. The basic idea of parallel processing is to create powerful computers simply by connecting many existing computers. It is widely accepted that the parallel computers[‡] based on the MIMD model are the most promising way to achieve teraflops of processing power. In [35], Frey and Fox anticipated that the next generation of microprocessors would be developed with performance estimated at over 20 megaflops. Therefore, to achieve a total performance of a teraflops, 2^{15} or 2^{16} uniprocessor nodes would be needed. In other words, teraflops performance can only be achieved by means of massively parallel processing (MPP)[§].

1.1. PARALLEL COMPUTERS

Parallel computers use a collection of uniprocessors operating in parallel. In most MIMD designs, each uniprocessor contains a microprocessor, such as those used in personal computers or advanced workstations.

There are two distinct models of parallel computers: *shared-memory multiprocessors* and *message-passing multicomputers*. Shared-memory multiprocessors have a shared common memory, while message-passing multicomputers have unshared distributed memories. A major shortcoming of shared-memory multipro-

---

[‡]In general, parallel processing means executing programs in the SIMD and the MIMD mode. However, in this thesis, the term parallel processing or parallel computers is limited to those that execute in the MIMD mode.

[§]The term *massively parallel processing* is widely used but is not defined clearly. It usually means parallel processing using more than 100 processors [36].

cessors is the lack of scalability** due to contention over access to the shared memory. To alleviate the memory contention problem, multiprocessors use cache memories. However, this causes a data inconsistency problem, known as the cache coherence problem, because multiple caches attached to processors may have different copies of the same memory block due to asynchronous and independent operations of multiple processors. Therefore, it is difficult to build massively parallel computers using the shared-memory model [2].

Message-passing multicomputers, or simply multicomputers, consist of multiple computers. Each computer, called a node, has its own memory and processor. The uniprocessors in a multicomputer are connected by a point-to-point direct interconnection network, and communication among the nodes is done by explicit message-passing. Therefore, *multicomputers* are more suitable for massively parallel processing because they are more scalable and do not suffer from memory inconsistency and contention problems. Figure 1.1 shows an organization of a typical multicomputer.

1.2. INTERCONNECTION NETWORK

In multicomputers, communication between nodes is natural and indispensable. Since communication in a multicomputer is done by message-passing through the network, the performance of a multicomputer heavily depends on network efficiency and communication protocols.

The network in a multicomputer connects each node to its neighbor nodes. Which neighbor nodes are connected is determined by the *network topology*. The

---

**Scalability is defined as the ability of a network to be modularly expandable with increasing machine resources.

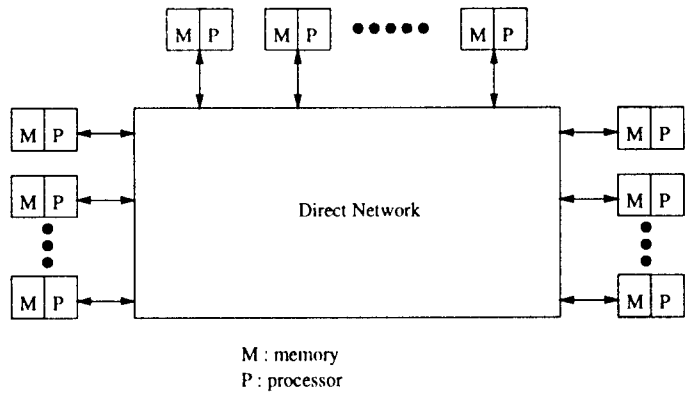


Figure 1.1. A typical parallel computer with a direct network

path which a message packet will travel through to arrive at its destination is defined by *routing algorithms*. *Switching methods* decide how the message packet in an input channel is to be placed in the output channel. Thus, the network behavior of a multicomputer is characterized by the topology, the switching technique, and the routing algorithm.

This thesis deals with routing algorithms. Routing algorithms can be classified either as *deterministic* or *adaptive*. In deterministic routing algorithms, the path is predetermined in the source node using only the destination address, without considering the network conditions. Adaptive routing algorithms can choose a number of alternate paths at every node depending on the network condition to detour around contentions and faulty nodes.

However, the conventional adaptive routing algorithms can make only short-sighted decisions for the next path because most implementations use limited information about the node and the channel that the message currently occupies. These short-sighted decisions may cause unnecessary waits at subsequent clock cycles if a selected path is blocked at the next cycle. If the routing algorithm can look-ahead to

the traffic condition in the network, it can possibly avoid unnecessary waits at subsequent clock cycles. Therefore, the look-ahead scheme will increase the performance of adaptive routing algorithms.

In this thesis, the look-ahead scheme is introduced. Also, three routing algorithms, the *XY*, the *West-First*, and the *Double-Y channel* routing algorithms[††], are simulated as representatives of deterministic, partially-adaptive, and fully-adaptive routing algorithms, respectively. The look-ahead scheme is incorporated to the West-First and the Double-Y channel routing algorithms and then simulated to evaluate its effectiveness.

1.3. THESIS ORGANIZATION

Chapter 2 surveys direct network topologies that have been proposed in the past. In this chapter, several network topologies are examined and compared using some valuable parameters.

Chapter 3 describes various switching techniques. There have been four techniques proposed: circuit switching, store-and-forward routing, virtual cut-through routing, and wormhole routing. Latency and hardware requirements for these techniques are examined.

Chapter 4 discusses the concept of deadlock, virtual channels, and the examples of routing algorithms. This chapter explains deadlock situations and deadlock avoidance techniques. Deterministic and adaptive routing algorithms are also discussed.

Chapter 5 introduces the look-ahead scheme for adaptive routing algorithms.

Chapter 6 discusses the simulation results for the XY routing algorithm as a deterministic routing algorithm, the West-First routing algorithm as a partially-adaptive

---

[††]The routing algorithms are studied in Chapter 4.

routing algorithm, and the Double-Y channel routing algorithm as a fully-adaptive routing algorithm for a 2D-mesh network. The look-ahead scheme is also tested for the West-First and the Double-Y channel routing algorithms.

Finally, **chapter 7** concludes this thesis and discusses possible future study.

2. NETWORK TOPOLOGIES

The efficiency of network communication is very critical to the performance of a parallel computer. There are two classifications of interconnection networks : *indirect networks* for dynamic connections and *direct networks* for static connections. Both of the networks are used for distributed networking of multicomputer nodes, or for internal connections among processors, memory modules, and I/O disk arrays in a centralized system. Indirect networks, such as Multistage Interconnection Networks(MIN), connect processors and memories through multiple intermediate stages of switching elements. Direct networks have fixed, point-to-point direct connections, allowing for direct communication between processors. Generally, direct networks are used in message-passing multicomputers while indirect networks are used in shared-memory multiprocessors. Direct Networks have become a popular interconnection architecture for constructing large-scale multiprocessors because they provide better scalability. They have been employed in many recent commercial or proposed machines. These include the Thinking Machines CM2 [9], Intel iPSC and Paragon, Cosmic Cube [10], MIT Alewife [11], Tera supercomputer [12], CMU-Intel iWarp [13], and Stanford DASH multiprocessor [14]. Because of the great potential of direct networks, the focus of this thesis is on direct networks.

Network topology defines how the nodes in a parallel computer are interconnected or, in other words, a connection method. The rest of this chapter surveys and compares typical topologies in direct networks.

2.1. NETWORK PARAMETERS

In a network topology, there are several physical parameters which characterize it. Before surveying network topologies, defining some of these important parameters is necessary for analyzing and understanding of interconnection networks.

Node Degree : The node degree (d) indicates the number of edges (links or channels) incident on a node. It also describes the number of I/O ports required per node, and thus determines the cost of a node. Therefore, in order to reduce cost, the smallest node degree possible is desirable. A constant node degree is also very desirable to achieve modular building blocks for scalable systems and to assure a symmetrical network*.

Network Diameter : The diameter (D) of a network is the shortest path between the most distant nodes in a network measured in links traversed. From a communication point of view, the network diameter should be as small as possible because it is directly related to network latency†. However, after the invention of wormhole routing, it became a less critical issue because a high degree of pipelining causes the communication latency between any two nodes to be almost a small constant‡.

---

*A symmetrical network is a network which looks the same when looking out from any node. Symmetrical networks tend to be easier to implement and program.

†The network latency is the time for a message to travel from the source to the destination.

‡Refer to chapter 3 for more information of the wormhole routing.

Bisection Width The channel bisection width (b) is the minimum number of links along the cut when a network is bisected into two equal halves. The bisection width is a good indicator of the maximum communication bandwidth along the bisection of a network. All other cross sections are bounded by the bisection width.

2.2. DIRECT CONNECTION NETWORK TOPOLOGIES

2.2.1. Linear Array

Using the preceding definitions, let us look first at the simplist of the direct networks, the linear array. It is a 1-dimensional network. The nodes are connected in a simple line as shown in Figure 2.1. A linear array with N nodes has a node degree of 2 for internal nodes or 1 for terminal nodes. The diameter would be $N-1$ and the bisection width would be 1. The linear array is very simple and economically implemented, but the structure is not symmetric and communication becomes more inefficient as the number of nodes (N) increases.

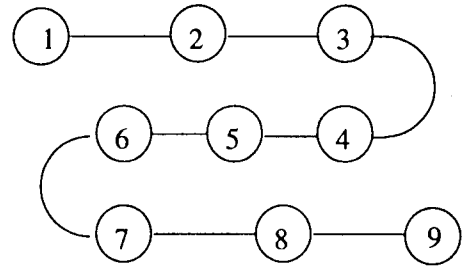


Figure 2.1. Linear array

2.2.2. Ring

A ring is simply a linear array which has the two ends tied together. When compared with the linear arrays, the ring's main advantages are its symmetry, its constant node degree of 2, and its small diameter of $\lfloor N/2 \rfloor$. However, the ring is also limited to a small number of computers because the average message latency and message traffic density increase linearly with N. Unless redundant or overlapping ring paths are used, rings are unreliable because a failure of a single node or path in the ring can stop most communication. The IBM token ring has this topology. In the IBM, messages circulate along the ring until they reach the destination with a matching token. The CDC Cyberplus multiprocessor(1985) and KSR-1 computer system(1992) adapted the packet-switched ring.

Rings become chordal rings by adding some more channels to each node. Chordal rings decrease the network diameter. However, their node degrees increase as the diameters decrease. The Figure 2.2 b and Figure 2.2 d show the chordal rings with node degrees of 3 and 4, respectively. The most complicated ring model is the completely connected network, shown in Figure 2.2 d. Its diameter is always 1, but the node degree is $N - 1$.

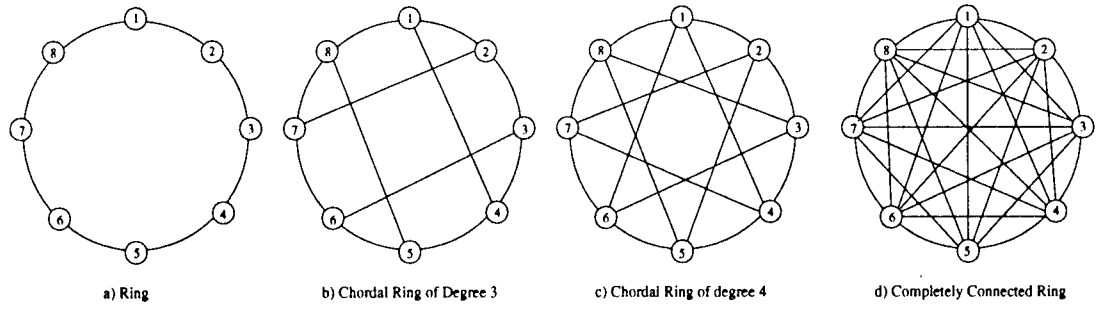


Figure 2.2. Ring

2.2.3. Star

Star networks are quite common, primarily because control of the network from a central computer is relatively simple for small N models . Any node can communicate with any other nodes via the central or hub node. However, for large N models, the central node becomes a bottleneck, thereby severely limiting the performance of the network. On the other hand, one of the most significant features of a star network is its ability to centralize much of the intelligence required to control the network into one place, the hub. For this reason, the star network is of particular value when linking it through a simple terminal connection to cheap devices that have little internal intelligence. The star network is, however, vulnerable to failures of the hub node. A single failure at the hub can disable the whole network. It is not practical to have redundant hub nodes, since the hub is by far the most expensive node in the network.

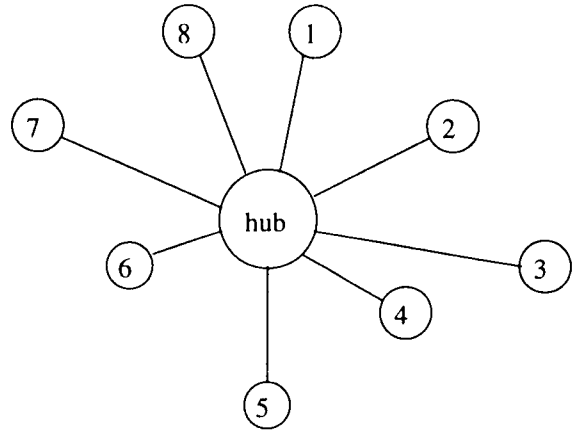


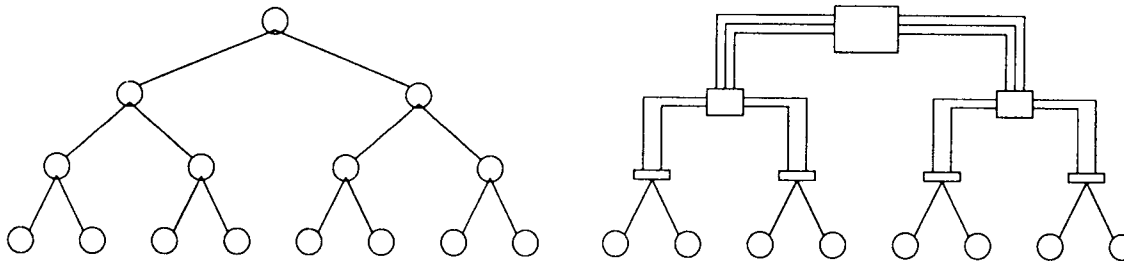
Figure 2.3. Star

2.2.4. Tree and Fat tree

Figure 2.4 shows examples of the binary tree and the fat tree with 3-level. In a k-level completely balanced binary tree, there are $N = 2^k - 1$ nodes. The maximum node degree is 3. The leaves and the root have the node degree of 1 and 2, respectively. The diameter of a binary tree is $2 * (k - 1)$ because a binary tree is divided into two branches at the root node. Since the binary tree has a low maximum node degree, this architecture is easily scalable.

There are two major shortcomings of the binary tree. The first is a long network diameter. The second and the most serious problem is the bottleneck problem at the root node. This is caused by the messages which have their destinations in the opposite half tree and which must pass through the root node.

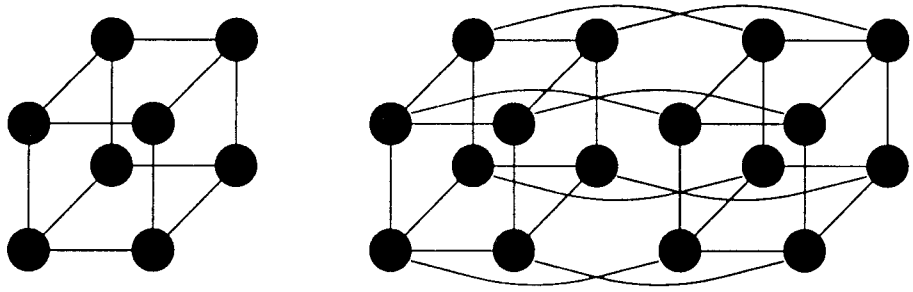
The fat tree, introduced in [18], alleviated this bottleneck problem by progressively increasing the bandwidth of the channels up to the root. An example of a production machine which uses a fat tree, as seen in Figure 2.4 b, is the Connection Machine CM-5.



a) Binary Tree

b) Fat Tree

Figure 2.4. Binary tree and fat tree



a) 3-cube b) 4-cube

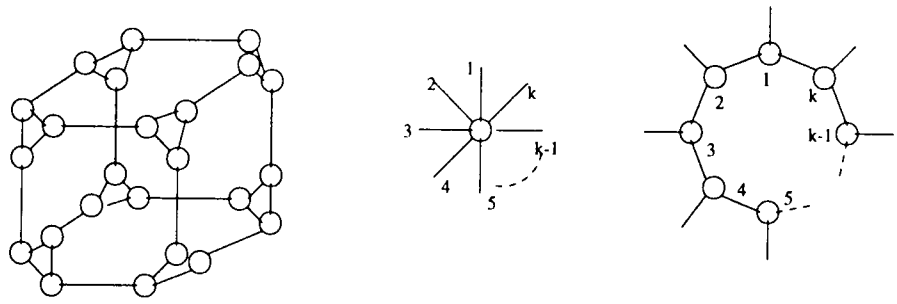
Figure 2.5. Hypercubes

2.2.5. Hypercubes

A binary hypercube was a very attractive topology for research and development in the 1980's due to its generality and flexibility. It delivers the lowest communication latency and the least dense traffic per link of any k-ary n-cube configuration[§]. However, the connection density in a node increases linearly with the dimension as does the node degree. Moreover, the number of nodes is increased by a power of 2 with the dimension. Therefore, the hypercube architecture lacks scalability and is difficult to implement. In spite of possessing many good characteristics, the hypercube network is gradually loosing its popularity. Hypercubes are being replaced by other architectures, such as a low dimensional mesh, which are more scalable and easier to implement. It reveals that the network architecture of the future must have good packaging efficiency and scalability to allow for modular growth.

[§]See page 16.

An n-cube has $N = 2^n$ nodes and its node degree is n, where n is dimension. Figure 2.5 shows a 3-cube with 8 nodes and a 4-cube with 16 nodes. The iPSC, nCUBE, and CM-2 systems adapted the binary n-cube architecture.



a) 3-Cube-Connected Cycles b) A corner of k-cube c) A cycle of k-CCC

Figure 2.6. Cube-Connected Cycles (CCC)

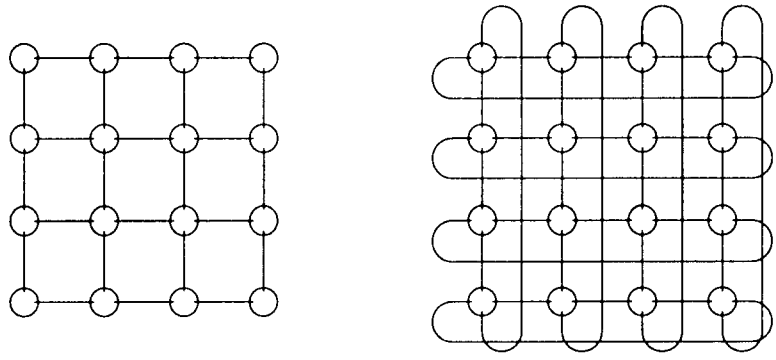
2.2.6. Cube-Connected Cycles (CCC)

The cube-connected cycle architecture [15] is a modification of the hypercube topology which makes it suitable for very large networks. Fig 2.6 illustrates a cube-connected cycle network (3-CCC). As can be seen in Figure 2.6, a 3-CCC topology is made from cutting off the corner nodes (vertices) of the 3-cube and replacing each corner cut with a triangle (cycle) of 3 nodes.

This architecture realizes a constant node degree of 3 with the merits of hypercubes. A k-CCC topology built from a k-cube with $N = 2^k$ nodes has the constant node degree of 3 and is independent of the dimension. Therefore, it overcomes the packing difficulty of hypercube.

A CCC topology with N nodes, where $N = 2^n$, must be built from k-cube such that $2^n = k{\cdot}2^k$ for some $k < n$. For example, consider a 64-node CCC network. $64 = 2^6 = 4 * 2^4$, therefore, $n = 6$ and $k = 4$. The 64-node CCC topology is formed

from the 4-cube by replacing the corner nodes with cycles of 4 nodes. The diameter of the CCC is $2k$ and is longer than that of the hypercube with the same number of nodes.



a) A mesh in 2 dimension

b) A torus in 2 dimension

Figure 2.7. Mesh and torus

2.2.7. Mesh and Torus

The mesh network, shown in Figure 2.7, is very simple and easy to implement. The pure meshes are asymmetric because the node degrees at boundary and corner nodes are 3 and 2, respectively. A k-dimensional mesh, which has $N = n^k$ nodes, has the node degree of $2k$ and the diameter of $k(n-1)$. This topology has many more shortcomings than hypercube. However, its simplicity and scalability make this architecture very popular for commercial parallel machines. Now, the majority of network topologies for wormhole routing in multicomputers are meshes. Most of these meshes are low-dimensional meshes. Low-dimensional meshes are generally preferable because they have low and fixed node degrees which make them more scalable than high-dimensional meshes [25]. A 2-D mesh is employed in the Intel

Touchstone DELTA [4], the Intel Paragon, and the Ametek 2010 [5]. A 3-D mesh is used in the J-machine at MIT [19].

A torus architecture is a modification of a mesh. In a torus, wrap-around connections are used along each row and column of the array. This increases the efficiency of the network. A 2-dimension torus is symmetric and has a constant node degree of 4 and a diameter of $2\lfloor n/2 \rfloor$.

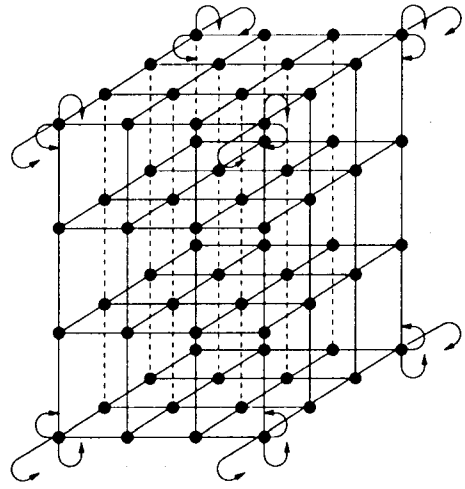


Figure 2.8. The k-ary n-cube

2.2.8. k-ary n-cube

The k-ary n-cube topology has a number of interesting properties [17]. The most commonly used direct networks discussed thus far are variants of the k-ary n-cube [16]. The parameters n and k represent the dimension of the cube and the number of nodes along each dimension, respectively. The number of nodes N can be calculated using

$$N = k^n, \qquad (k = \sqrt[n]{N}, n = \log_k N)$$

Network type	Node degree	Network diameter	No of links	Bisection width	Symmetry	Remarks on Network size
Linear Array	2	$N-1$	$N-1$	1	No	N nodes
Ring	2	$\lfloor N/2 \rfloor$	N	2	Yes	N nodes
Completely Connected	$N-1$	1	$N(N-1)/2$	$(N-1)^2$	Yes	N nodes
Binary Tree	3	$2(h-1)$	$N-1$	1	No	Tree height $h = \lceil \log_2 N \rceil$
Star	$N-1$	2	$N-1$	$\lfloor N/2 \rfloor$	No	N nodes
2D-Mesh	4	$2(r-1)$	$2N-2r$	r	No	$r \times r$ mesh where $r = \sqrt{N}$
2D-Torus	4	$2\lfloor r/2 \rfloor$	$2N$	$2r$	No	$r \times r$ torus where $r = \sqrt{N}$
Hypercube	n	n	$nN/2$	$N/2$	Yes	N nodes, $n = \log_2 N$ (dimension)
CCC	3	$2k-1+\lfloor k/2 \rfloor$	$3N/2$	$N/(2k)$	Yes	$N = k2^k$ nodes with a cycle length $k \geq 3$
k-ary n-cube	$2n$	$2n\lfloor k/2 \rfloor$	nN	$2k^{n-1}$	Yes	$n = k^n$ nodes

Table 2.1. Summary of static network characteristics

Examples of a k-ary n-cube include the ring (n=1), 2-D mesh or torus (n=2), 3-D mesh or torus (n=3), and hypercube (k=2). The generality and flexibility of the k-ary n-cube make it an excellent choice for exploring various network design trade-offs.

2.2.9. Summary

Table 2.1 [2] summarizes the important characteristics of static networks discussed in this chapter. Of the characteristics listed, the number of links has

the greatest effect on the network cost. A small number of links is very desirable to reduce the cost of constructing a network. In the network, the bisection width affects the bandwidth. Also, the property of symmetry affects scalability and routing efficiency.

A constant node degree of 4 or less is used for most networks. These values are also desirable in order for a node to function as a building block. The completely-connected and star networks have high node degree. Hypercubes also suffer from high node degree as the number of nodes becomes large.

Network diameter directly affects network latency in store-and-forward routing**. However, the invention of wormhole routing with the pipelining nature made the effect of the network diameter on the latency negligible. The average diameter in a network may be a better measurement to characterize a network.

Based on the information in Table 2.1, mesh, torus, k-ary n-cube, and cube-connected cycles (CCC) are the most suitable topology for massively parallel processing (MPP).

**Chapter 3 deals with store-and-forward and wormhole routing.

3. SWITCHING TECHNIQUES

One of the basic considerations in the design of a computer communication network is the switching method. A switching method determines how the data in an input channel is removed and placed on an output channel. The communication latency greatly depends on the switching technique chosen.

Four switching techniques have been employed in direct networks: *circuit switching, store-and-forward, virtual cut-though,* and *wormhole routing.* Among these four, the wormhole routing has become the most popular due to its low communication latency and small hardware requirements.

In this chapter, we first explain the message format used in switching techniques and then survey the switching schemes currently in use.

3.1. THE MESSAGE FORMAT

A message is the logical information unit used to send and receive information between nodes. The message length may vary depending on the amount of information that has to be sent. A message is divided into an arbitrary number of fixed length packets since routing hardware can handle only a fixed length of data. Thus, a packet is a basic unit which has all the necessary information, such as the destination address, needed for routing. Since the packets may not arrive at their destination in the order they were sent if an adaptive routing is used, packets also have a sequence number to reassemble the original message. The Figure 3.1 illustrates the basic concept of the message format.

With the store-and-forward routing, a packet is the smallest unit of information for transmission. A packet can be divided into smaller units called *flits* (flow

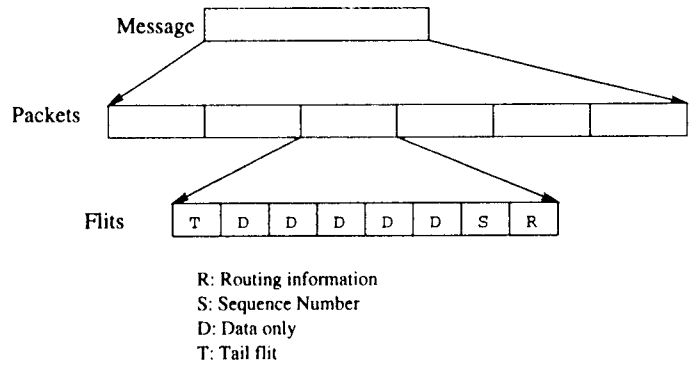


Figure 3.1. The message format

control digits) which are the smallest units for the wormhole routing. Among the flits in a packet, the header flit must have the routing information or the destination address, and the last flit (tail) in a packet should have a special mark for a router to identify the end of a packet.

Figure 3.2 shows the typical message of the J-Machine network* [19]. The first three flits of the packet contain the X-, Y-, and Z-axis addresses for the destination. The final flit is marked as the "tail".

3.2. CIRCUIT SWITCHING

Of the switching schemes used in the computer communication networks, the circuit switching scheme was the first devised. It was originally designed for human communication in telephone networks. In this method, a complete communication circuit (communication path) between the source and the destination nodes is set

---

*J-machine network is 3-D mesh and uses *e*-cube routing and two virtual channels per a physical channel. The *e*-cube routing and virtual channel are explained in chapter 4.

Flit No	Contents	Remark
1	5: +	X address
2	1: -	Y address
3	4: +	Z address
4	MSG:00	Method
5	00440	to call
6	INT:00	Argument
7	00023	to method
8	INT:00	Reply
9	<1:5:2> T	address

Figure 3.2. A typical message format in the J-Machine

up before the real communication begins. If a communication path can not be constructed because the desired channel is being used by other messages, then the message is blocked. Once a circuit is set up, no more signaling for addressing is necessary and the packet is transmitted along the circuit to the destination. During the transmission period, the channels are reserved exclusively for the message. Thus, intermediate nodes need not have buffers. After an entire message has been transmitted along the path to the destination, the circuit is released.

The circuit switching has some drawbacks. First, it has a very long set-up time which delays the transmission of messages. The original purpose of circuit switching was for telephone networks and a long set up time is not a problem in telephone networks because a typical conversation message is much longer than the set-up time. However, computer networks typically use frequent short messages with occasional long ones. Because the circuit for a message path is set up for every message, the excessive set-up delay is a serious disadvantage. The second drawback

to the circuit switching is its low capacity for channel utilization (channel sharing). Because all the channels in a circuit are reserved by a single message exclusively during message transmission, the channels can not be used by others during that period.

3.3. STORE-AND-FORWARD (SAF) ROUTING

The store-and-forward routing can achieve better channel utilization than the circuit switching. In the store-and-forward routing, an entire packet is transmitted from a source node to a destination node through the intermediate nodes. Each node has the packet buffer to store an entire packet. When a packet reaches an intermediate node, it is first stored in the packet buffer. Then, the packet is forwarded to the next node if the desired output channel and the packet buffer in the receiving node are both available [2].

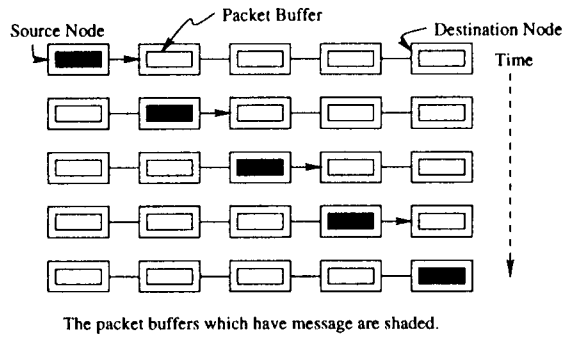


Figure 3.3. The store-and-forward routing scheme

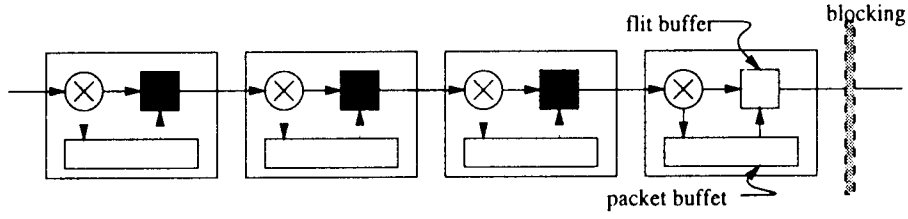
As shown in Figure 3.3, a packet message is forwarded to the next node after it is stored in an intermediate node. Therefore, the latency in SAF depends on the distance between the source and the destination nodes. The store-and-forward scheme, also called the packet switching, was used in the first generation multicom-

puters, such as iPSC-1, Ncube 1, Ametek 14, the research prototype Cosmic Cube and FPS T-series [21].

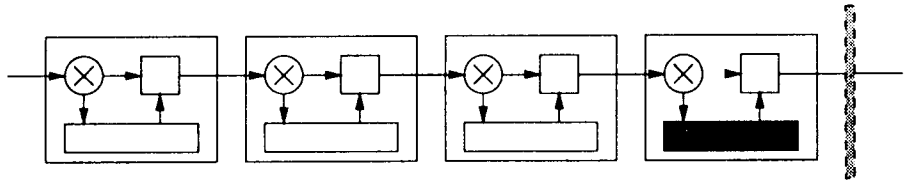
3.4. VIRTUAL CUT-THROUGH

The virtual cut-through was introduced by Kermani and Kleinrock in 1979 [22]. It transmits a packet at the flit level. In the virtual cut-through, an entire packet is stored at an intermediate node *only* when the next required channel is not available. If the next channel is not busy, it is forwarded immediately at the flit level without buffering. Thus, the virtual cut-through technique saves lots of time spent transmitting data compared to the store-and-forward routing.

This technique requires enough buffers to temporarily store blocked packets. Figure 3.4 shows an example of the virtual cut-through routing.



a) Before blocked, a packet is spread over 3 routers.



b) When blocked, a packet is stored in a packet buffer.

Figure 3.4. The virtual cut-through routing scheme

3.5. WORMHOLE ROUTING

The wormhole routing was introduced in [20]. The wormhole routing applies a cut-through approach to switching and reduces the tremendous hardware requirements of the virtual cut-through routing. In the wormhole routing, the basic unit of information flow is a flit instead of a packet as in SAF. The message transmission is done on flit-by-flit basis from the source node to the destination node via intermediate nodes. Since only the header flit has the destination information, all the flits in the same packet must be inseparable to prevent any mix up when different packets are interleaved during transmission. The relationship of the engine car to the box cars in a train is a good analogy for a packet in the wormhole routing scheme[†].

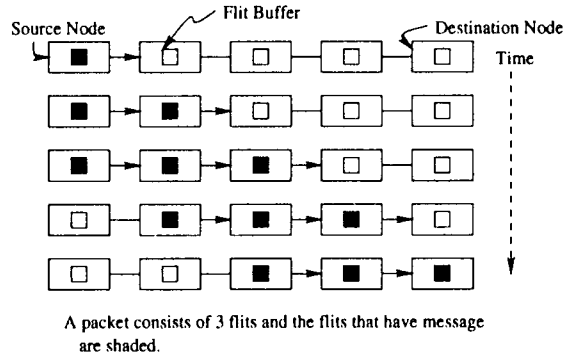


Figure 3.5. The wormhole routing scheme

As shown in Figure 3.5, the wormhole routing needs only small flit buffers in each node. The latency is almost independent of the distance between the source and the destination due to the pipelined nature of the wormhole routing[‡]. Unlike

---

[†]See the page 52.

[‡]It is proved in next section.

the virtual cut-through technique, blocked messages in the wormhole routing are spread over several intermediate nodes and remain in the network [23].

The wormhole routing uses a handshaking protocol asynchronously to pass flits. As shown in Figure 3.6, along the path, a 1-bit request/acknowledge (R/A) line associates with a data channel to handshake between neighbor nodes. The receiving router B requests the sending router A to send a flit by pulling the input side R/A line low when its buffer is available (Figure 3.6 a). Then, if the router A is ready to send, it raises the output side R/A line high for acknowledgement and sends the flit through the data channel (Figure 3.6 b). The R/A line is kept high while an

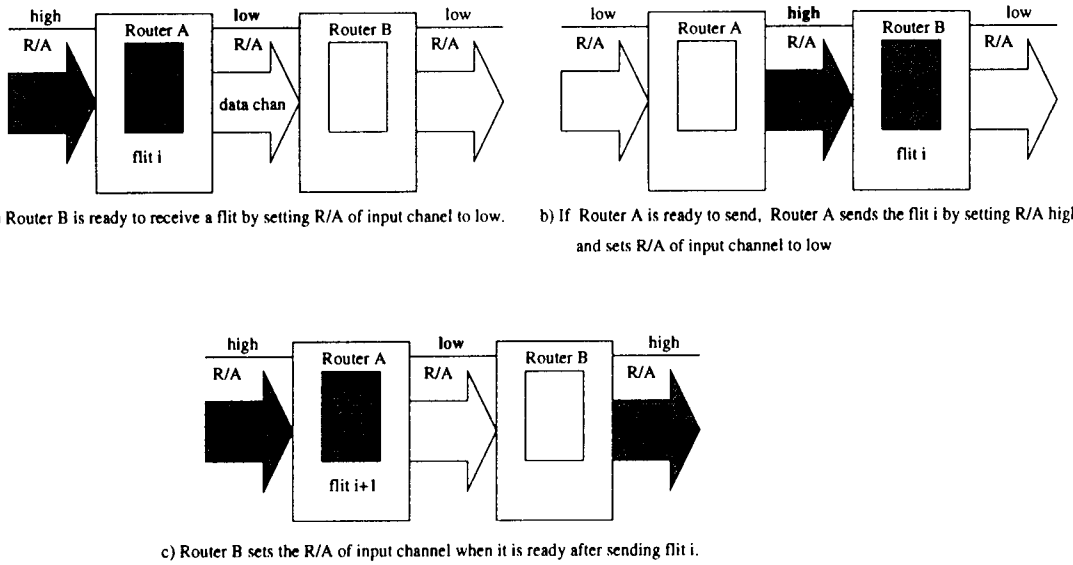


Figure 3.6. Handshaking protocol in the wormhole routing

entire flit is being transmitted by the sending router A and received by the receiving router B. The R/A line is lowered when the router B is ready to receive after sending the flit (Figure 3.6 c). This cycle is repeated for the next flits. This asynchronous handshaking protocol can be very efficient and faster than synchronous systems.

The wormhole routing has two big advantages over the SAF routing. First, each router requires just enough buffer space to store a flit for each channel, while the SAF routing requires enough buffer space to store an entire packet for each channel. Second, the latency of the wormhole routing is much lower than that of SAF routing due to its pipelined nature. For those reasons, the wormhole routing is the most promising switching technique for new generation multicomputers. Figure 3.7 shows an example of the wormhole routers in a mesh network [2].

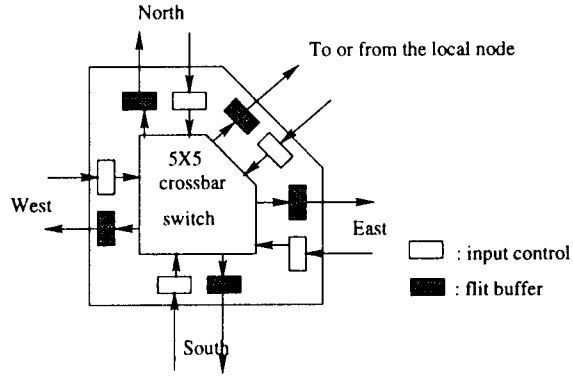


Figure 3.7. The structure of a wormhole router in a mesh network

3.6. PERFORMANCE COMPARISON OF SWITCHING TECHNIQUES

The network latency for the circuit switching T_{CS} is given by

$$T_{CS} = (\frac{L_c}{W})D + \frac{L}{W} \tag{3.1}$$

where L_c is the control packet length (in bits) transmitted to establish the circuit, W the channel bandwidth (in bits/sec), D the distance (number of channel traversed) and L the packet length.

The network latency of the SAF T_{SF} is

$$T_{SF} = (\frac{L}{W})D \tag{3.2}$$

The network latency for the virtual cut-through T_{VCT} is

$$T_{VCT} = (\frac{L_h}{W})D + \frac{L}{W} \tag{3.3}$$

where L_h is the header field length.

The latency for the wormhole routing T_{WH} is

$$T_{WH} = (\frac{F}{W})D + \frac{L}{W} \tag{3.4}$$

where F is the flit length (in bits).

Equation 3.2 shows that T_{SF} is directly proportional to the distance D. Equation 3.4 implies that T_{WH} is almost independent of D because $T_{WH} \approx L/W$ when $L >> F$. Likewise, Equation 3.3 and 3.1 can approximate to L/W. The Figure 3.8 compares the behaviors of the various switching schemes.

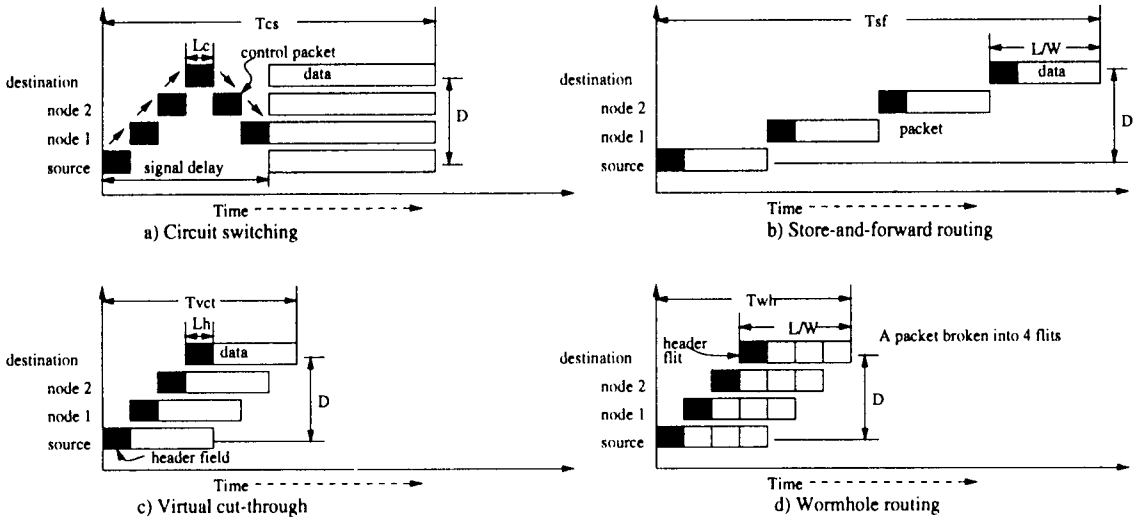


Figure 3.8. The time comparison of switching methods

4. ROUTING

One of the most critical aspects to the performance of direct networks is *routing*. Routing determines how a message chooses the path to its destination in a network. Routing algorithms for a given network topology must have the following three characteristics: low communication latency, high network throughput, and ease of implementation [24].

There are several ways to classify routing schemes. One way to classify is based on the decision maker of the path.

Source routing: In *source routing*, the entire path is selected by the source node before the message is sent. Each packet must carry the routing information determined by the source node, which increases the packet size. Once the path is chosen and the packet leaves the source node, the path cannot be changed.

Distributed routing: In *distributed routing*, the router decides the channel through which the packet proceeds to the neighbor router or to the local processor. Since the router decision is limited to only its neighboring routers or local processor, the knowledge of the global state of the network is not required. The routing decision in a router should be made as fast as possible to reduce the network latency. Most direct networks use this method for routing.

Another classification is by the flexibility of the path between source and destination.

Deterministic: If the path is completely determined by the source and destination address and there is no alternate path, the routing is called *deterministic*. In other words, the routing path is predetermined in advance, independent of network conditions. This routing is usually very simple and requires only a

small amount of hardware. The XY routing algorithm in 2-D mesh and the E-cube routing in hypercube are examples of the deterministic algorithm.

Adaptive: A routing technique is referred to as *adaptive* if packets have the ability to use alternative paths toward their destinations depending on traffic congestion or fault in the network. An adaptive routing may be *fully-* or *partially-adaptive*. In MPP, the adaptivity is a desirable feature to have in order to improve the channel utilization and to detour around faulty nodes. However, the ability to make a path decision in a router may slow down and the hardware requirements may increase as the adaptivity increases due to the complexity of the routing algorithm.

A routing algorithm also can be distinguished as *minimal* or *nonminimal*.

Minimal: A routing is called *minimal* if the routing restricts its choice to the shortest paths between the source and the destination.

Nonminimal: The *nonminimal routing* allows packets to choose the longer paths, not just the shortest paths. The nonminimal routing provides better fault tolerance and better adaptivity. But, if the nonminimal routing is used, the problem of livelock* must be considered.

4.1. DEADLOCK

There may be situations in which a message will never arrive at the destination. Routing algorithms must avoid those situations. Therefore, routing must be free from the following conditions: *deadlock, indefinite postponement,* and *livelock.*

---

*Refer to page 30

Deadlock: The deadlock occurs when a set of message packets is blocked indefinitely and cannot proceed. The deadlock situation is explained in more detail later.

Indefinite postponement: The indefinite postponement occurs when a message packet waits for a network resource while other packets with higher priority continually access the resource. Thus, the waiting packet never gets the resource. It differs from deadlock in that it does not cause the circular wait condition.

Livelock: When a message does not reach its destination and goes around forever in the network without deadlock, this situation is called livelock. The livelock may occur only in adaptive routing algorithms. However, it can be avoided by using *minimal* routing.

Among the three cases, the deadlock is the most serious problem. The indefinite postponement and the livelock are less likely to occur and are easier to resolve. For this reason, only the deadlock situation is considered in this thesis.

4.1.1. What is deadlock?

Deadlock happens when packets hold one resource while requesting another resource, creating a circular wait condition. In the store-and-forward and the virtual cut-through routing, the resources are buffers. On the other hand, in the circuit switching and the wormhole routing, the resources are channels. Figure 4.1 illustrates the deadlock on buffers in the store-and-forward routing. Figure 4.2 shows an example of the deadlock on channels in the wormhole routing with four routers and four packets waiting for the channels [21]. The deadlock can also be explained by the channel dependency graph [23], illustrated in Figure 4.3. In the wormhole

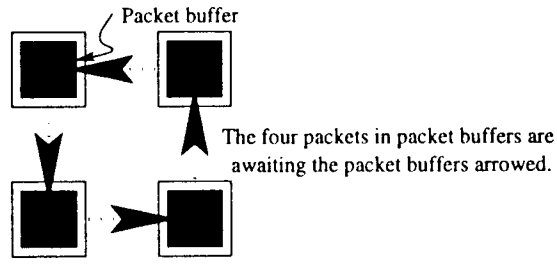


Figure 4.1. The deadlock on buffers in four nodes with store-and-forward routing. Four packets make a wait loop so that no one can proceed.

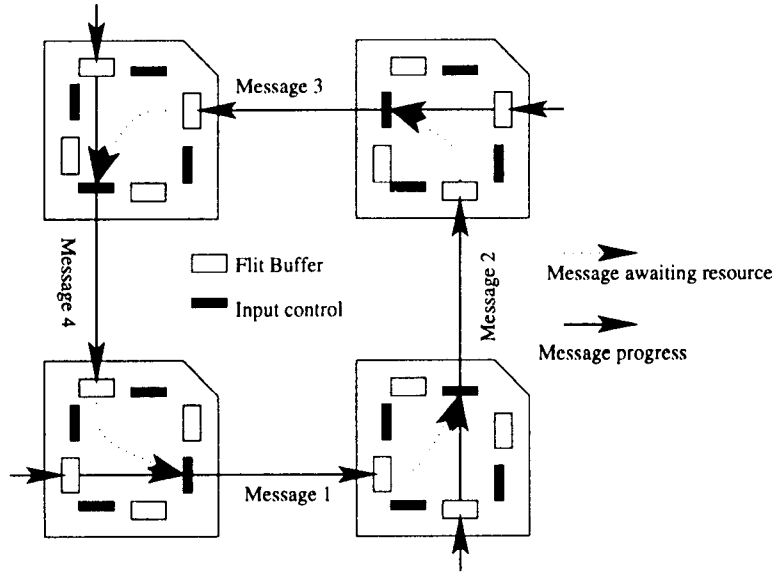


Figure 4.2. The deadlock on channels in four nodes with wormhole routing. Four packets make a wait loop so that no one can proceed.

routing, the decision to choose the next channel for a packet is made by a routing function. This function, or routing algorithm, is based on the channel that a packet currently occupies and on the position of the destination node [23]. The progress of a packet depends on the condition of the next channel chosen by the routing function. In Figure 4.3 a, the vertices of the graph represent the set of processing nodes and the edges represent the set of communication channels. The channels are occupied by 4 different packets. For example, the next channel for the packet at the channel c_3 is c_2. The packet at the channel c_3 must wait until the channel c_2 is available, because the progress to the next channel depends on the condition of the next channel. Therefore, a dependency graph, Figure 4.3 b, can be drawn from Figure 4.3 a. A deadlock situation arises if a cycle exists in the channel dependency graph. All packets at channels c_3, c_2, c_1, and c_0 are waiting for each other to move. In this situation, no channel will ever be vacated.

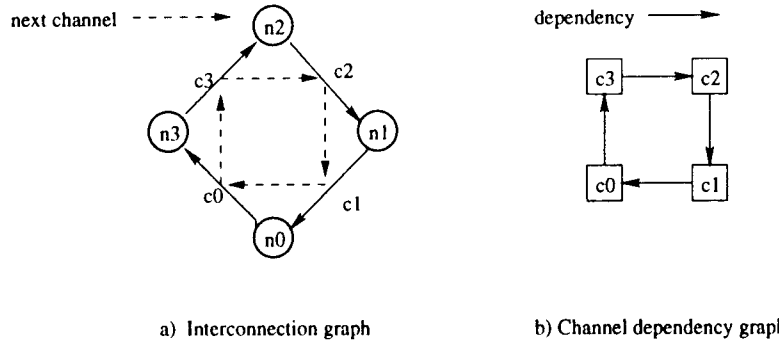


Figure 4.3. Deadlock situation in channel dependency graph.

4.1.2. Deadlock avoidance

Deadlock situations must be avoided, therefore many deadlock avoidance strategies have been presented in the literature. One typical approach, called struc-

tured buffer pool, is based on the buffer pool reservation. Many deadlock free algorithms for the store-and-forward routing have been proposed using this approach [26–29]. In the structured buffer pool method, the message buffers in each node of the network are partitioned into several classes, and only a restricted set of packets is permitted to access a given buffer class. This approach has some advantages. It is "local" in nature and thus can be implemented very easily without the need for complex global synchronization mechanisms. In addition, it does not require the exchange of special control information packets. It requires only the usual control packets which contain positive acknowledge, negative acknowledge, etc.., that are usually exchanged during data transmission [29]. However, the problem with the structured buffer pool is that it needs a certain minimum number of buffers in a node. For example, when the structured buffer pool scheme is based on the number of hops [28,30], where H is the maximum number of hops that any packet can travel in the network, then, to guarantee freedom from deadlock, the nodes should have at least $H + 1$ buffers. Figure 4.4 [31] shows an example of a buffer graph constructed with the structured buffer pool method so that there is no cycle.

Another approach used to avoid deadlock is to prohibit packets from being sent from some specified channels to other specified channels so as to avoid creating dependency cycles. This solution is accomplished without adding more channels or more buffers. Figure 4.5 gives an example of this approach. By restricting a message forwarding from channel $c0$ to $c1$ and from $c6$ to $c7$, freedom from deadlock is accomplished. The disadvantage of this approach is that some resources that could have been used are prohibited, therefore resource contention becomes serious. As illustrated in Figure 4.5, to send a packet from node 0 to node 2, the path through node 3 must be used instead of node 1 because the path through node 1 is

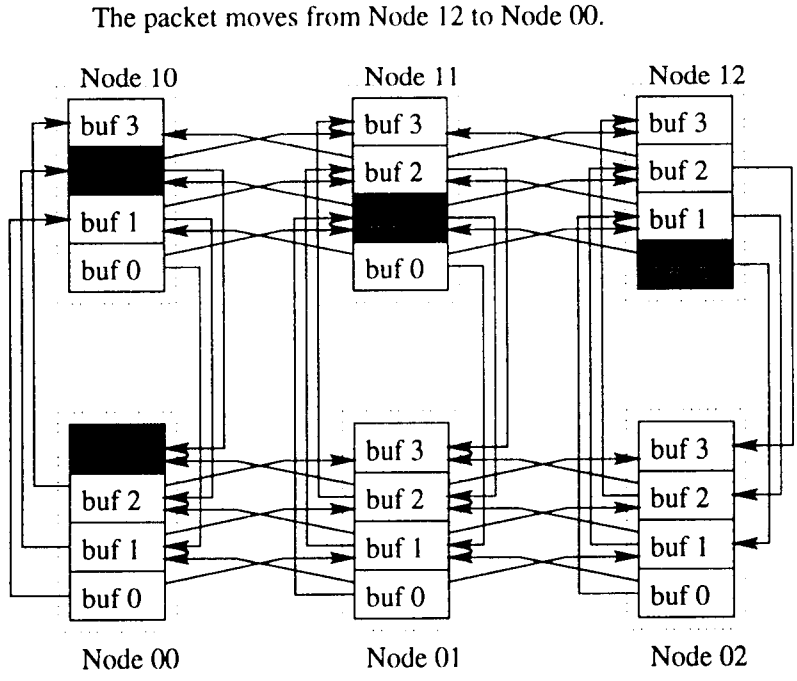


Figure 4.4. Structured buffer pool method : There is no cycle of buffer graph.

prohibited. This means that a resource, the path through node 1, cannot be used and thus the contention to the path through node 3 becomes heavier.

Another approach is to use virtual channels[†]. Dally and Seitz introduced this approach in 1987 [23] to avoid deadlock. Figure 4.6 [23] shows an example of how to avoid deadlock by adding virtual channels. The example network in Figure 4.6 a has 4 nodes, $N = \{n_0, \ldots, n_3\}$ and 4 channels, $C = \{c_0, \ldots, c_3\}$. The dependency graph shows that it has a deadlock situation. As shown in Figure 4.6 b, each channel is split into high channels, $c_{00}, \ldots, c_{03}$, and low channels, $c_{10}, \ldots, c_{13}$. Messages at a node numbered less than their destination node are routed on the high channels,

[†]Virtual channels are discussed in detail in next section.

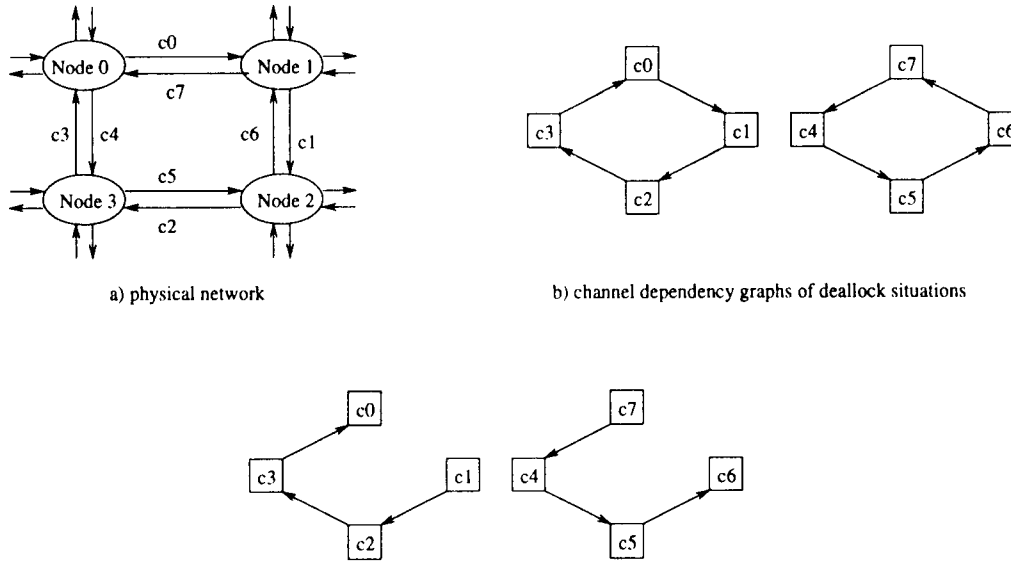


Figure 4.5. Prohibiting transmission of packets from channel $c0$ to $c1$ and from $c6$ to $c7$ prevents deadlock.

and messages at a node numbered greater than their destination node are routed on the low channels [23]. Therefore, Channels c_{00} and c_{13} are never used. This routing algorithm routes every packet along channels with strictly decreasing numbers, i.e. $c_{12} > c_{11} > c_{10} > c_{03} > c_{02} > c_{01}$. Therefore, adding virtual channels changes the cycle to a spiral and therefore breaks the channels dependence cycle, thus avoiding a deadlock (Figure 4.6).

4.2. VIRTUAL CHANNELS

A virtual channel is a logical channel with its own flit buffer, control line, and shared data path between two neighboring nodes. As shown in Figure 4.7, flit buffers in a transmitting node and a receiving node time-share a physical channel. Buffers in the transmitting node are paired with buffers in the receiving node, re-

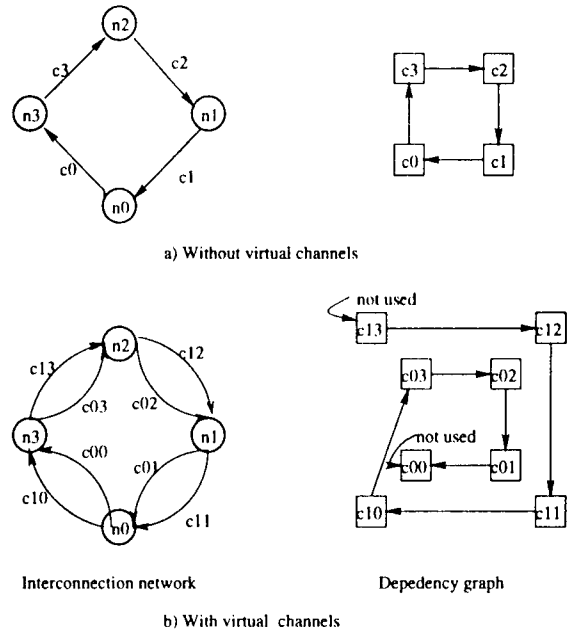


Figure 4.6. Deadlock avoidance with adding virtual channels

spectively. Each node of an interconnection network should have crossbar switches, a multiplexer, and a demultiplexer to multiplex and arbitrate a physical channel among many virtual channels. Conceptually, through virtual channels, a physical network can be divided into multiple disjointed logical networks.

The original purpose of virtual channels was to avoid deadlock. Virtual channels also provide full adaptivity, better channel utilization, and better throughput. The most costly resource in an interconnection network is the physical channels (wires). The virtual channels make full adaptivity economically feasible because they share a single physical channel. On the other hand, another fully-adaptive

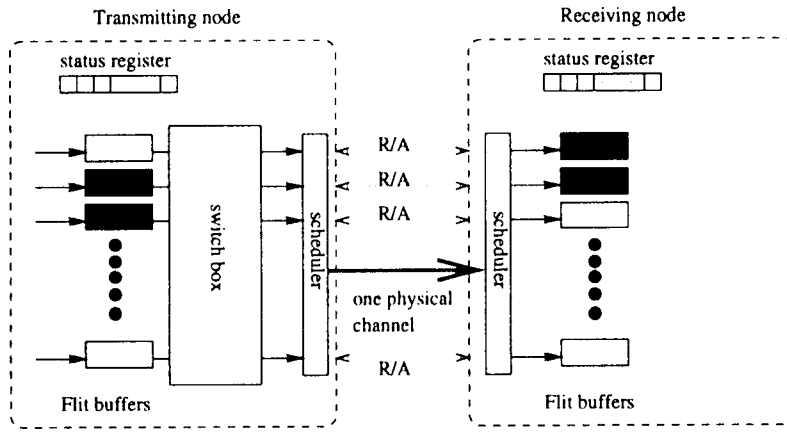


Figure 4.7. Virtual channels

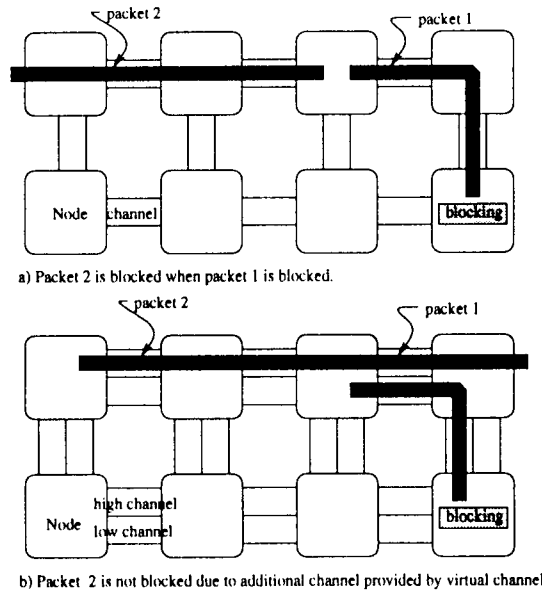


Figure 4.8. Analogy between virtual channels and multi-lane street

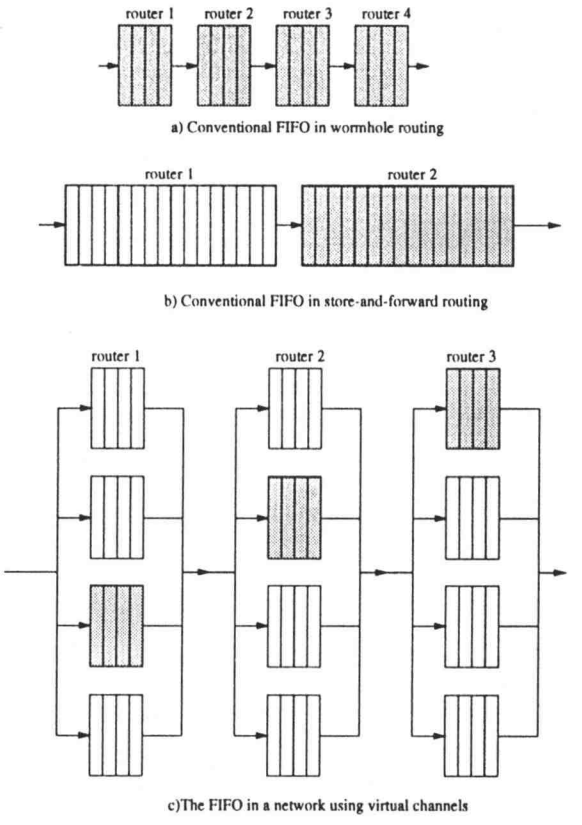


Figure 4.9. FIFO queues in routing algorithms

routing algorithm, the Double-Y channel routing algorithm[‡], is expensive to implement because it needs double channels along the Y-axis.

Adding virtual channels to an interconnection network is like expanding a one-lane street to a multi-lane street. As shown in Figure 4.8, a message can pass the blocked message through the added virtual channel. This aspect increases the channel utilization and thus throughput.

[‡]See page 44.

A node in a conventional network has a first-in, first-out (FIFO) queue buffer and thus can contain only flits from a single packet. If the packet in a FIFO queue is blocked, the physical channel is idle because no other packet can use it. With virtual channel flow control, a FIFO in a conventional network is divided into several independent buffers (lanes) as shown in Figure 4.9. When a packet is blocked, only one of the buffer lanes is idle and other packets can access the channel, thereby increasing the channel utilization.

4.3. ROUTING ALGORITHMS

This section introduces several examples of routing algorithms using wormhole routing.

4.3.1. Deterministic routing algorithms

A simple way to design a deadlock-free routing algorithm is with the *dimension ordered routing* algorithm, which routes each packet in one dimension at a time, arriving at the proper coordinate in each dimension before proceeding to the next dimension [21]. By enforcing a strictly monotonic order on the dimensions traversed, deadlock-free routing is guaranteed. Since this routing method is independent of current network conditions, it is deterministic.

4.3.1.1. X-Y routing algorithm

This routing algorithm is applied in 2-D mesh network. In a 2-D mesh, each node is identified by its coordinate (x, y). Each packet is routed along the X-dimension first and then the Y-dimension. Let (s_x, s_y) and (d_x, d_y) be the source and the destination nodes, respectively. And, let relative address $(g_x, g_y) = (d_x -$

$s_x, d_y - s_y$). The first two flits carry g_x and g_y information. Each router has 4 external input/output channel pairs, labeled as +X, -X, +Y, and -Y, and an internal channel pair for the local processor (See Figure 4.10.). When a packet leaves a local processor, the router of the local processor sets the input channel C=(+X) if $g_x < 0$; otherwise, C=(-X). If g_x=0, it sets the input channel C=(+Y) if $g_y < 0$; otherwise, C=(-Y).

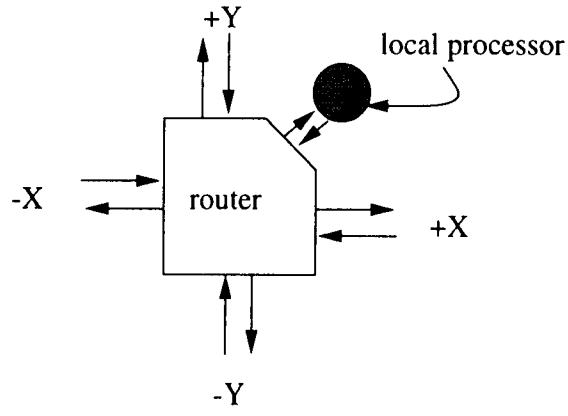


Figure 4.10. A router for the XY routing algorithm

If the input channel C=(+X) or (-X),

- If $g_x = 0$, discard the first flit, g_x.

  1. If $g_y > 0$, forward the packet to the output channel +Y.
  2. If $g_y < 0$, forward the packet to the output channel -Y.
  3. If $g_y = 0$, discard the second flit and forward the packet to the local processor.

- If $g_x > 0, g_x = g_x - 1$ and forward the packet to the output channel +X.

- If $g_x < 0, g_x = g_x + 1$ and forward the packet to the output channel -X.

If the input channel C=(+Y) or (-Y),

- If $g_y > 0$, forward the packet to the output channel +Y.
- If $g_y < 0$, forward the packet to the output channel -Y.
- If $g_y = 0$, discard the second flit and forward the packet to the local processor.

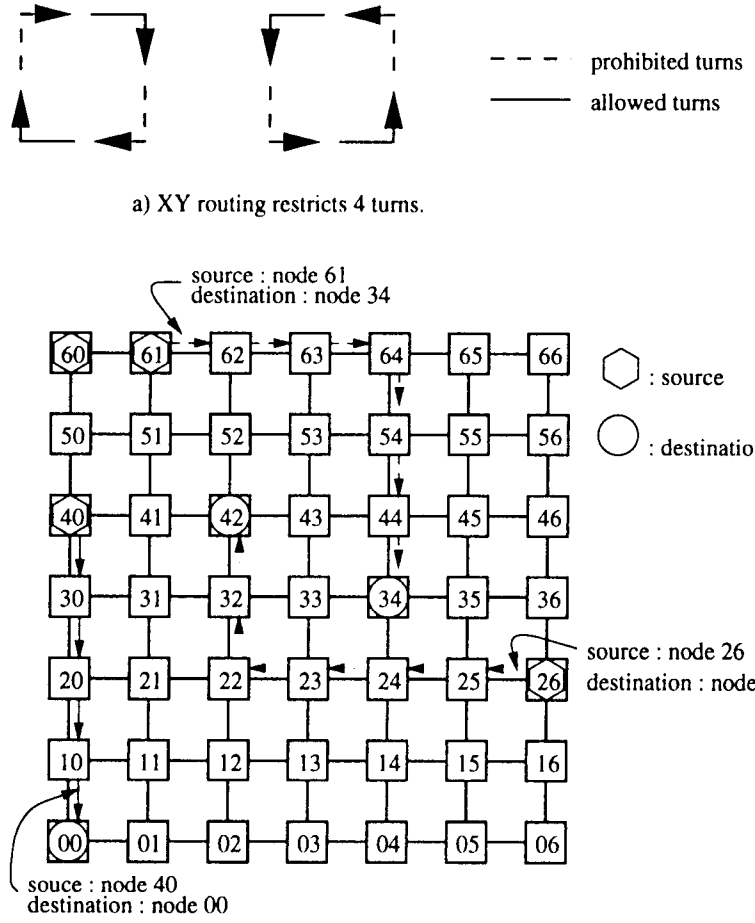


Figure 4.11. The XY routing algorithm on 2-D mesh

Simply said, the XY routing algorithm restricts some turns, thus avoiding deadlock situations, as illustrated in Figure 4.11 a. Figure 4.11 b shows examples of the XY

routing algorithm. If a flit size is 8 bits, the maximum 2D-mesh size is $128 \times 128(2^8 - 1)$. The XY routing algorithm can be easily extended to an n-dimensional mesh.

4.3.1.2. E-cube routing algorithm

The E-cube routing algorithm was proposed for hypercube networks in [32]. The E-cube routing algorithm is also a dimension ordered and deterministic routing algorithm. Each node in a hypercube network has a n-digit binary address, such as $x = x_{n-1}, x_{n-2}, x_{n-3}, \ldots, x_2, x_1, x_0$ where n represents the dimension. The E-cube routing is implemented as follows:

Let the source node $s = s_{n-1}, s_{n-2}, s_{n-3}, \ldots, s_2, s_1, s_0$, the destinaiton node $d = d_{n-1}, d_{n-2}, d_{n-3}, \ldots, d_2, d_1, d_0$, and the current node $c = c_{n-1}, c_{n-2}, c_{n-3}, \ldots, c_2, c_1, c_0$.

1. k=LeastOne($c \oplus d$). The function LeastOne() returns the location of the least significant 1 among two n digit binary numbers, c and d. For example, if $c = 10010$ and $d = 01110$, $c \oplus d$=11100. And the function LeastOne() returns 3, because the least significant 1 is the 3rd bit from right.

2. If $k < n$, forward the packet to the neighboring node $c \oplus 2^{k-1}$ using the kth dimension channel.

3. If $k = n$, forward the packet to the local processor.

Figure 4.12 shows an example of the E-cube routing algorithm. Let the source node $s = 0010$, and the destination node $d = 1101$. Current node $c = 0010$, and $c \oplus d = 1111$, therefore k=LeastOne($c \oplus d$)=1. Thus the packet is passed through the channel on the dimension 1.

Next, the current node $c = 0011$, and $c \oplus d = 1110$, and k=LeastOne($c \oplus d$). Therefore, the packet is passed through the dimension 2, and so on. Finally, the packet

arrives at the destination node 1101. The XY and the E-cube routing algorithms

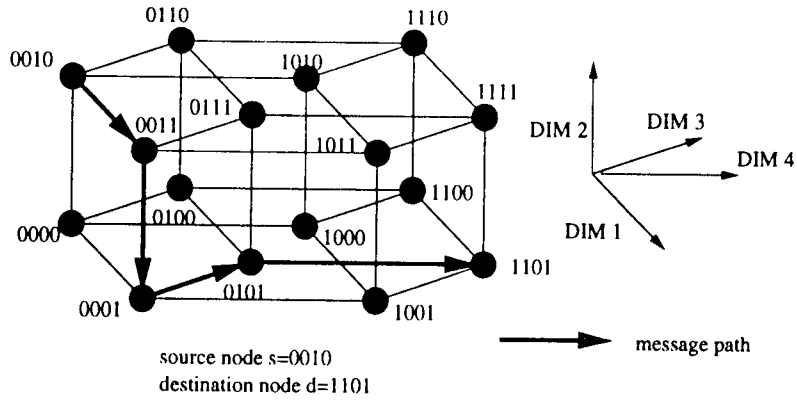


Figure 4.12. The E-cube routing algorithm on 4-cube.

both are minimal routing algorithms and guarantee freedom from deadlock. Moreover, both can be employed in either store-and-store and wormhole routing. Unfortunately, it is impossible to construct a deadlock-free, minimal, and deterministic dimension ordering routing algorithm for k-ary n-cube, such as torus [33].

As seen in page 37, if some virtual channels are added to deterministic algorithms, the performance can be improved due to better channel utilization. For example, J-machine network in MIT [19] uses the E-cube routing algorithm and two virtual channels that share the same physical channel.

4.3.2. Adaptive routing algorithms

As shown in Figure 4.13, adaptive routing can have packets choose one path among alternate paths so that they can proceed without interfering with other packets. It also provides fault tolerance capability because adaptive routing algorithms can detour around faulty nodes or channels.

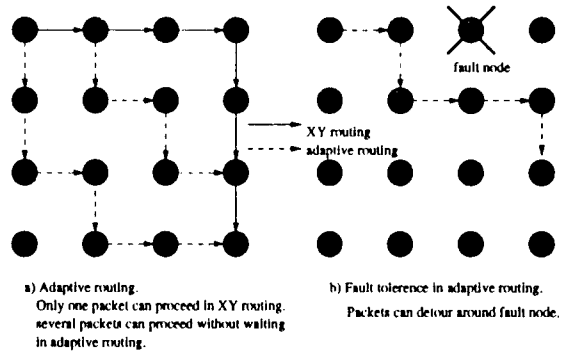


Figure 4.13. Adaptive paths.

4.3.2.1. Double-Y channel routing algorithm

The Double Y-channel routing algorithm is a minimal and fully-adaptive routing algorithm. This routing algorithm is discussed in [25]. As illustrated in Figure 4.14, the Double-Y channel routing algorithm needs two pair of opposite directional physical channels in the Y-axis and two opposite directional physical channels in the X-axis. Thus, it can be partitioned into two subnetworks, -X and +X. Since any subnetwork does not create a dependency circle, it is deadlock free. This routing algorithm is expensive to construct because the most costly parts in a network are the physical channels (wires).

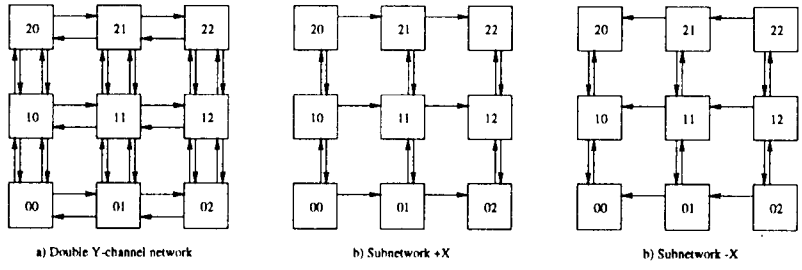


Figure 4.14. The Double Y-channel routing algorithm

4.3.2.2. Turn models

C. J. Glass and L. M. Ni proposed the *turn models* in [24], which are partially adaptive routing algorithms for mesh and k-ary n-cube networks without any adding physical channels or virtual channels. They introduced three turn models: *West-First*, *North-Last*, and *Negative-First*. These algorithms can be minimal or nonminimal. The basic idea of turn models is to restrict some turns for freedom from deadlock as in the XY routing algorithm. Turn models prohibit only two turns, while the XY routing algorithm restricts four turns. Turn models provide adaptivity that the XY routing algorithm can not. As shown in Figure 4.15, prohibiting only two turns guarantees freedom from deadlock, since no channel dependency circle can be created without the prohibited two turns.

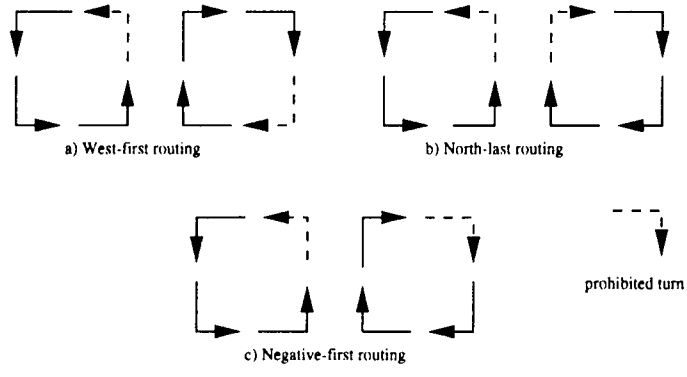


Figure 4.15. Turn models.

The West-First routing algorithm routes a packet adaptively south, east, and north prohibiting two turns, north-to-west and south-to-west - thus named "west-first". For minimal routing, if the destination node is located on right-hand side of the source node, then the algorithm is fully adaptive; otherwise, it is deterministic. If nonminimal routing is allowed, the West-First routing algorithm is adaptive in either case [33]. The North-Last routing algorithm restricts north-to-east and north-

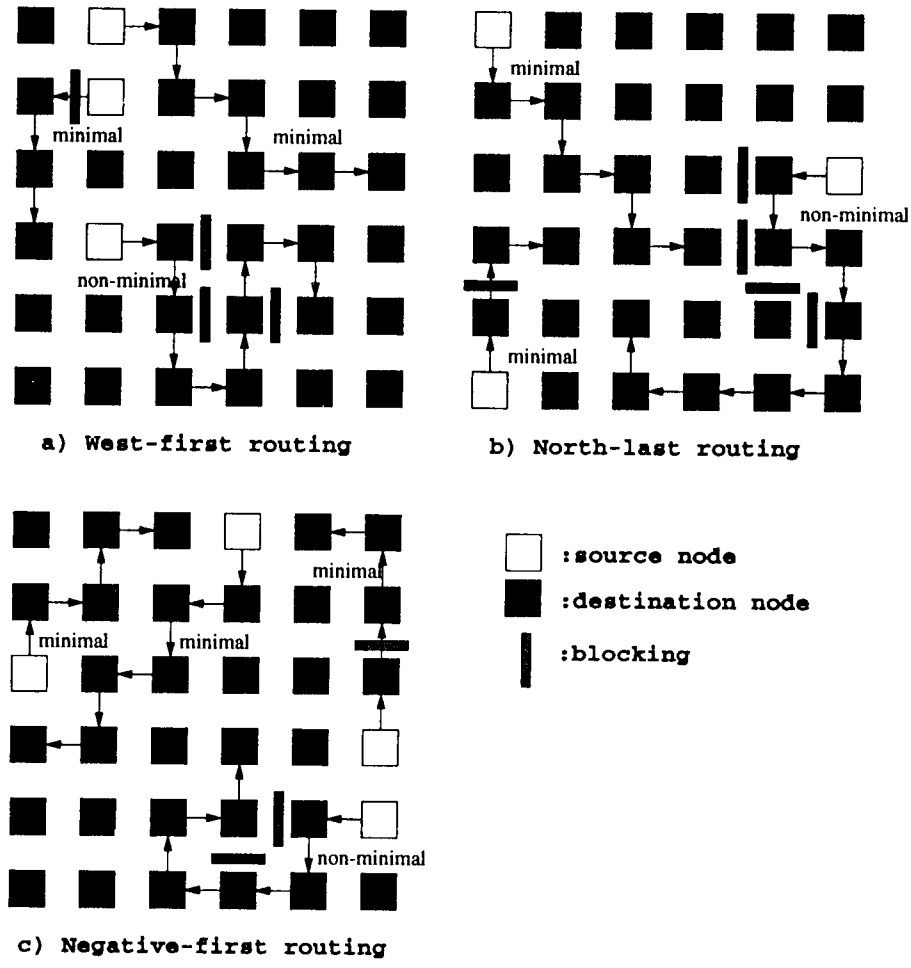


Figure 4.16. Routing examples for turn models.

to-west turns. In the Negative-First routing algorithm, north(positive)-to-west and east(positive)-to-south turns are not allowed.

The number of possible shortest paths (S) from the source node (d_x, d_y) to the destination node (s_x, s_y) is

$$S_{west-first} = \begin{cases} \frac{(\Delta x+\Delta y)!}{\Delta x!\Delta y!} & \text{if } d_x \geq s_x \\ 1 & \text{otherwise} \end{cases}$$

$$S_{north-last} = \begin{cases} \frac{(\Delta x+\Delta y)!}{\Delta x!\Delta y!} & \text{if } d_y \leq s_y \\ 1 & \text{otherwise} \end{cases}$$

$$S_{negative-first} = \begin{cases} \frac{(\delta x+\Delta y)!}{\Delta x!\Delta y!} & \text{if } d_x \leq s_x \text{ and } d_y \leq s_y \\ & \text{or if } d_x \geq s_x \text{ and } d_y \geq s_y \\ 1 & \text{otherwise} \end{cases}$$

Figure 4.16 shows the routing examples of turn models.

4.3.2.3. Hop schemes

Hop schemes are the typical routing algorithms that use the buffer pool method to avoid deadlock. They were designed originally for SAF routing. Boppana and Chalasani showed that with virtual channels, the hop schemes for wormhole routing could be derived from the hop schemes for SAF [37]. Figure 4.17 illustrates the derivation of wormhole routing from SAF routing. These algorithms are minimal and fully-adaptive.

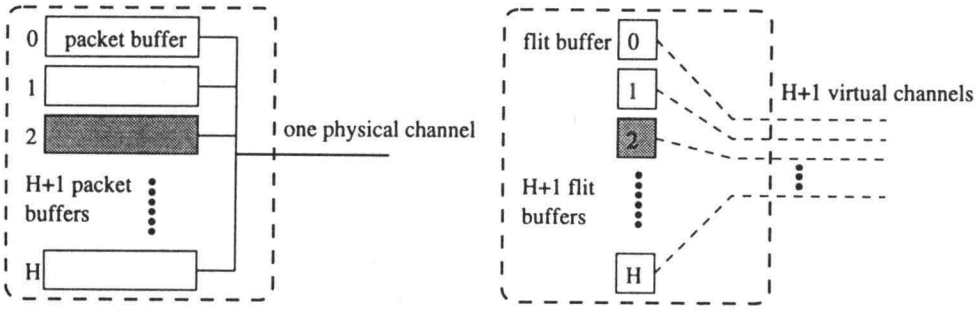


Figure 4.17. Hop scheme for wormhole routing is derived from that for SAF routing.

Positive hop algorithm : As mentioned on page 32, a positive hop routing algorithm with a conventional buffer pool method needs $H+1$ buffers in a router, where

H is the diameter of the network. The buffers are numbered consecutively from 0 to indicate the classes.

A message is placed in the buffer class 0 in the source node. As the packet proceeds, the number of hops is counted and the packet is placed in the buffer class of the counted number in a intermediate node. Therefore, the number of packet buffers in a node is the diameter plus 1. For instance, in a $n \times n$ mesh, the diameter is $2(n-1)$, and thus $2(n-1)+1$ buffers are required in this algorithm. Since

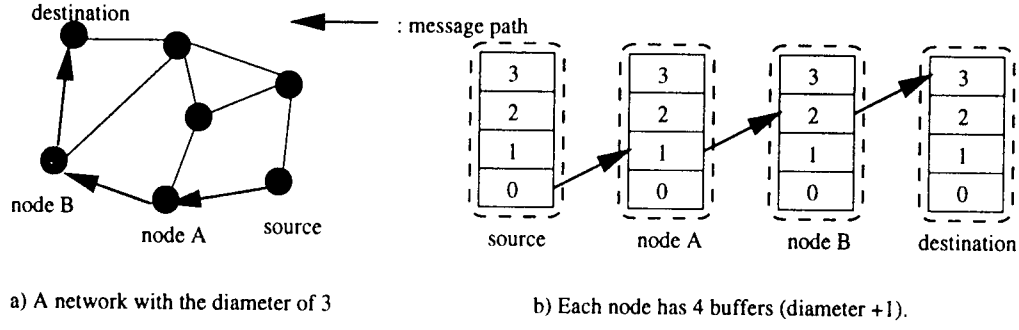


Figure 4.18. Positive hop scheme

buffers have monotonically increasing ranks, packets cannot make any dependency circle, thus freedom from deadlock is guaranteed. Figure 4.18 shows the positive hop scheme.

Negative hop algorithm : The negative hop scheme was introduced by Gopal to reduce the number of buffers required in the positive hop scheme [29]. In the negative hop scheme, a network is partitioned into several subsets, such that no subset contains adjacent nodes. These subsets are labeled 1, 2, 3, $\cdots$, M. The hop is counted as a negative hop only if a message moves from a higher labeled node to a lower labeled node. This method significantly reduces the number of negative hops and thus the number of buffers. The other parts of this algorithm are the

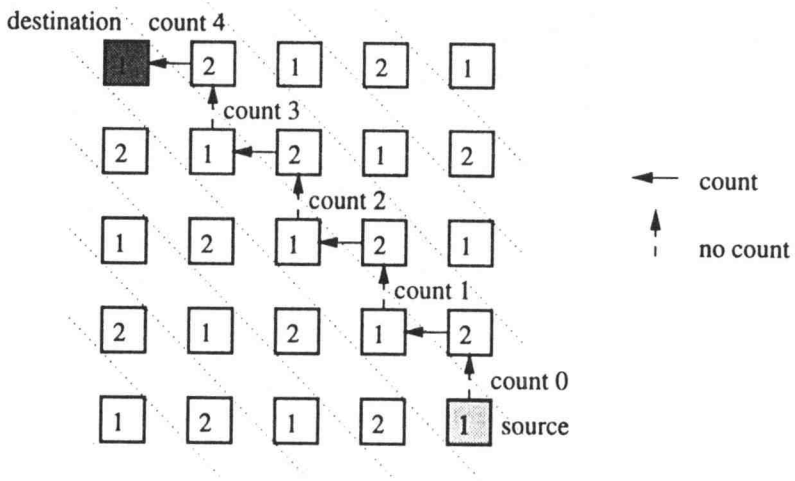
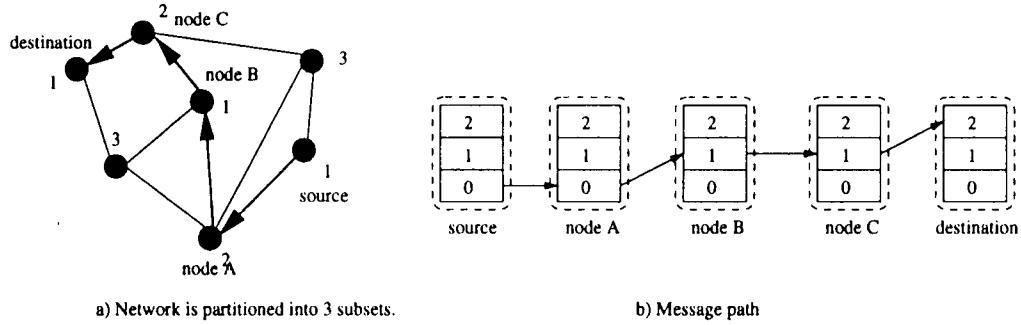


Figure 4.19. Negative hop routing in 5×5 mesh

same as those in the positive hop. For instance, if k is an even number, a k-ary n-cube is partitioned into 2 subsets. Thus, the maximum number of negative hops a message takes is at most half the diameter of the network, which equals $\lceil n\lfloor k/2 \rfloor/2 \rceil$. Therefore, the number of buffers required in a node in the negative hop scheme is $\lceil n\lfloor k/2 \rfloor/2 \rceil+1$ [29]. In a $n \times n$ mesh, the network is partitioned into 2 subsets and the number of buffers required in a node in a negative hop scheme is $\lceil n-1 \rceil+1$ (See Figure 4.19.). Figure 4.20 shows an example of network partition and negative hop routing.

4.3.2.4. Two-power-n routing

Another fully adaptive deadlock free algorithm for mesh and k-ary n-cube is the two-power-n routing. In the two-power-n routing, each node has $(n+1)2^{n-1}$ virtual channels for k-ary n-cube and 2^{n-1} virtual channels for mesh to be fully-adaptive. This algorithm is discussed in [38]. This algorithm uses the smallest

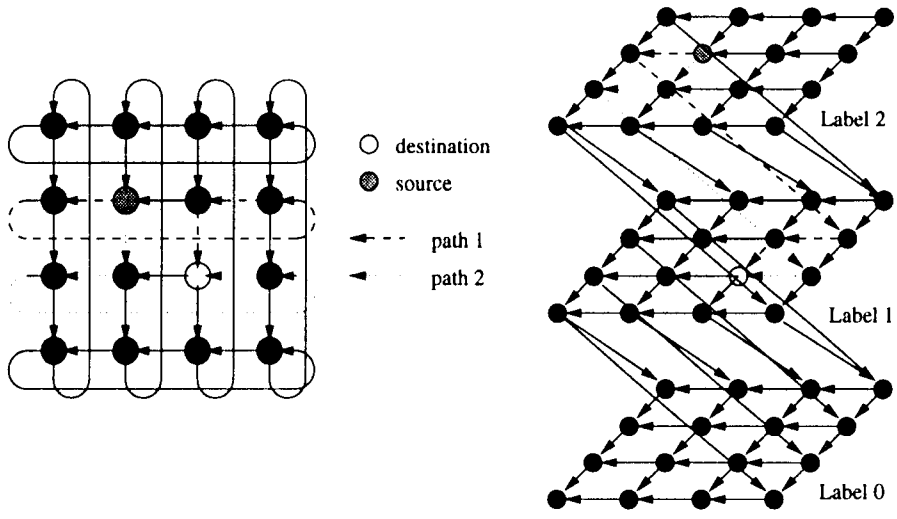


a) Network is partitioned into 3 subsets.

b) Message path

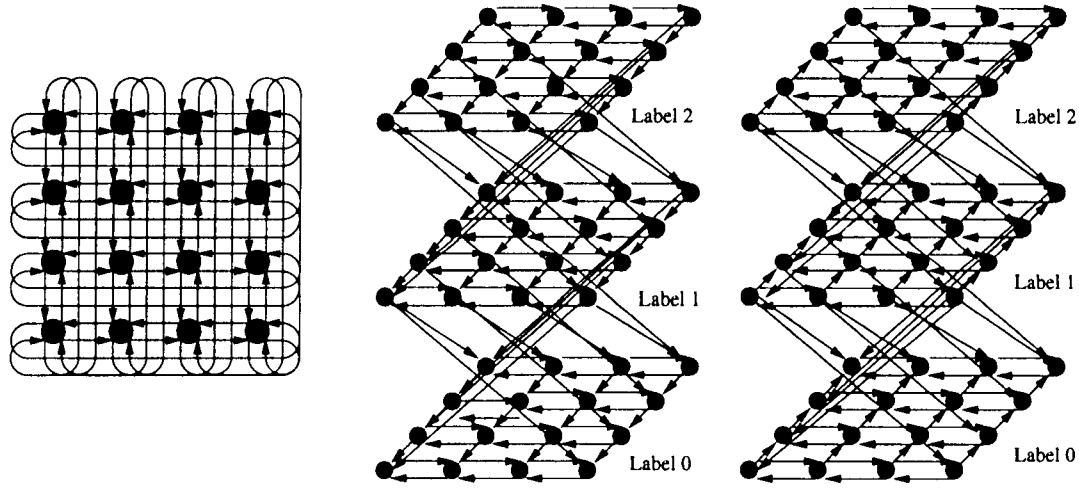
Figure 4.20. An example of network partition and negative hop scheme

number of virtual channels among the minimal fully-adaptive routing algorithms proposed so far. A physical k-ary n-cube or mesh network is divided into several virtual networks that guarantee freedom from deadlock. The virtual nodes and virtual channels in a virtual network can be mapped to the physical network. In order to eliminate deadlock due to the end-round connections, a virtual node has a label (integer 0 to n) in each *virtual network* VI_i, where i represents the ith virtual network. Whenever a packet proceeds through an end-around connection, it goes down by one level as shown in Figure 4.21. The number of levels equals the maximum number of end-around edges through which a packet can travel. For minimal routing in k-ary n-cube, the number of levels is $n+1$. Since a virtual network in each level is deadlock-free, and the virtual channels which go down to a lower level are unidirectional, no channel dependency cycle exists, thus it is deadlock-free. The bidirectional k-ary n-cube has two virtual networks. Generally, a bidirectional k-ary n-cube has 2^{n-1} virtual networks [38]. Figure 4.22 illustrates the physical and virtual networks of a bidirectional 4-ary 2-cube.



a) A unidirectional 4-ary 2-cube physical network b) Virtual network for unidirectional 4-ary 2-cube

Figure 4.21. A unidirectional 4-ary 2-cube has only one virtual network. There are 2 shortest paths between the source (2,1) and destination (2,1). To ensure freedom from deadlock, packets step down by one level through end-around edges and never go up.



a) A bi-directional 4-ary 2-cube physical network b) 2 virtual networks for bi-directional 4-ary 2-cube

Figure 4.22. A bidirectional 4-ary 2-cube has 2 virtual networks. In each network, the routing algorithm is the same as that in unidirectional.

5. THE PROPOSED METHOD: LOOK-AHEAD ROUTING

Currently, message routing through a network is done rather blindly. In other words, packets for messages are sent only with a destination address, and conflicts among packets are not resolved until after they occur. However, if conflicts can be predicted ahead of time, packets can be rerouted, perhaps, reducing the number of bad choices for routing and thus waits.

5.1. WAITS IN WORMHOLE ROUTING

A message consists of a number of packets. A packet is the basic unit which contains the destination address for routing purposes and can be divided into a number of fixed length flits. The flits within a packet are inseparable during their transit through network using wormhole routing. Thus, a packet is like a train; when a train runs, it occupies a length of the track (channels) proportional to the number of cars (flits). If it stops to wait, it stays on the track for some time, not allowing other trains to run on the same track. Therefore, any packet waiting may cause other packets to wait and in turn may cause more packets to wait. This problem becomes more serious as the packet length increases.

If a channel (a flit buffer) is occupied by a flit at any given clock cycle, the probability that it will be occupied by the same packet in the next cycle is $(n-1)/n$, where n is the number of flits in a packet. This is because the channel is used by the same packet until the entire packet passes through the channel. For example, if a packet has an average of 8 flits and a channel is occupied by one of the flits in the packet during a particular clock cycle, the probability that the channel will be occupied in the next cycle by the same packet is 7/8. In a linear network, if a channel

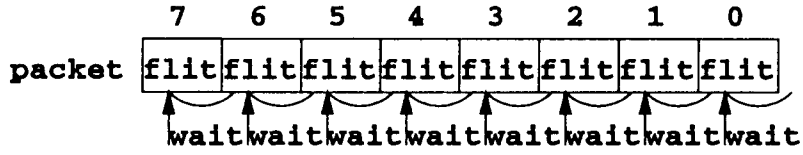


Figure 5.1. Once a packet is blocked , it waits average 4.5 clock cycles.

is occupied by a packet, other packets must wait for at least 1 cycle (present cycle) or at most n cycles (it is assumed that the packet proceeds without waiting until the entire packet passes the channel) to use the channel (Figure 5.1). Therefore, the average number of wait cycles is $\frac{\sum_{i=1}^{n} i}{n} = (n+1)/2$. For example, when the number of flit is 8 per packet, once a packet is blocked, it must wait average 4.5 clock cycles.

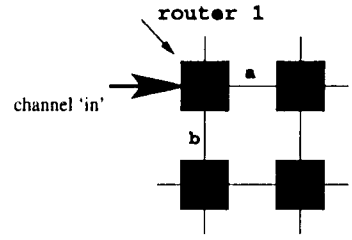


Figure 5.2. The waits in a 2D mesh

In a 2D mesh network shown in Figure 5.2, if the channels, 'a' and 'b' are busy and a packet is waiting for one of those two channels at router 1 (i.e., a packet is waiting at channel 'in' to be routed by router 1 to 'a' or 'b'), the probability that the packet has to wait in the next cycle is $(\frac{n-1}{n})^2$ because the probability for both channels 'a' and 'b' to be busy at the next cycle is $(n-1)/n \times (n-1)/n$, and

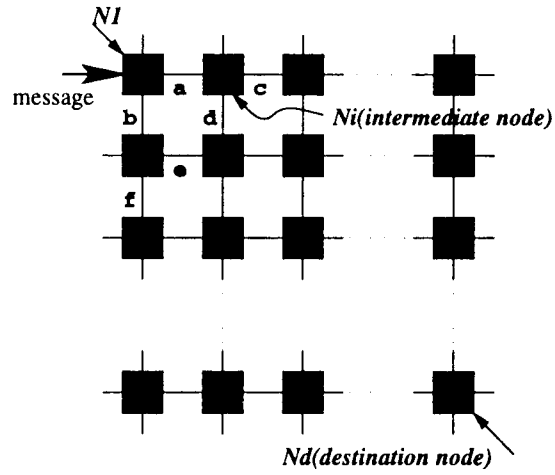


Figure 5.3. The channels considered for the look-ahead scheme in 2D mesh

the average number of the wait cycles is $\frac{\sum_{i=1}^{n} i(2n-2i+1)}{n^2}=(n+1)(2n+1)/2n$ (See Appendix A for more detail.).

5.2. LOOK-AHEAD TO AVOID BAD CHOICES

A look-ahead scheme can be applied to any adaptive algorithm. Consider the West-First routing algorithm using minimal routing in a 2D mesh shown in Figure 5.3, where $d_x > N1_x$ and $d_y < N1_y$. Suppose that a packet is waiting for channel 'a' or 'b' in the router $N1$. Assuming that channels 'a' and 'b' are idle, and the channel 'a' (X-axis) has priority to be chosen over the channel 'b', the West-First routing algorithm will then route the packet to the channel 'a' even though both channels 'c' and 'd' are busy. Therefore, the packet will wait an average of $(n+1)(2n+1)/2n$ cycles in node Ni. These waits are wastful if one or both channels 'e' and 'f' are idle. If a router can look ahead, the packet can be rerouted to channel 'b' instead of channel 'a'. The packet will not wait at the next clock

Cases	Channels						Channel selection
	a	b	c	d	e	f	
1	1	1	x	x	x	x	must wait
2	1	0	x	x	x	x	channel b is selected
3	0	1	x	x	x	x	channel a is selected
4	0	0	1	1	1	1	channel b is selected
5	0	0	1	1	1	0	channel b is selected
6	0	0	1	1	0	1	(can avoid unnecessary
7	0	0	1	1	0	0	waits at next cycle.)
8	0	0	0	1	x	x	channel a is selected
9	0	0	1	1	x	x	channel a is selected
10	0	0	0	0	x	x	channel a is selected

1: busy, 0: idle, x: don't care

Table 5.2. An example for one-hop look-ahead channel selection policy

cycle if one or both channels 'e' and 'f' are idle, because if a channel is idle at a particular clock cycle, the probability for the channel to be idle at the next cycle is very high. Therefore, the look-ahead scheme will improve channel utilization and network latency, thus the performance of a network.

Table 5.2 shows an example of one-hop look-ahead channel selection for the network in Figure 5.3. Look-ahead is effective only when both channels of 'a' and 'b' are idle and when both channels of 'c' and 'd' are busy. In the proposed look-ahead channel selection policy, channel 'b' is selected when channel 'c' and 'd' are busy without considering channels 'e' and 'f'. This reduces the hardware requirements because the routers do not need to look-ahead along the Y-axis. As shown in Table 5.2, only cases 5, 6, and 7 can take advantage of the look-ahead scheme.

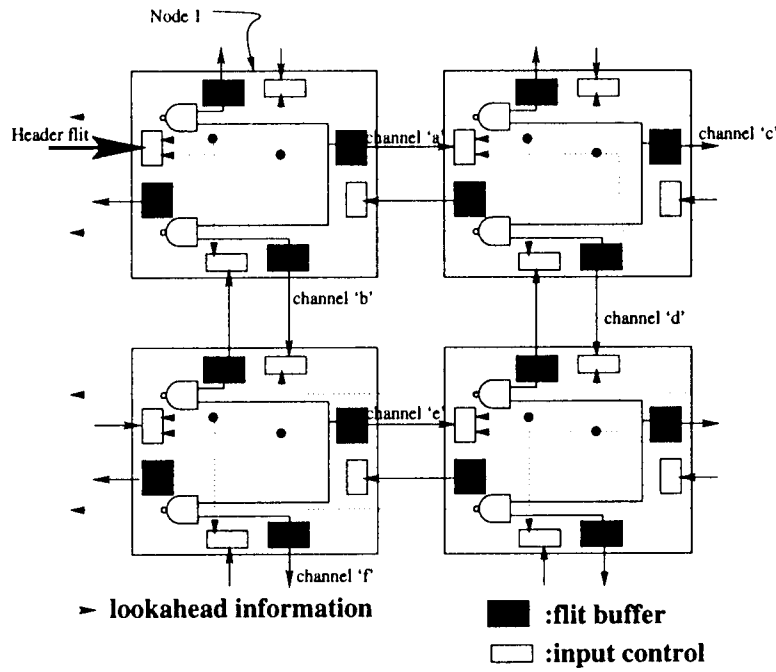


Figure 5.4. Hardware requirements for one-hop look-ahead scheme for West-First

5.3. HARDWARE REQUIREMENTS FOR ONE-HOP LOOK-AHEAD

The one-hop look-ahead scheme needs to add wires to send the look-ahead information. Figure 5.4 shows the hardware requirements of the one-hop look-ahead scheme used in conjunction with the West-First routing algorithm. As can be seen, each flit buffer has a flag bit which indicates the state of the buffer. If the flag bit is 0, it means that the flit buffer and the channel are being used (not available). Two wires (dotted lines in Figure 5.4) are added along the X-axis. The upper look-ahead line is used when $d_y > current\ Node_y$ and the lower look-ahead line is used when $d_y < current\ Node_y$. Thus, routers sense only one of the look-ahead lines depending on d_y. The upper look-ahead line senses the states of north and east buffers in the east neighboring router along the X-axis, while the lower look-ahead line senses the

states of south and east buffers. The look-ahead lines send 1 to the input controllers only when both buffers are not available. If the look-ahead line is 1, the router routes the packet to the Y-axis by appropriately setting the switch element (the switch boxes are omitted in Figure 5.4). These look-ahead lines act like a red signal light not allowing packets to proceed along the X-axis when there is contention. For example, in Figure 5.4, a packet is waiting at node 1. The destination (d_x, d_y) address the packet is carrying is $d_x >$ node-1_x and $d_y <$node-1_y. If the channels 'a' and 'b' are idle, then the packet will be routed to channel 'b' only if the lower look-ahead line is 1, which means that the channels 'c' and 'd' are busy.

The look-ahead lines along the Y-axis are not necessary for the example in Table 5.2 because a router can decide the next channel only with the states of the buffers in the east neighboring node. Moreover, opposite look-ahead lines along the X-axis also are not needed in the West-First routing algorithm because the routing is deterministic when $d_x < s_x$.

6. SIMULATION RESULTS

To simulate the performances of various routing algorithms using wormhole routing, a 16×16 mesh interconnection network and 4 different packet sizes, 4, 16, 32, and 64 flits are used. The algorithms simulated were the XY, the West-First, and the Double-Y channel routing algorithms, which are deterministic, partially-adaptive, and fully-adaptive, respectively. The look-ahead scheme was applied to the West-First routing algorithm and the Double-Y channel routing algorithm.

The simulation programs were written in C^{++} containing approximate 2,500 lines of code and provide a flit-level detail simulation. The simulators accept a number of input parameters, such as the mesh size, the packet size, the packet generation rate of each node, and the simulation time. The following assumptions were made for the simulators:

- One clock cycle is needed for a packet to request a channel and be routed to a channel.
- It takes one clock cycle to transfer a flit from a router to a neighboring router. It also takes one clock cycle to inject the header flit of a packet into the network from a source node.
- The network is synchronized to move all the serial flits in a packet in one clock cycle.
- Messages are unformly generated over the network by a given packet generation rate and also have uniformly distributed destinations.

Each simulation was run for 20,000 clock cycles to gather a variety statistics as outputs. Data was gathered about the average latency, the total number of packets

injected into the network, the average number of waits per block, and the total number of waits and blocks versus the packet generation rate*.

The following sections discuss the simulation results for the packet sizes of 4 flits and 16 flits. These two flit sizes were chosen as representatives because other packet sizes produced similar results. See Appendix B for more detail.

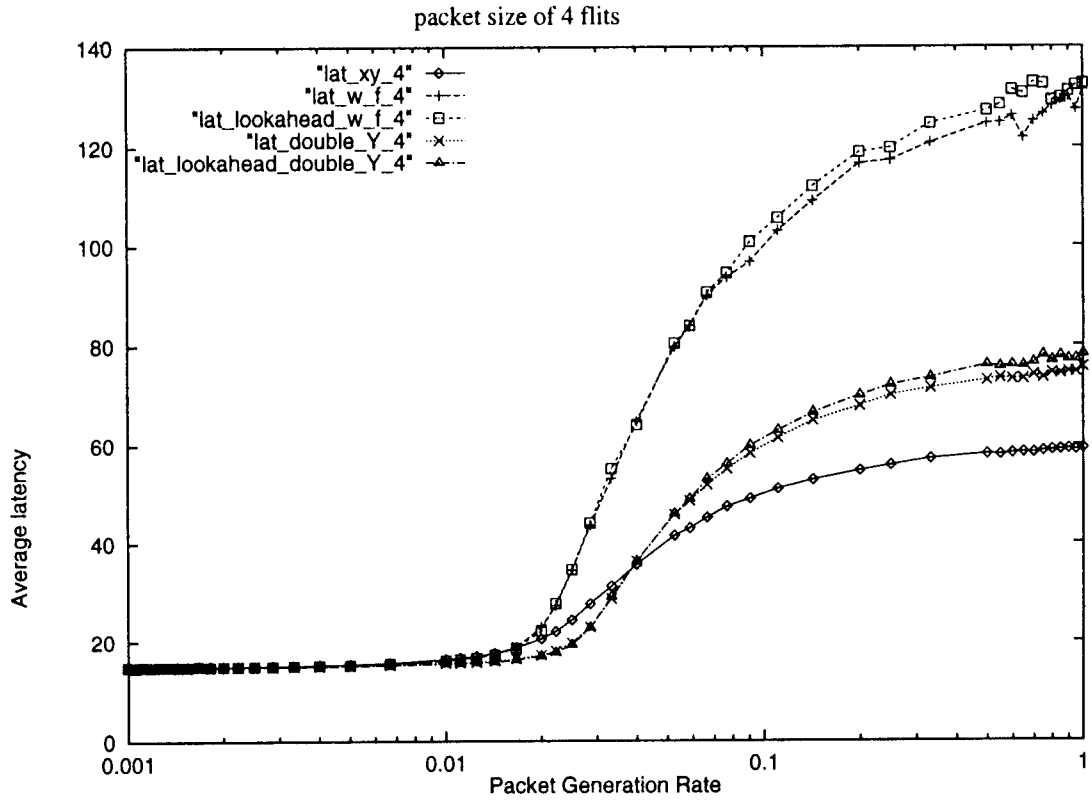
6.1. AVERAGE LATENCY

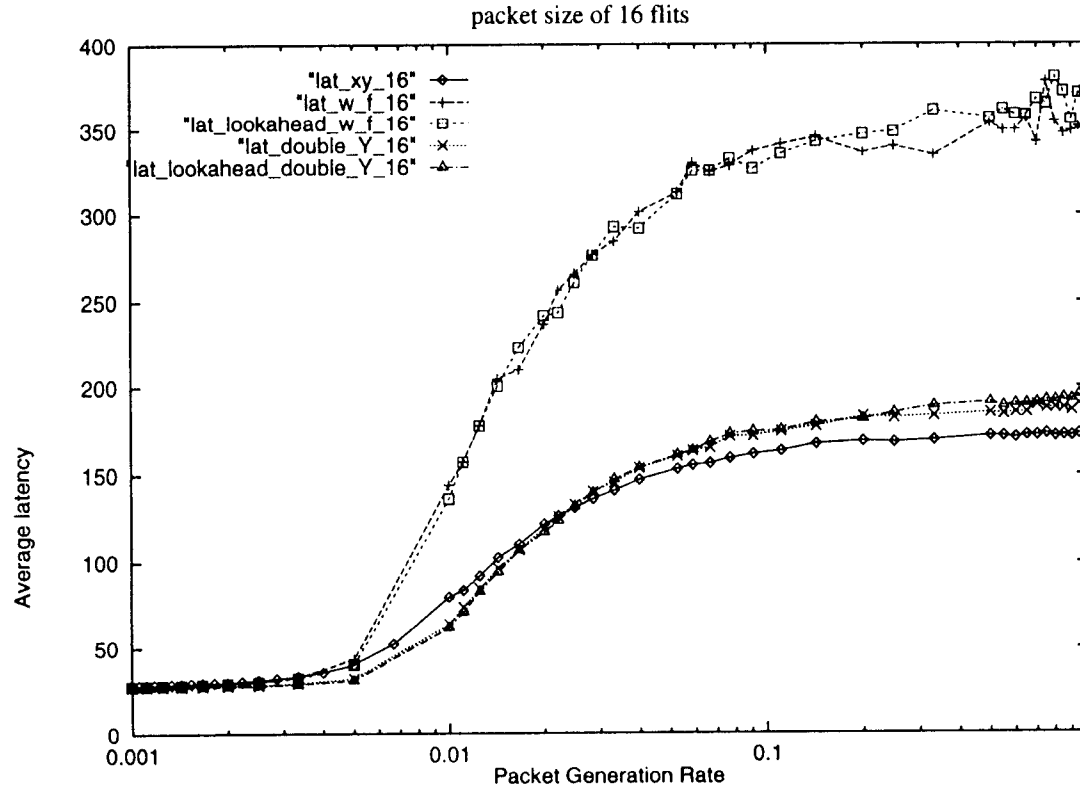
The average latency is one of the most important factors that determines the performance of a routing algorithm. It is simply defined as the elapsed time between the injection of the header flit of a packet into the network and the arrival of the tail flit at the destination node. In other words, it is the total time a packet resides in the network. Latency depends on the wait time, the message length measured in flits, and the average network diameter. At a first glance, it would seem that the more adaptive a routing algorithm is, the less latency it would have. However, the simulation results prove that this is not the case.

Figure 6.1 shows that the average latencies of the deterministic XY routing algorithm are always less than the partially-adaptive West-First routing algorithm. The average latencies of both routing algorithms start to saturate at a packet generation rate of 0.02. The term 'saturate' is used to indicate when packets begin to experience contention. The West-First routing algorithm saturates more rapidly than the XY routing algorithm. The fully-adaptive Double-Y channel routing algorithm begins to saturate at a packet generation rate of 0.03 and its average latency is slightly less than that of the XY routing until the packet generation rate reaches

---

*The packet generation rate means the probability that a packet is generated by a node when the node is not blocked.


packet size of 4 flits
"lat_xy_4"
"lat_w_f_4"
"lat_lookahead_w_f_4"
"lat_double_Y_4"
"lat_lookahead_double_Y_4"
Average latency
Packet Generation Rate
0
20
40
60
80
100
120
140
0.001
0.01
0.1
1


packet size of 16 flits
"lat_xy_16"
"lat_w_f_16"
"lat_lookahead_w_f_16"
"lat_double_Y_16"
lat_lookahead_double_Y_16"
Average latency
Packet Generation Rate
0
50
100
150
200
250
300
350
400
0.001
0.01
0.1
1

0.04. In the saturation region, the XY routing exhibits much better performance than the West-First and the Double-Y channel routing algorithms. The look-ahead schemes for partially- and fully-adaptive routing algorithms had a negligible improvement or a slightly worse effect on latency especially in the saturation region. Figure 6.1 illustrates that as the average packet length becomes longer, the saturation occurs earlier, and the latency curve of the XY routing algorithm in the saturation region approaches that of the Double-Y channel routing algorithm.

6.2. TOTAL NUMBER OF PACKETS INJECTED INTO NETWORK

The total number of packets injected into a network at a given time represents the throughput of the network. Throughputs of the five algorithms all increase together until saturation, as seen in Figure 6.2. It should be noted that there are peak points for adaptive routing algorithms. The Double-Y channel and the West-First routing algorithms have their throughput peak at packet generation rates of 0.032 and 0.022, respectively. As shown in Figure 6.2, the West-First routing algorithm performs worse than the XY routing algorithm, and the Double-Y channel routing algorithm has better throughput until the the packet generation rate reaches about 0.06. As seen in Figure 6.1 and Figure 6.2, the deterministic XY routing outperforms the partially-adaptive West-First routing algorithm over all ranges of packet generation rate and also outperforms the fully-adaptive double-Y channel routing algorithm in the saturation region. Moreover, the effects of the look-ahead versions on latency and throughput are worse in the saturation region.

6.3. ANALYSIS

To investigate the above results, the total number of blocks and the average number of waits for each block were calculated. Figure 6.3 illustrates the total

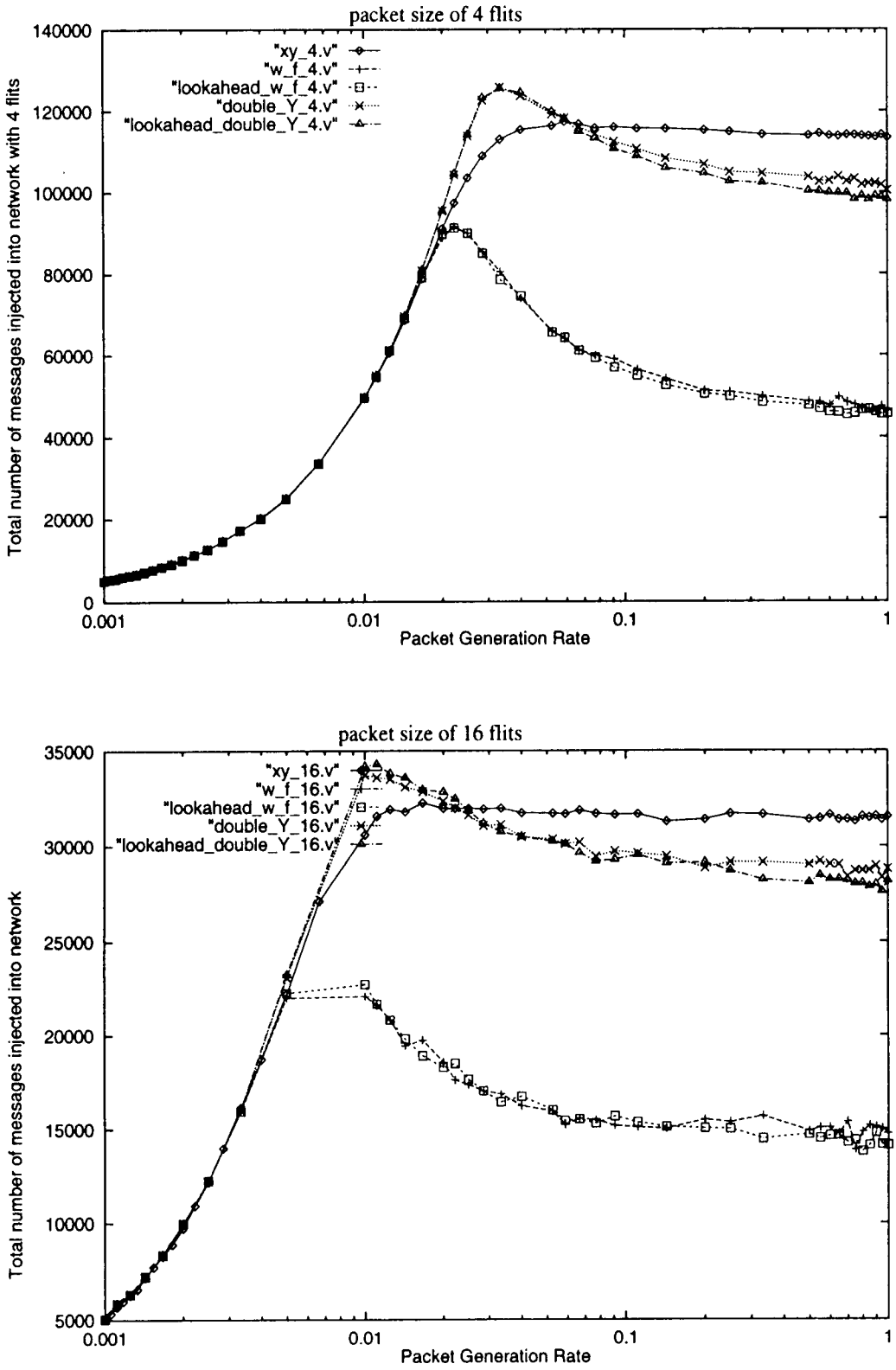


Figure 6.2. Comparison of the total numbers of messages injected into network within 20,000 clock cycles.

number of blocks in the network and the average number of waits per block. As shown in Figure 6.3, adaptive routing algorithms significantly reduced the number of blocks. However, the number of waits per block was larger than in the deterministic XY routing algorithm. Thus, even though adaptiveness can avoid some blocks temporarily, if a packet is blocked once, it must wait longer causing more latency and less throughput.

The look-ahead versions reduced the number of blocks slightly, but increased the number of waits per block. The disappointing results of the look-ahead versions are basically caused by the additional adaptiveness introduced into an already adaptive routing algorithm.

Figure 6.4 compares the numbers of waits at each flit buffer when the packet size is 4 flits. It shows that the West-First routing algorithm has severe contention at the nodes and the west flit buffers. The contention problem in the west flit buffers causes the contention in the nodes and then the contention in nodes prohibits the packets from being injected into the network, degrading the overall performance. This is one of the reasons the West-First routing algorithm performs worst among the three algorithms.

The simulation results show that partially- or fully-adaptive routing algorithms do not necessary provide better latency and throughput than deterministic routing algorithms, and the look-ahead effect added to adaptive algorithms has a negligible effect. The West-First routing algorithm looks very attractive at first because it provides adaptivity without adding any physical or virtual channel. However simulation results show that it actually degrades the network performance causing early contention and saturation. It also shows that, especially in the saturation region, the XY routing algorithm outperforms adaptive routing algorithms.

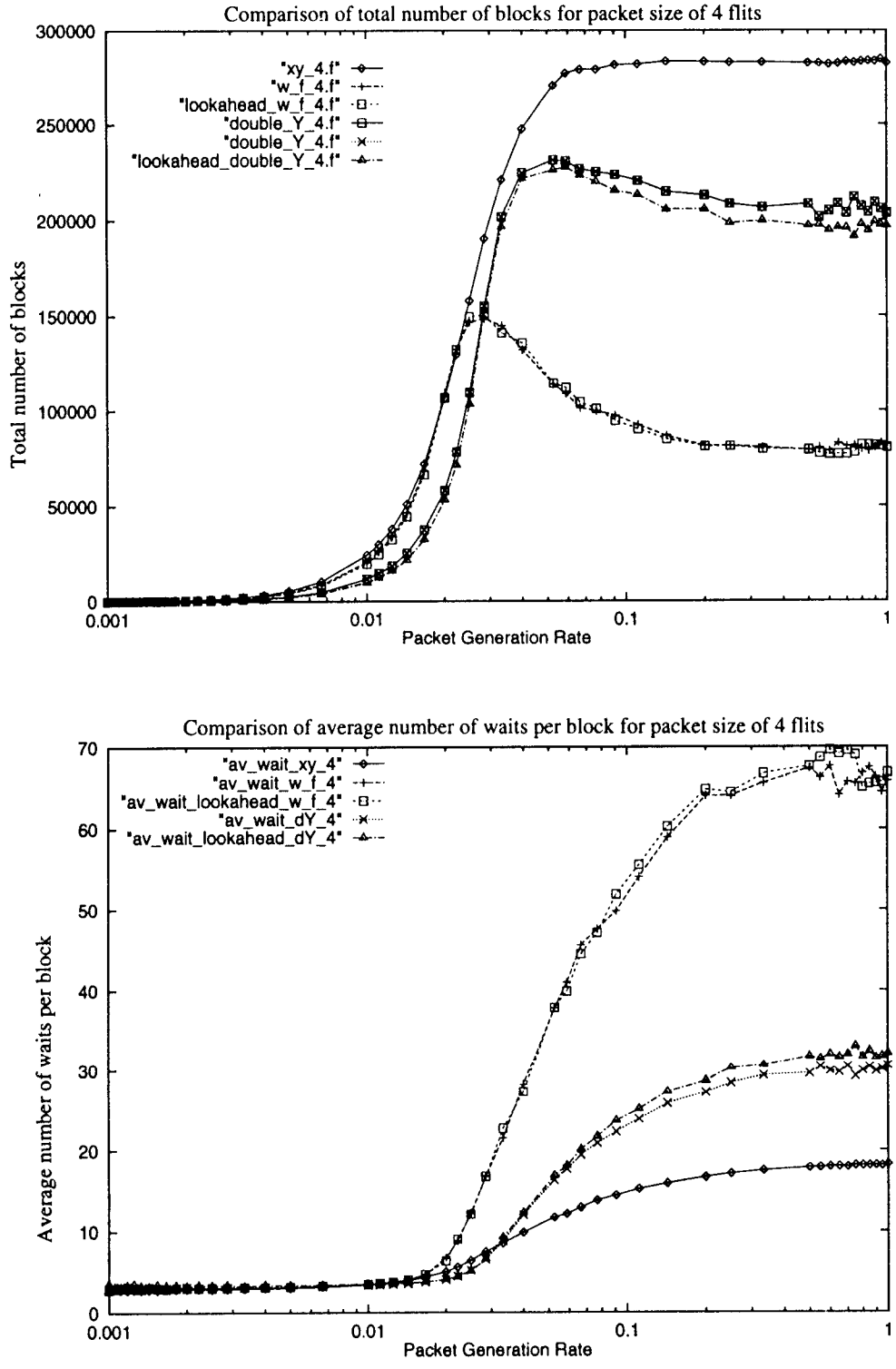


Figure 6.3. The average number of waits per block explains the reason adaptive and look-ahead routing algorithms have inferior latency figures. Even though the XY routing has more blocks than the other algorithms, it has fewer waits per block, thus resulting in less overall latency results.

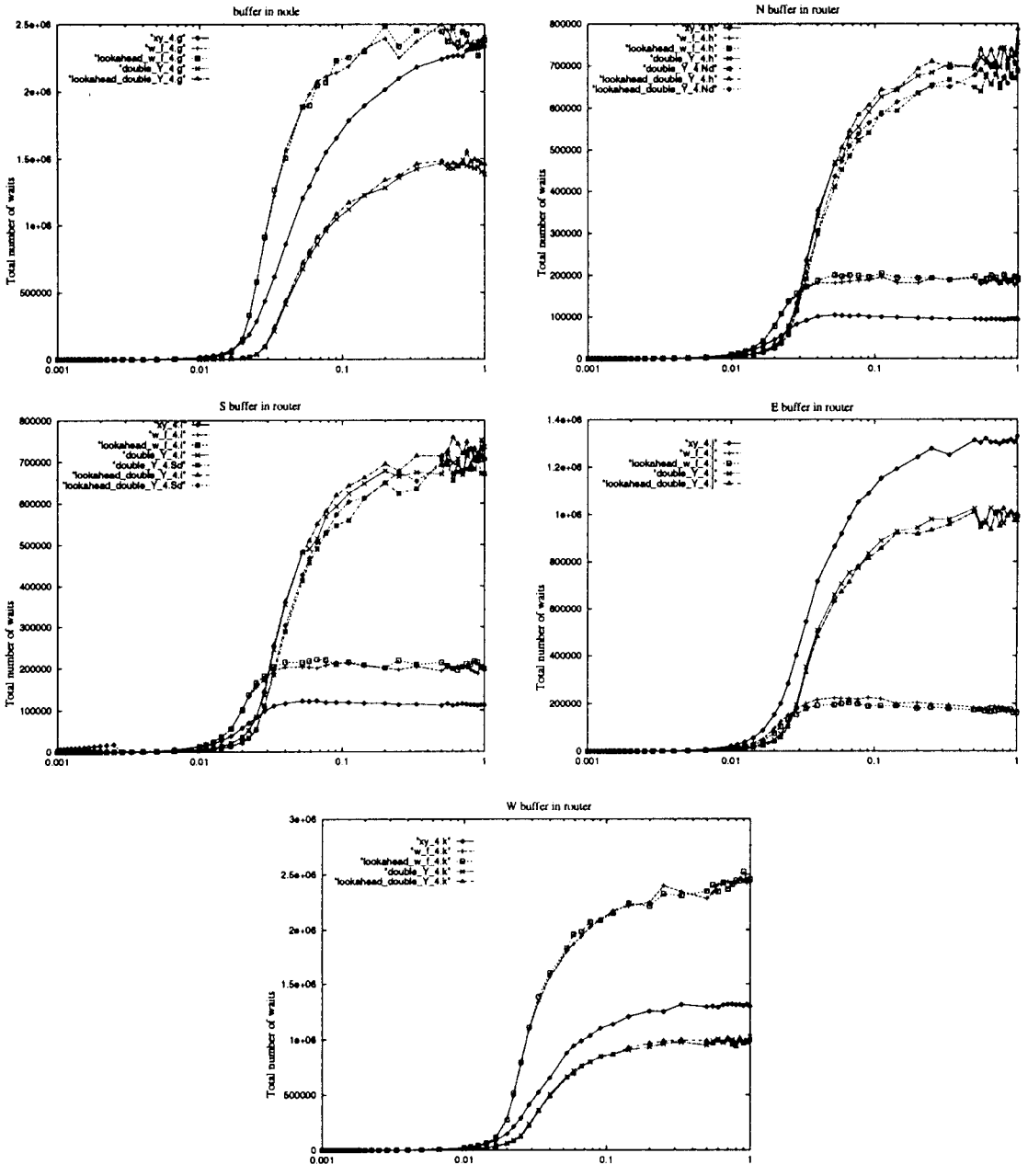


Figure 6.4. A comparison of the number of waits at each buffer in a 16×16 mesh.

7. CONCLUSION REMARKS

In this thesis, various adaptive and deterministic routing algorithms have been surveyed and the look-ahead routing scheme was introduced as a cost-effective method for increasing adaptiveness. In order to compare the performance, the XY, the West-First, and the Double-Y channel routing algorithms have been simulated using the wormhole routing technique. The one-hop look-ahead routing scheme has also been examined for both the West-First and the Double-Y channel routing algorithms. The simulation study shows that fully- or partially-adaptive routing algorithms do not necessarily improve the network performance, rather the West-First routing algorithm degrades network performance over all ranges of packet generation rates. The Double-Y channel routing algorithm slightly outperforms the XY routing algorithm until the saturation region; however, in the saturation region the reverse occurs. The simulation results also show that the look-ahead effects are almost negligible. The reason for this is even though adaptive and the look-ahead routing schemes reduce number of blocks temporarily, they increase waits per block. Thus, performance is not improved.

The simulation study in this thesis has been limited to only 2-D mesh networks which have 1 or 2 flit buffers (1 or 2 channels) in each direction. Boppana and Chalasani reported similar results for 16-ary 2-cube tori in [37]. But they did not analyze the cause of their results. They also simulated the positive-hop and the negative-hop routing schemes, which use many flit buffers (many virtual channels) and are fully-adaptive. Their simulation results showed that the positive-hop and the negative-hop schemes improved latency and channel utilization.

These simulation results prove that adaptive and look-ahead routing schemes in a static network, which has a small number of channel resources, do not result in improved latency and throughput compared to deterministic routing algorithms. In such networks, adaptiveness is meaningful only for fault tolerance.

Future study of adaptive routing algorithms will be focused on the routing algorithms which use many virtual channels to improve better performance. Adaptive and look-ahead routing schemes will be focused on fault tolerance. Fault tolerance is becoming a big issue for Massively Parallel Processing. It would be unfortunate if multicomputers which have more than 1,000 nodes do not work because of one or two faulty nodes. In order to avoid this problem, adaptive routing is necessary. Especially, non-minimal routing will be required to provide high fault tolerance.

BIBLIOGRAPHY

[1] D.J. Farber and K. C. Larson,"**The System Architecture of the Distributed Computer System - The Communications System,**" in *Proc. Symp. on Comput. Commun. Networks and Teletraffic,* Brooklyn Polytechnic Press, Apr. 1972, pp. 21-27.

[2] Kai Hwang **"Advanced Computer Architecture, Parallelism, Scalability, Programmability"**, McGraw-Hill, Inc. 1993.

[3] R. M. Metcalfe and D. R. Boggs,**"Ethernet: Distributed Packet Switching for Local Computer Networks"**,*Commun. Ass. Comput. Mach.*, vol. 19, pp. 395-403, July 1976.

[4] Intel Corporation, *A Touchstone DELTA System Description,* 1991.

[5] C. L. Seitz, W. C. Athas, C. M. Flaig, A. J, Martin, J. Seizovic, C. S. Steels, and W. -K. Su,**"The Architecture and Programming of the Ametek Series 2010 Multicomputer,"**,*Proc. 3rd Conf. Hypercube Concurrent Computers and Applications*, vol. I, (Pasadena, CA), pp. 33-36, Jan. 1988.

[6] Larry D. Wittie,**"Communication Structures for Large Networks of Microcomputers,"** in*IEEE trans. on Comput.*, vol. C-30, No 4, Apr 1981, pp. 264-273.

[7] M. Maekawa *et al.*,**"Experimental Polyprocessor System (EPOS)-Operating System,"** in*Proc. 6th Symp. on Comput. Arch.*, Apr. 1979, pp. 188-195.

[8] S. I. Saffer *et al.,* **"NODAS-The Net Oriented Data Acquisition System for the Medical Environment,"** in *AFIPS Conf. Proc.,* vol. 46, NCC 1979, pp. 295-300.

[9] W. D. Hillis,**"The Connection Machine"**. Cambridge, MA: JIT press. 1985.

[10] C. L. Seitz, **"The Cosmic Cube,"** in *commun. ACM,"* vol. 28, No. 1, pp. 22-32. Jan. 1985

[11] A. Agarwal, B. H. Lim, D. Kranz and J. Kubiatowicz, **"APRIL: A Procesor Architecture for Multiprocessing,"** in *Proc. 17th Ann. Int. Symp. Comput. Architecture,* May 1990, pp. 104-114.

[12] R. Alverson, D. Callahan, K. Cummings, B. Koblenz, A. Porterfield, and B. Smith, **"The Tera Computer System,"** presented at the *Proc. 1990 Int. Conf. Supercomputing,* June 1990.

[13] T. Feng, **"A Survey of Interconnection Networks,"** in *IEEE comput.,* pp. 12-27, Dec. 1981.

[14] D. Lenoski, J. Laudon, K. Gharachorloo, A. Gupta, J. Hennessy, M. Horowitz, and M. Lam, **"Design of the Stanford DASH Multiprocessor,"** CSL Tech. Tep. 89-403, Stanford Univ., 1989.

[15] F. P. Preparata and J. Vuillemin, **"The Cube-Connected Cycle: A Versatile Network for Parallel Computation,"** in *Proc. 20th Symp. on found. of Comput. Sci.,* 1979, pp. 140-147.

[16] H. Sullivan and T. R. Bashkow, **"A Large Scale, Homogeneous, Fully Distributed Parallel Machine,"** in *Proc. 4th Ann. Int. Symp. Comput. Architecture.,* Mar. 1977, pp. 105-117.

[17] W. J. Dally, **"Performance Analysis of k-ary n-cube Interconnection Networks,"** in *IEEE Trans. Computers,* 39(6): 1990, pp. 775-785.

[18] C. C. Leiserson, **"Fat trees: Universal Networks for Hardware-Efficient Supercomputing,"** in *IEEE Trans. Computers,* 34: 1985, pp. 892-901.

[19] Peter R. Nuth and William J. Dally, **"The J-Machine Network,"** in *IEEE Int. Conf. Comput. Design,* 1992, pp. 420-423.

[20] W. J. Dally and C. L. Seitz,**"The Torus Routing Chip,"** in *Journal of Distributed Computing,* 1(3):187-196, 1986.

[21] Lionel M. Ni and Philip K. McKinley **"A Survey of Wormhole Routing Techniques in Direct Networks,"** in *IEEE computer,* pp. 62-76, Feb. 1993.

[22] Parviz Kermani and Leonard Kleinrock, **"Virtual Cut-Through: A New Computer Communication Switching Technique,"** in *Computer Networks,* vol. 3. no. 4, pp. 267-286, 1979.

[23] W. J. Dally and C. L. Seitz, **"Deadlock-Free Message Routing in Multiprocessor Interconnection Networks,"** in *IEEE trans. comput.,* pp. 547-553, May, 1987.

[24] Christopher J. Glass and Lionel M. Ni, **"The Turn Model for Adaptive Routing,"** in *Proc. 19th Ann. Int. Symp. on Comput. Arch.,* pp. 278-287, 1992.

[25] Christopher J. Glass and Lionel M. Ni, **"Maximally Fully Adaptive Routing in 2D Meshes,"** Technical Report MSU-CPS-ACS-51, Jan. 12, 1992.

[26] A. Giessler, J. Hönle, A. König, E. Pade, **"Free Buffer Allocation - An Investigation by Simulation,"** in *Computer Network 2,* pp. 191-208, 1978.

[27] P. M. Merlin and P. J. Schweitzer, **"Deadlock Avoidance in Store-and-Forward Network-I Store-and-Forward Deadlock,"** in *IEEE Trans. on Comm.,* vol. COM-28, no. 3, March 1980, pp. 345-354.

[28] S. Toueg and J. D. Ullman, **"Deadlock-Free Packer Switching Networks,"** in *Proc. 11th ACM Symp. Theory Comput.,* pp. 89-98, 1979.

[29] Inder S. Gospal, **"Prevention of Store-and-Forward Deadlock in Computer Networks,"** in *IEEE Trans. on Commun.,* vol. COM-33, No. 12, Dec 1985, pp. 1258-1264.

[30] K. D. Gunther, **"Prevention of Deadlock in Packet-Switched Data Transport System,"** in *IEEE Trans. on Commun.,* vol. COM-29, Apr 1985, pp. 512-524.

[31] K. Boyanov, **"Parallel and Distributed Processing '91,"** Elsevier Science Publishers B. V. 1992, pp. 159-177.

[32] H. Sullivan and T. R. Bashkow, **"A Large Scale, Homogeneous, Fully Distributed Parallel Machine,"** in *Proc. 4th Annu. Symp. Comput. Arch.,* vol. 5. pp. 105-124, Mar. 1977.

[33] L. M. Ni and P. K. McKinley, **"A Survey of Routing Techniques in Wormhole Networks,"** in *technical report MSU-COS-ACS-46,* Oct.1991.

[34] M. J. Flynn, **"Some Computer Organizations and Their Effectiveness,"** in *IEEE Trans. Comput. Arch.,* 21(9): 948-960, 1972.

[35] A. J. Frey and G. C. Fox, **"Problem and Approachs for a Teraflop Processor,"** in *ACM Hypercube .....,* 21-25, 1988.

[36] D. A. Patterson and J. L. Hennessy, **"Computer Organization & Design, The Hardware/Software Interface,"** Mogan Kaufmann Publishers, 1993.

[37] R. V. Boppana and S. Chalasani, **"A Comparison of Adaptive Wormhole Routing Algorithms,"** in *Proc. 20th Ann. Int. Symp. on Comput. Arch.,* 351-360, May., 1993.

[38] D. H. Linder and J. C. Harden, **"An Adaptive and Fault Tolerant Wormhole Routing Strategy for $k-$ary $n-$cubes,"** in *IEEE Trans. on Computers,* 40(1): 2-12, 1991.

APPENDICES

APPENDIX A

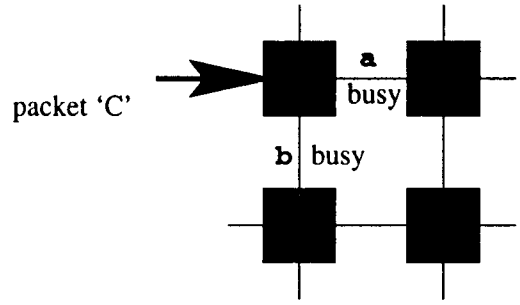


Figure 7.1. The average number of waits for the packet 'C' is $(\sum_{i=1}^{n} i(2n-2i+1))/n^2 = (n+1)(2n+1)/2n$.

Assume that in Figure 7.1, a packet 'A' occupies the channel 'a' and another packet 'B' occupies the channel 'b'. The length of each packet is n flits. Then, each channel has the probability of $(n-1)/n$ to be occupied at the next clock cycle. Therefore, the probability for both channels to be busy at the next clock cycle is $((n-1)/n)^2$. If the packet 'C' is waiting for the channel 'a' or 'b', there are the following wait cases.

Number of waits	Cases	Number of cases
1	$(A = n$ and $B \leq n)$ or $(A < n$ and $B = n)$	$n + (n-1)$
2	$(A = n-1$ and $B \leq n-1)$ or $(A < n-1$ and $B = n-1)$	$(n-1) + (n-2)$
3	$(A = n-2$ and $B \leq n-2)$ or $(A < n-2$ and $B = n-2)$	$(n-2) + (n-3)$
.	.	.
.	.	.
i	$(A = n-(i-1)$ and $B \leq n-(i-1))$ or $(A < n-(i-1)$ and $B = n-(i-1))$	$(n-(i-1)) + (n-i)$ $= 2n - 2i + 1$
.	.	.
.	.	.
n-1	$(A = 2$ and $B \leq 2)$ or $(A < 2$ and $B = 2)$	$2 + 1$
n	$(A = 1$ and $B = 1)$	1

A and B in the above table are the flit numbers which occupy the channels 'a' and 'b', respectively.

For example, when the nth flit of the packet 'A' occupies the channel 'a' or the nth flit of the packet 'B' occupies the channel 'b', the packet 'c' waits 1 clock cycle. Therefore, there are $n + (n - 1)$ cases where the number of waits is 1.

The total number of cases is $\sum_{i=1}^{n} 2n - 2i + 1 = n^2$ and the total number of waits for all cases is $\sum_{i=1}^{n} i(2n - 2i + 1)$. Therefore, the average number of waits of the packet 'C' is

$$(\sum_{i=1}^{n} i(2n - 2i + 1))/n^2 = (n + 1)(2n + 1)/2n.$$

APPENDIX B

The following Figures compare the performances of the XY, the West-First and the Double-Y channel routing algorithms as well as the look-ahead versions of the West-First and the Double-Y channel routing algorithms for different packet sizes 4, 8, 16, 32, and 64 flits, in a 16 $\times$ 16 mesh network. Figure 7.2 - 7.5 compare the latencies, the total number of packets injected into the network, the total number of blocks which the injected packets experienced, and the average number of waits per block, respectively.

These comparisons show that adaptive routing algorithms and the look-ahead scheme do not necessarily improve the network performance in a 16 $\times$ 16 mesh network which has a small number of channel resources.

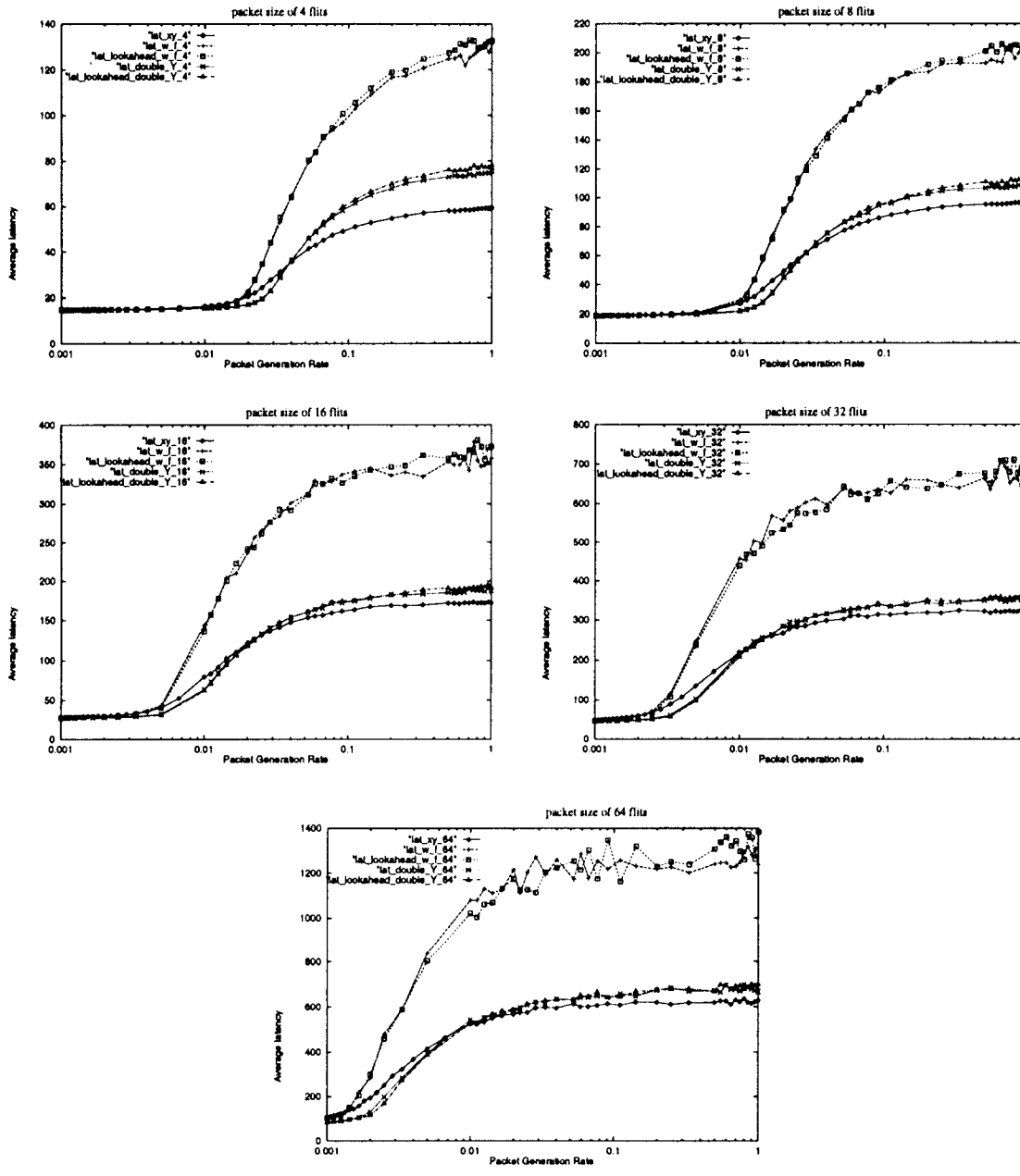


Figure 7.2. Latencies

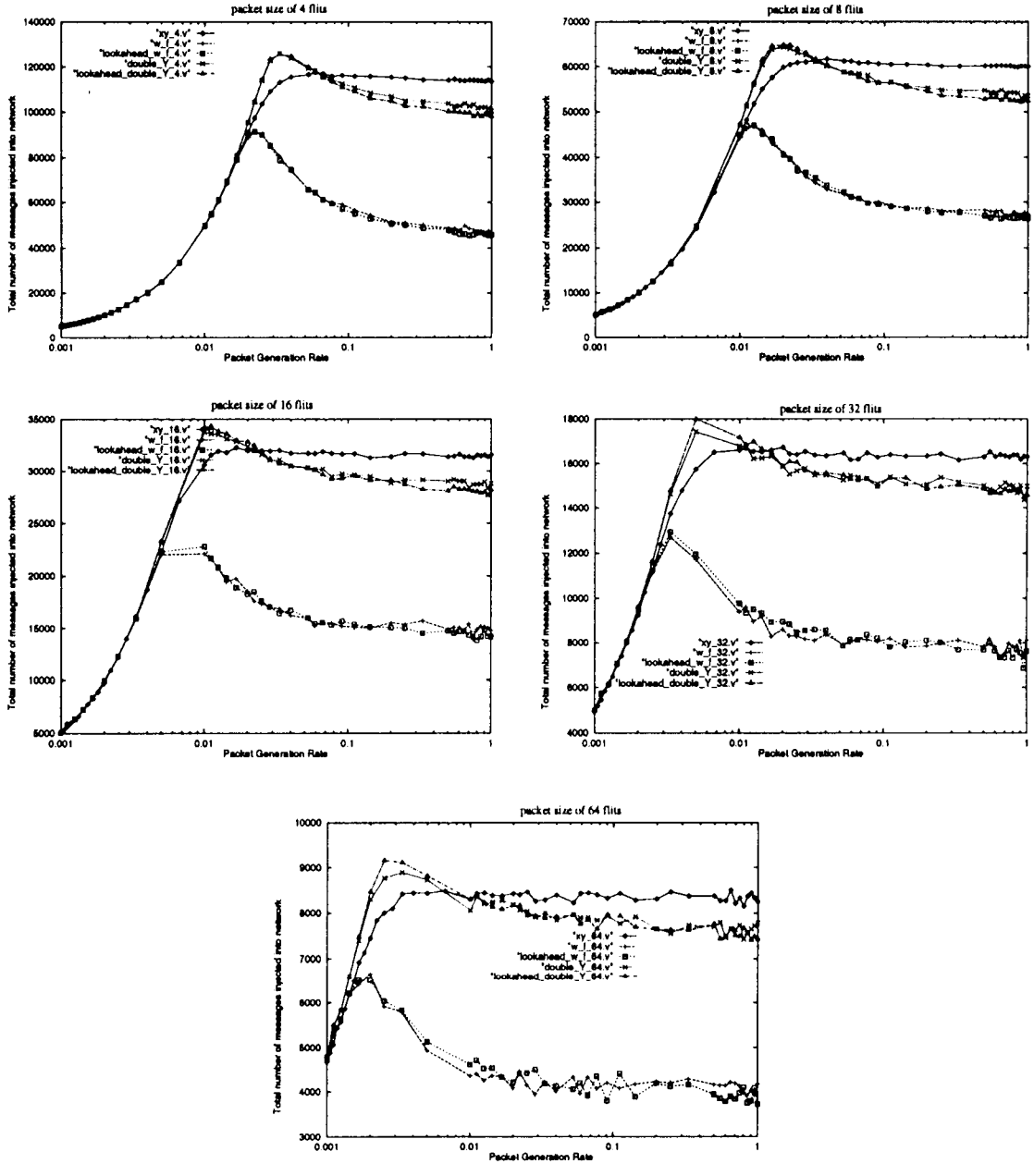


Figure 7.3. Total number of messages injected within 20,000 clock cycles

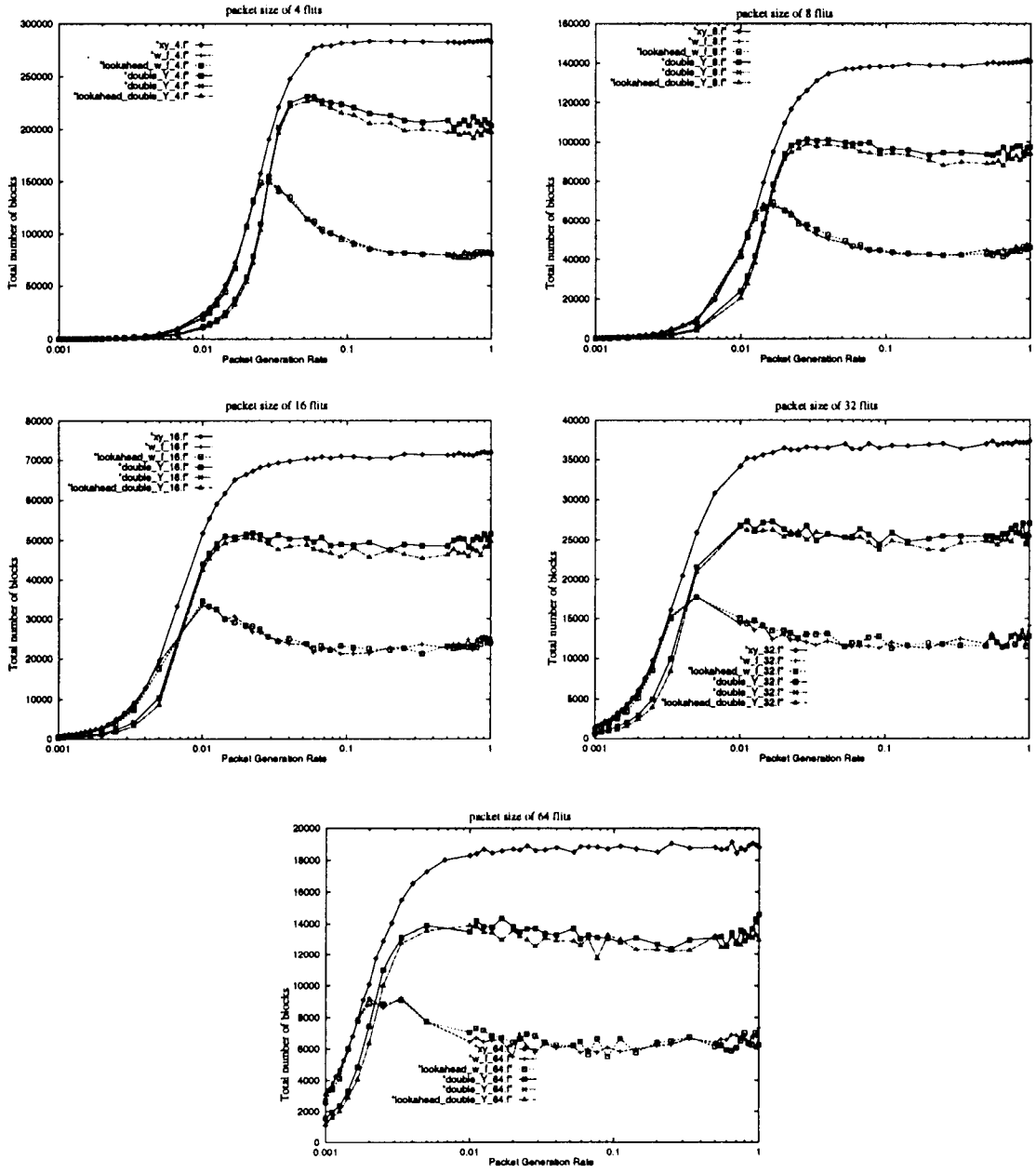


Figure 7.4. Total number of blocks

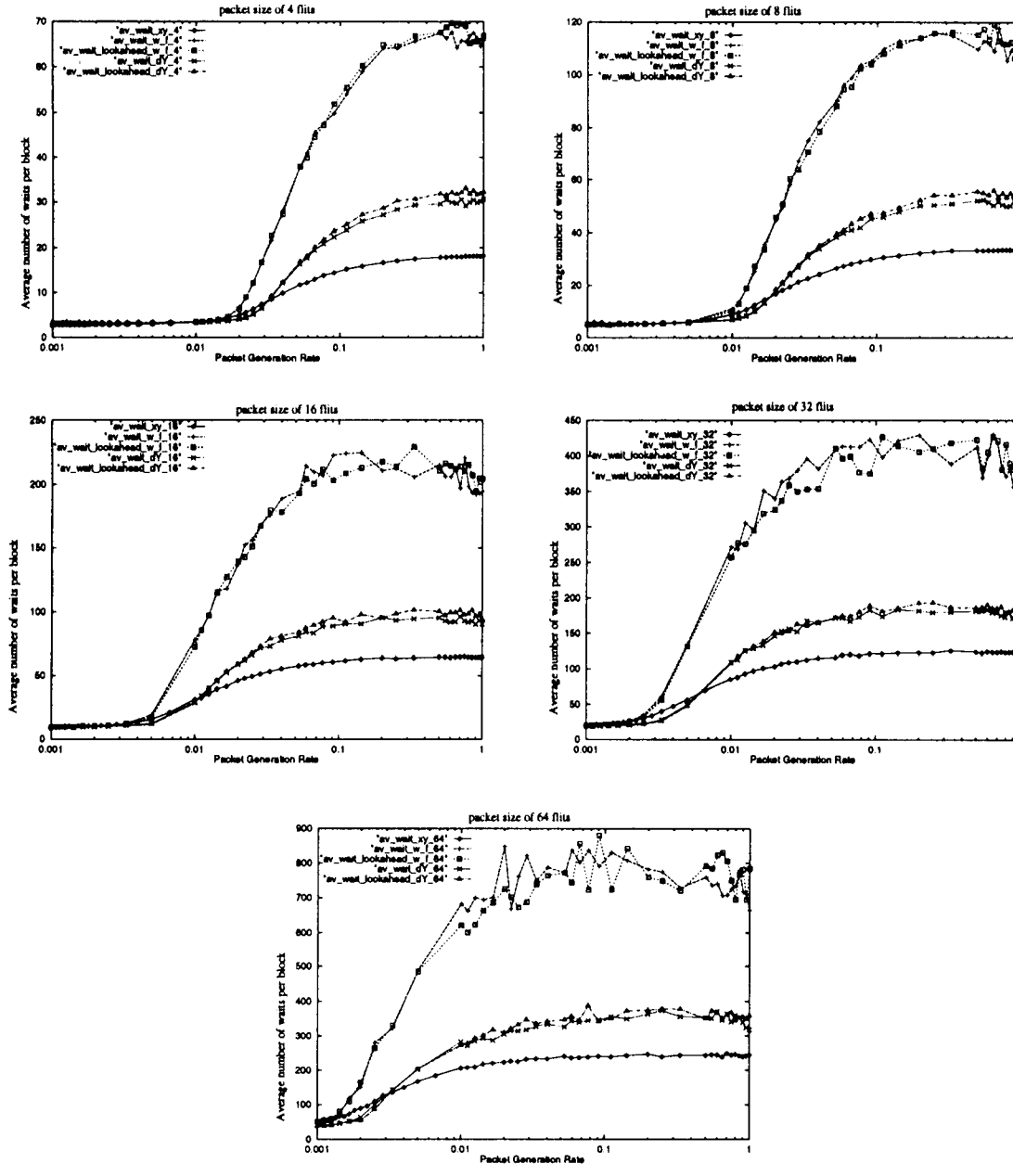


Figure 7.5. Average number of waits per block