1998

Interprocessor communication in wormhole-switched multicomputers

Vara Varavithya
Iowa State University

INFORMATION TO USERS

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

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

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

Oversize materials (e.g., maps, drawings, charts) are reproduced by sectioning the original, beginning at the upper left-hand corner and continuing from left to right in equal sections with small overlaps. Each original is also photographed in one exposure and is included in reduced form at the back of the book.

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

UMI

A Bell & Howell Information Company
300 North Zeeb Road, Ann Arbor MI 48106-1346 USA
313/761-4700 800/521-0600

Interprocessor communication in wormhole-switched multicomputers

by

Vara Varavithya

A dissertation submitted to the graduate faculty

in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY

Major: Computer Engineering

Major Professor: Prasant Mohapatra

Iowa State University

Ames, Iowa

1998

Graduate College
Iowa State University

This is to certify that the Doctoral dissertation of

Vara Varavithya

has met the dissertation requirements of Iowa State University

Signature was redacted for privacy.

Committee Membe

Signature was redacted for privacy.

Committee Member

Signature was redacted for privacy.

Committee Member

Signature was redacted for privacy.

Committee Member

Signature was redacted for privacy.

Committee Member

Signature was redacted for privacy.

Major Professor

Signature was redacted for privacy.

For the Major Program

Signature was redacted for privacy.

For the Graduate College

TABLE OF CONTENTS

LIST OF FIGURES

ACKNOWLEDGMENTS

I express my sincere appreciation to my major professor: Prof. Prasant Mohapatra. His guidances, suggestions, and encouragements during the past five years were precious to my learning process. I thank the committee members, Prof. David Fernandez-Baca, Prof. Sachin Sapatnekar, Prof. Arun Somani, Prof. Johnny Wong, and Prof. Charles Wright for their suggestions and encouragements.

I thank the following graduate students of Iowa State University who made my stay fruitful and enjoyable: Chung-yen Chang, Jatin Upadhyay, Xiaoye Jiang, Sunil Mishra, Sunita Dash, Jatan Shah, Jayesh Sidhiwala, Daphna Nathanson, Kee-Tae Kim, Sudheer Vemulapalli, Fugui Wang, Wei Chen, Xiangping Chen, Hon-Chi Ng, Oyvind Haehre, Jian-Feng Shi, Huibo Hou, Naresh Maheshwari, Nawapak Eua-Anant, Supachai Pathumnakul, Worapa Anothayanon, Pongchai Athikomratanakul, and Sittiporn Uasloong. Special thanks to the system administrators: Jerry, Andrew, and Idarto.

I want to thank my mentor, Prof. Somchai Chatratana, for all his encouragement and advice. Several faculty from King Mongkut's Institute of Technology North Bangkok, such as Vech Vivek, Srawut Kleesuwan, Bundit Fungtammasan, and Somkiat Sucharitpanich, gave me tremendous support and guidance to pursue my graduate study. I express gratitude to Rebecca Matters who offered me a job when I most needed. Thanks to Professors Doyle Wilson and Edward Jones for giving me the opportunities to work with them. As most of my graduate studies were supported by Thai Government, I would like to acknowledge the contributions of Thai citizens.

Above all, I am indebted to my parents, Prof. Chaloem Varavithya and Prof. Wandee Varavithya for their never ending contributions to my well being. Finally, I wish to thank my wife, Jaruwan Varavithya, and son, Pat Varavithya, who endured the most during my study.

ABSTRACT

With the advances in processor technology, the processing speed is increasing at a faster rate than the speed of interconnection networks. The communication subsystem is therefore a bottleneck of the current generation multicomputers. A low-latency communication mechanism is required to improve the system performance. Most of the contemporary multicomputers use wormhole routing mechanism to support their interprocess communication. This dissertation reports techniques for improving performance of wormhole-switched multicomputers. We have developed hardware algorithms to support both unicast and multicast communications.

We have studied several issues that are involved in developing low latency adaptive wormhole routing schemes for two-dimensional meshes. It is observed that along with adaptivity, balanced distribution of traffic has a significant impact on the system performance. Thus, we have developed a new fully adaptive routing algorithm called *positive-first-negative-first* for two-dimensional meshes that distributes the system load uniformly throughout the network.

We present two new multicast frameworks for two-dimensional meshes and multistage interconnection networks (MINs). For meshes, the algorithm uses a wormhole routing mechanism and can send messages to any number of destinations within two start-up communication phases. The algorithm allows some intermediate nodes that are not in the destination set to perform multicast functions. This feature allows flexibility in the multicast path selection and therefore improves the performance. An asynchronous tree-based multicasting (ATBM) framework is developed for MINs in which deadlocks are prevented by serializing the initiations of tree operations that have a potential to create deadlocks. Using the ATBM framework, algorithms are developed for both unidirectional and bidirectional multistage interconnection networks.

The performance of the proposed algorithms is evaluated through simulations. We have considered realistic parameters and have included the associated overheads in our experiments. Simulation results show that the proposed algorithms perform significantly better than the algorithms proposed earlier.

1 INTRODUCTION

Multiprocessor systems have been widely accepted as a solution to solve grand challenge problems in high performance computing. These systems exploit parallelism embedded in the applications and execute multiple operations in parallel. A single task is partitioned into smaller subtasks which are executed concurrently using multiple processors. The speedup is gained from the faster execution rate of the subtasks. Ideally, the speedup should increase linearly with the number of processors. Linear speedup cannot be achieved in practice mainly because of two factors, namely the serial portion in the program and the communication overhead. Amdahl Law [1] shows that the speedup of parallel systems is bounded by the serial portion in the program in which parallelism cannot be exploited. It is difficult to eliminate the serial portion in the program which depends on the characteristics of the application. In parallel program execution, several processors cooperate to execute on a single problem using interprocessor communications, which facilitate information exchange and synchronization.

With the advances in processor technology, the processing speed is increasing at a faster rate. Thus a low latency communication mechanism is required to keep up with the processing speed. The communication subsystems is therefore considered as a bottleneck. The improvement of communication overhead is thus very important for the realization of the high performance computers.

1.1 Multiprocessor Systems

Multiple processor can be classified as *shared memory multiprocessors* and *distributed memory multicomputers* [2]. In shared memory multiprocessors, global memory is accessible by all processors via the interconnection network. To cooperate on a single task, the processors communicate with each other using the shared address space. A distributed memory multicomputer consists of multiple autonomous computers, called *nodes*. The nodes are interconnected by a message passing communication network. Only the local processor is allowed to access the local memory. Communication between nodes is achieved by passing messages between the processing nodes.

Due to the sensitivity of the memory access latency, scalability of the shared memory multiprocessors is limited. Distributed memory multicomputers are more scalable. Compared to shared memory multiprocessors, the software development for the distributed memory multicomputers is more difficult since all the message passing information must be explicitly specified in application programs. Some distributed memory multicomputers, such as, Stanford DASH [3] and Cray 3TD/3TE [4, 5] use distributed shared memory programming paradigm to facilitate the programming effort.

The complexity of the multiprocessor systems prohibits the custom design efforts to be cost-effective. Most of the current generation systems are therefore built using low-cost commodity products which make distributed memory multicomputers even more attractive. In this dissertation, the distributed memory multicomputers will be focused.

1.2 Communication Subsystems

In high performance multicomputers, tasks are executed by a set of intercommunicating nodes or processors. The communication is usually carried out by means of passing messages from one node to another over the interconnection network. The performance of the interprocessor communication depends largely on the network topology, the switching technique, and the path selection technique.

The network topologies can be divided into two classes, *direct* and *indirect* networks. Each switch is directly connected to the processing node in the direct networks. Examples of direct networks include ring, k-ary n-cube, and n-dimensional mesh networks. Many contemporary multicomputer systems adopt low dimensional k-ary n-cube networks due to their low communication latency and high bandwidth. In the indirect networks, only a subset of switches are connected to the processing nodes. The most common form of indirect networks belongs to the class of multistage interconnection networks (MINs) [6, 7]. Both k-ary n-cube networks and MINs are popular topologies adopted in high performance multicomputers.

The three major switching techniques used in communication systems are circuit, store and forward, and cut-though switching [8, 9]. Due to lower latency, cut-through switching is preferred and is widely used in recent multicomputers [10]. The cut-through switching with limited buffers is known as *wormhole switching* [10]. Examples of the experimental systems that have adopted wormhole switching technique include Caltech Mosaic [11], MIT Alewife [12], Stanford DASH [3], MIT J-machine [13], and MIT M-machine [14]. Wormhole switching is also implemented in several commercial systems which include nCUBE [15], Intel Paragon [16], Cray T3D [17], Cray 3TE [5], NEC Cenju-3 [18], Tera Computer [19], IBM SP1/SP2

[20, 21], CM-5 [22], Meiko CS-2 [23], AutoNet [24] and Myrinet [25]. We have considered the wormhole switched interconnection networks in this dissertation.

1.3 Interprocessor Communication

Multiprocessor systems increase their computing speed by performing several computations concurrently. These activities often require coordination and synchronization between processing elements through interprocessor communication which can be either one to one (unicast communication) or within a group of processors (collective communication).

Message passing in multicomputer systems is implemented based on a routing algorithm that determines the path a message follows to reach its destination. Unicast communication is concerned with sending a message from a source node to one destination. Collective communication involves a group of intercommunicating nodes. Several applications can benefit from the collective communication primitives [26, 27]. Some parallel programming languages provide efficient support for these applications. The importance of collective communication is further demonstrated by their inclusion in the message passing interface (MPI) standards [28], which allow users to develop portable message passing applications. Examples of collective operations include broadcast, multicast, gather, scatter, barrier synchronization, and global reduction. Several of these operations can be supported through an efficient implementation of the multicast communication [29, 30].

Multicast communication is concerned with the delivery of a message from one source node to multiple destinations. The multicast services can be considered as basic services for other collective operations such as broadcast and barrier synchronization. In barrier synchronization [31], a multicast operation is performed to distribute a signal to resume the executions. The multicast operation is also important in distributing data to processes. In shared memory systems, multicasting is used for cache invalidations.

Communication latency and network throughput are usually considered as the performance metric. Communication latency is the time elapsed between the initiation of the packet and the reception of the whole packet at the destination. Throughput is the number of the packets delivered per unit time. Minimum communication latency and maximum network throughput are considered as the objectives of network design.

1.4 Scope of the Dissertation

The main objective of this dissertation is to study, analyze, and propose techniques that enhance the performance of wormhole-switched interconnection networks in multicomputers.

In addition to a low latency communication mechanism, both unicast and collective communications need to be efficiently supported to improve the performance of high performance multicomputer systems. In this dissertation, we develop communication techniques that efficiently support both unicast and multicast communication in multicomputers. The path selection techniques for both unicast and multicast communication are designed. The goal of these techniques is to make to design functional, efficient and simple .

A new approach of analyzing adaptive routing algorithms in two dimensional meshes is proposed. We observed that the system performance depends not only on the adaptivity of the algorithm, but also on how evenly the network traffic is distributed. We introduce a new concept called the *region of adaptivity*, the region where messages can be routed using all the available paths. A deadlock-free traffic-balanced fully adaptive routing scheme for two-dimensional (2-D) meshes [32, 33] is developed based on this concept.

The basic low level communication primitives in wormhole switched networks are studied and extended to a set of advanced communication primitives. The set of advanced communication primitives is very important to provide hardware-supported collective communication. From these communication primitives, an algorithm called two-phase multicast (TPM) algorithm for multicasting in two-dimensional mesh networks is proposed [34]. The algorithm requires at most two communication start-up steps to multicast to any member of destinations.

The study on MINs have regained popularity due to the success of IBM SP1/SP2 [20, 21]. Most of the current MIN-based systems support only unicast communication. We propose an asynchronous tree-based multicasting (ATBM) technique for MINs [35]. The deadlock issues in tree-based multicasting in MINs are analyzed first to examine the main cause of deadlocks. The method of deadlock prevention in tree-based multicasting is proposed. An ATBM framework is developed based on this deadlock prevention technique. The ATBM approach is not only simple to implement but also provides good communication performance using minimal overheads. Using the ATBM framework, algorithms are developed for both unidirectional MINs [35] and bidirectional MINs [36].

The performance of the proposed algorithms were evaluated through simulations. We have developed a flit-level wormhole routing simulator. The simulator was designed to support several network topologies and communication primitives. We have considered realistic parameters and have included the associated overheads in our experiments. Extensive simulation results are presented to show the effectiveness of the algorithms.

1.5 Organization of the Dissertation

The rest of this dissertation is organized as follows. Chapter 2 presents an overview of communication subsystem and a review of pertinent literature. An efficient unicast routing algorithms for two-dimensional meshes is described in Chapter 3. Chapter 4 presents the hardware based multicasting techniques for mesh networks. The tree-based multicasting algorithms for MINs are discussed in Chapter 5. Finally, the concluding remarks and future works are enumerated in Chapter 6.

2 INTERPROCESSOR COMMUNICATION SUBSYSTEMS

The performance of multicomputers depends largely on the processor speed as well as the communication subsystems. A considerable amount of research works has been reported in the last decade in the area of interconnection networks. With new communication techniques, new generation communication subsystems incur significantly lower delay in passing massages between any two processors. In this chapter, we present an overview of multicomputer communication subsystems along with the related works. The wormhole routing techniques for both unicast and multicast communication are reviewed. Detailed treatments of interconnection networks can be found in [37]. Surveys on wormhole routing techniques are reported in [10, 38].

2.1 System Model

Figure 2.1 (a) shows a simple model representing multicomputer systems. Each processing node has its own processor, memory unit, and other supporting peripherals. A common bus is used as an interconnection medium between the processor its memory, and the peripherals. A *network interface* unit provides a set of communication services to the processor. The interprocessor communication is invoked by sending a request for communication services to the network interface. Message passing is carried out through a network of interconnected switches.

The basic communication involves two parties, the source node and the destination node. The message passing operation is initiated by the application process of the source node. An abstract model of the message passing networks is shown in Figure 2.1 (b). The application process sends a message through the messaging layer. If there is a packet size limitation, the packetization is performed in the messaging layer. The path selection information together with error checking is encapsulated with the data. The packet is then forwarded to its destination via the physical link(s). At the receiving end, the incoming packet is received and is processed by the destination. The whole message is eventually passed onto the application process at the receiving end using interrupt or polling mechanisms.

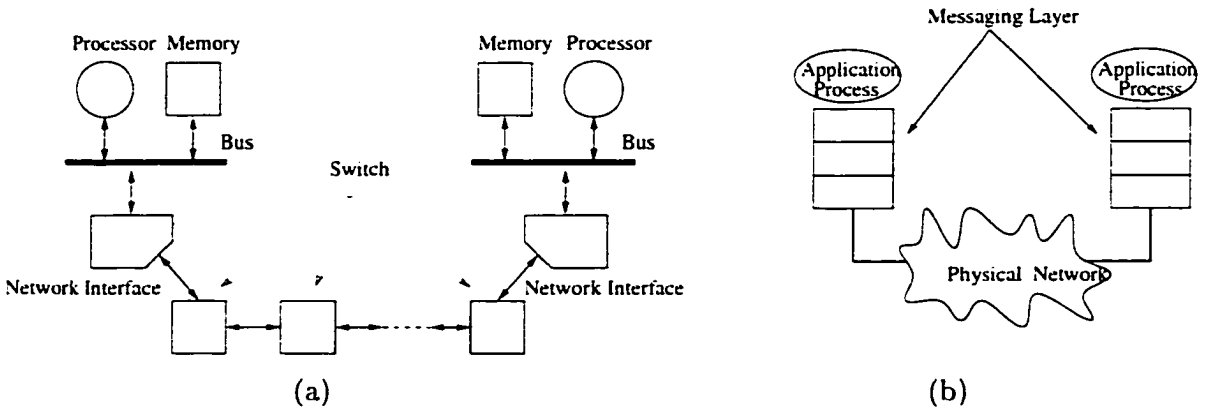


Figure 2.1 Abstraction of communication systems.

Interprocessor communication affects the overall performance of the parallel systems. The performance metrics for the communication subsystems include *communication latency* and *message throughput*. The communication latency is the time elapsed between the initiation of the message and the reception of the entire message at the destination. The message throughput is the number of the messages delivered per unit time. The communication latency T_c is given by following equation,

$$T_c = T_s + T_n + T_r,$$

where T_s, T_n, and T_r represent startup latency, network latency, and reception latency, respectively. The startup latency T_s includes the message processing overhead such as the time incurred in system call and memory copying. The network latency T_n includes the transmission delay plus the blocking delay. Finally, the destination node needs to process the incoming message and transfers it to the application. This additional time reflects on the reception latency.

The operations performed in the software incur much higher delay compared to the operation in the dedicated hardware [39, 40]. The startup and reception time which involve the software operations are therefore the dominant components of the communication latency [41, 42]. To alleviate the software overhead, several light weight protocols have been proposed for multicomputer communication. The objective of these works is to reduce the software overhead in the messaging layer which in turn reduces the startup latency and the reception latency. Examples of these techniques include Illinois Fast Message [43] and Berkeley Active Message [44]. For tightly coupled multicomputer systems, the startup and the reception latency can be reduced by granting low level access to the network interface from the application processes.

Therefore, the application can initiate the message passing operations without involving the operating system. The research works in the area of network interface design can be found in [45, 46, 47].

The transmission latency depends on two factors, propagation delay and channel bandwidth. The propagation delay can be determined from the speed at which a signal travels through the communication medium and the distance covered. The speed of the signal depends on characteristic of the communication link and is limited by the speed of light. In multicomputer systems, the distance between two nodes is usually small and thus the propagation delay is generally negligible. The physical channel bandwidth, denoted as BW, and measured as bits per second, indicates the speed of data transmission. The channel bandwidth is the product of the clock frequency that drives the communication links and the number of physical links per channel.

The communication network resources include switching devices, communication links and buffers. To reduce the cost, the network resources are usually shared among the processing nodes. When a resource contention occurs, only one message can utilize the particular resource while others have to wait. The resource contention delay is referred to as the blocking delay. The blocking probability depends on several factors, such as network topology, buffer size, switching technique, routing algorithm, and network traffic.

2.2 Interconnection Networks

The shared-medium networks, i.e., contention bus (Ethernet), token bus, token ring, FDDI, etc., have been wildly adopted in many applications due to their low cost. However, the shared-medium networks cannot keep up with the requirements of the high performance computing applications. Alternatively, switched networks provide better scalability performance and more aggregated bandwidth. The communication between two processing nodes in switched networks is accomplished by passing a message through one or more switching elements. In multicomputers, two popular classes of interconnection networks are *direct*, and *indirect* networks. The direct and indirect networks are differentiated by the mapping between the processing nodes and the switching elements. Each switch in direct networks is directly connected to the processing node. In the indirect networks, only a subset of switches are connected to the processing nodes.

The direct and indirect networks are typically represented as a graph $G(N, C)$. The vertices of the graph denote the switches (nodes) and the edges of the graph denote the communication links. A formal definition of the interconnection networks is given as follow.

Definition 1.1: An *interconnection network*, IN is a strongly connected graph, $IN = G(N, C)$, where N represents the set of switching elements and C represents the set of communication channels.

2.2.1 Direct Networks

In direct networks, each node consists of an autonomous computer and a switching elements. The node has its own processor, memory, network interface, and other peripherals. The processing node model for direct network is shown in Figure 2.2 (a). The router switch provides communication services to the host processor through the network interface. The node is connected to its neighboring nodes using input and output channels. Non-neighboring nodes can communicate by passing message through intermediate nodes. In Figure 2.2 (a), the unidirectional links are used to represent the communication channels. Bidirectional links can be considered as two bidirectional links in the opposite directions. The connection between input and output channels can be executed by either the node processor or by using dedicated hardware. In most multicomputer systems, the dedicated switch router is used to provide communication services to the processor. The dedicated router switch decouples the node processor from the communication tasks and therefore allows overlap between computation and communication.

Figure 2.2 (b) shows the architecture of a switch. The router switch comprises of a routing control unit, a set of external channels with the associated buffer(s), a set of internal channels, a crossbar interconnection, and a central buffer space. The routing control unit is responsible for the routing calculation, buffer allocation and flow control. Routing decisions and the network resource allocations are performed by the router hardware. A crossbar structure is usually used to establish the connections between input channels and output channels because it allows all possible combination of the input and output connections.

The external channels (input and output channels) are used to connect the router to its neighboring nodes. The internal channels are used as communication links between the router and the local processor. The incoming/outgoing internal channels to/from the router are usually referred as injection/consumption channels. The number of injection/consumption channels implies the number of messages that can be sent/received concurrently by the processor. The number of internal channels is sometimes used to classify the communication system architecture [26]. An *one-port* model refers to a system that has only one pair of internal channels at each node and therefore it can receive or send only one message at any instant of time. In an *all-port* model, the number of injection and consumption channels are equal to the number of input and output channels, respectively. Multiple internal channel pairs can be

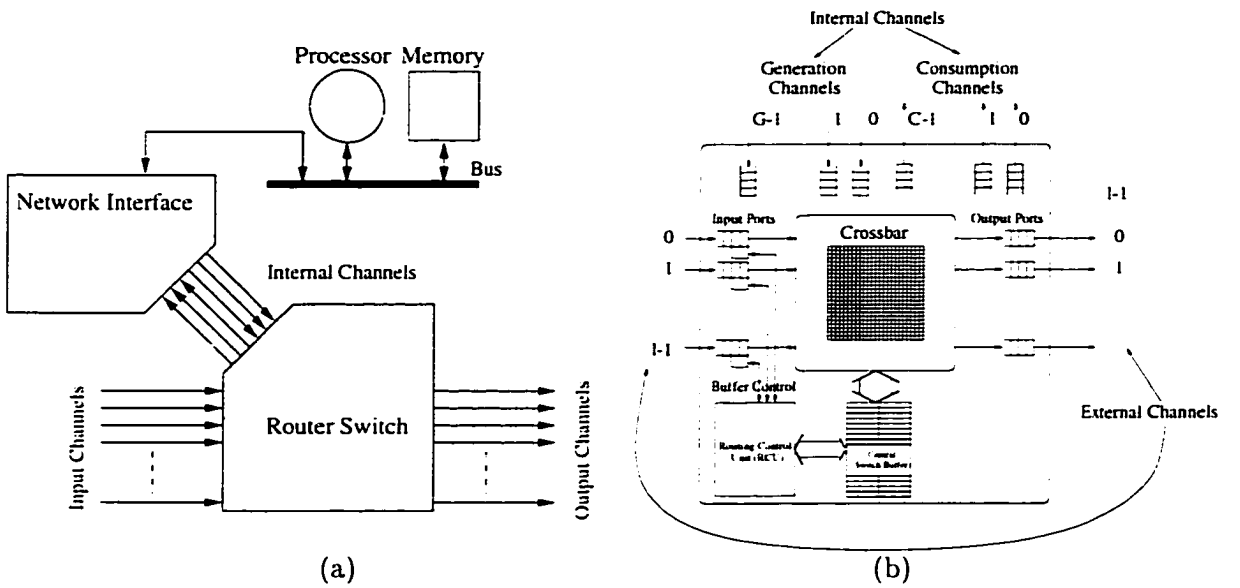


Figure 2.2 Direct network node architecture (a) processing node model (b) router architecture.

used to avoid bottleneck between the processor and the router switch.

Several network parameters are often used in analyzing communication networks and their performances. The *network size N* refers to the number of nodes in the network. The number of input/output channels connected to a node is called the *node degree*. The node degree reflects the degree of connectivity per node. The *network diameter D* describes the maximum shortest distance between any two nodes in the network. Because the maximum permutation between any node is determined by the network diameter, it reflects the communication merit of that particular network. Notice that with equal number of nodes, the high dimensional networks have smaller network diameter than low dimensional networks. For this reason, the high dimensional networks require less number of hops in order to exchange a message between the two remotest nodes. In *regular networks*, all nodes in the network have the same node degree. The direct network is *connected* if every two nodes are reachable from each other. The network is *symmetric* if all nodes have the same properties.

The direct networks can be scaled up by additional nodes and communication links based on the predefined network topology. As the system size increase, not only does the processing capability of the system increases but also the communication bandwidth.

2.2.1.1 Direct Network Topologies

Network topology defines the connectivity of the communication channels of all the nodes in the network. The highest degree of connectivity is found in the completely connected networks where at least one pair of direct communication links exists as a connection between any two nodes. As the network size increases, the cost of complete connection is prohibitive and might not be feasible for practical implementation. The other extreme case of network topology is the linear chain, as shown in Figure 2.3 (a). Figures 2.3 (b), and (c) show two simple topologies, ring and tree, respectively.

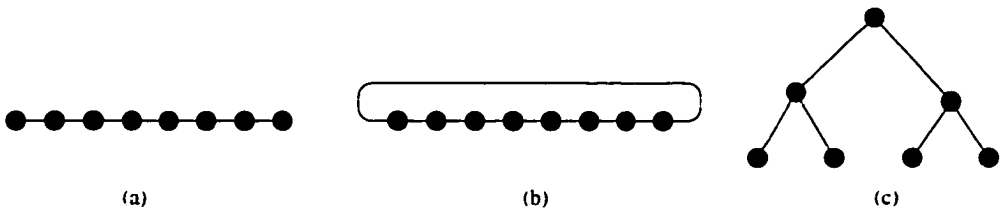


Figure 2.3 Direct network topologies (a) linear chain (b) ring (c) tree.

Several popular topologies are classified as *orthogonal topology*. The nodes in the orthogonal networks can be arranged in the orthogonal n-dimensional space [37]. Due to their scalable properties, the orthogonal n-dimensional topologies are the basic topologies used in most contemporary multicomputers. Two important orthogonal n-dimensional topologies are the *n-dimensional mesh* and *k-ary n-cube* topologies and are defined as follows [38]:

Definition 1.2: An *n-dimensional mesh network* is defined as an interconnection network that has $k_0 \times k_1 \times k_2 \times \ldots \times k_{n-1}$ nodes where k_i is the network radix of dimension i and n is the network dimension. The particular node is identified by the position in each dimension which can be represented by vector $(x_1, x_2, x_3, \ldots, x_n)$. Two nodes, $(x_1, x_2, x_3, \ldots, x_n)$ and $(y_1, y_2, y_3, \ldots, y_n)$ are neighbors to each other if and only if there exists an i such that $x_i = y_i + 1$, and $x_j = y_j$ for all $i \neq j$.

Definition 1.3: An *k-ary n-cube network* is defined as an interconnection network that has n dimensions having k nodes in each dimension. The particular node in *k-ary n-cube* is identified by the position in each dimension which can be represented by vector $(x_1, x_2, x_3, \ldots, x_n)$. Two nodes, $(x_1, x_2, x_3, \ldots, x_n)$ and $(y_1, y_2, y_3, \ldots, y_n)$ are neighbors to each other if and only if there exists an i such that $x_i = (y_i + 1) \ mod \ k$, and $x_j = y_j$ for all $i \neq j$. There are wraparound channels in the k-ary n-cube, with are not present in the n-dimensional mesh networks. If $k = 2$, then every node has n neighbors. If $k > 2$, then every node has $2n$ neighbors.

Figure 2.4 (a) shows an 8×8 two-dimensional mesh network. The node degree in the mesh network depends on its location. The utilization of channels in the center area of the mesh is higher than the channels near the edges. An *8-ary 2-cube* network (two-dimensional torus) is shown in Figure 2.4 (b). Unlike mesh network, the *k-ary n-cube* network is regular and symmetric since all the nodes are identical. The special case of k-ary n-cube is the hypercube when k is equal to 2. Figure 2.4 (c) shows a four-dimensional hypercube network.

One important network parameter is the *bisection width*. The bisection width is defined as the number of communication channel that must be remove to partition the network into two equal subnetworks [48]. The bisection density is the product of the bisection width and the channel width. The bisection density reflects the cost of the network [10]. For equal bisection density, the low dimensional meshes can provide wider channel compared to torus and hypercube. Therefore, the blocking probability is lower and thus incurs low communication latency.

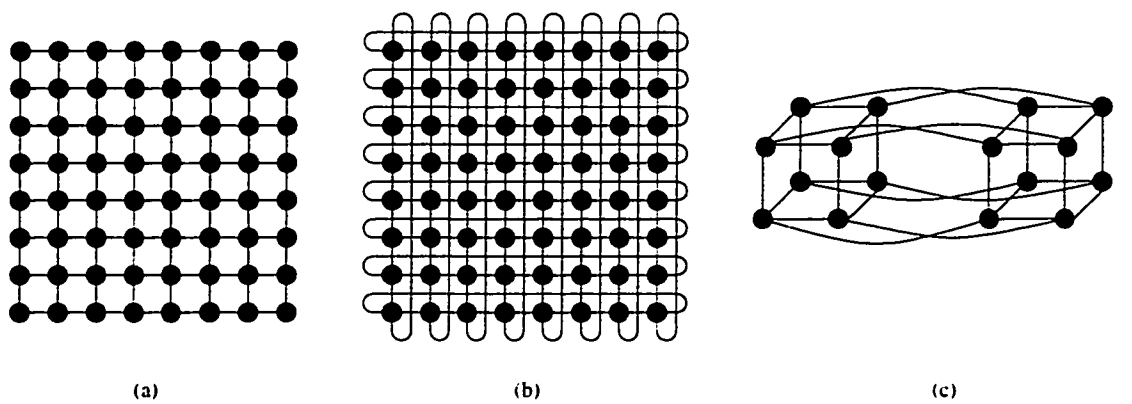


Figure 2.4 Direct network topologies (a) 8×8 2-dimensional mesh (b) *8-ary 2-cube* (c) 4-dimensional hypercube.

Other interconnection topologies proposed for multicomputer interconnection fabrics are cube connected cycles, de Brujin network, star graph, and cayley graphs. The detail treatments of these network topologies can be found in [37]

2.2.2 Indirect Networks

Indirect (switch-based) networks have been extensively studied and adopted as an interconnection fabric for multiprocessor systems [6]. The most common form of indirect networks belongs to the class of multistage interconnection networks (MINs) [6, 7]. Examples of contemporary multiprocessors that use MINs include NEC Cenju-3 [18] and IBM SP1/SP2 [20, 21].

Figure 2.5 displays the architecture of a switch. The switch comprises of a set of input channels with the associated buffer(s), a crossbar interconnection, a central buffer space, and a set of output channels. Unlike the switch router in direct networks, the communication ports to processing nodes are the same as the communication ports to the switches.

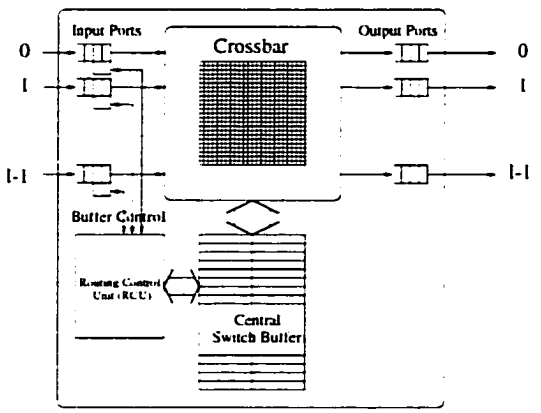


Figure 2.5 Indirect network switch architecture.

Figure 2.6 shows multiprocessor systems interconnected through MINs. The communication links can be either unidirectional or bidirectional. The bidirectional communication links can be viewed as two unidirectional communication links connected in opposite directions. A unidirectional MIN (UNI-MIN) system is shown in Figure 2.6 (a). The processing nodes are connected to the input ports of the MIN (stage zero). A message sent by a node is forwarded through the switches and reaches the receiving node via wrapped around communication links. A MIN using bidirectional communication links, called bidirectional MIN (BI-MIN), is shown in Figure 2.6 (b). The processor's communication channels are connected to the BI-MIN (stage zero) using bidirectional communication links. The right hand side ports are used for scalability purpose. The BI-MIN system can be scaled up by increasing the number of stages or by connecting another set of processing nodes to the system (as shown in Figure 2.6 (c)).

A MIN using $b \times b$ switches with n stages and r rows has $N = b^n$ connection ports on each side. A switch at row i and stage j is labeled as (i, j) where $\{i \in (0, 1, \ldots, r-1)\}$ and $\{j \in (0, 1, \ldots, n-1)\}$.

The representation of the MIN topology using $b \times b$ switches can be expressed as

$$C_0(N)G_0(N/b)C_1(N)G_1(N/b)\ldots C_{n-1}(N)G_{n-1}(N/b)C_n(N),$$

where C_i is the i^{th} connection and G_i is the i^{th} stage. The connection C_i specifies the connection from the right hand side of the switches at the stage G_{i-1} to the left hand side of the switches at the stage G_i. The connection C_i is derived from the topology permutation.

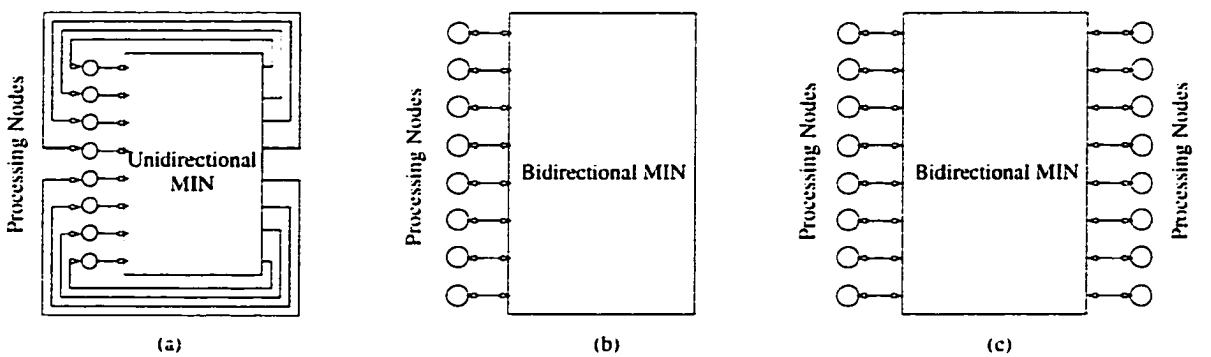


Figure 2.6 MIN systems (a) unidirectional MIN (b) bidirectional MIN (c) two-sided bidirectional MIN.

Four common connections, perfect shuffle, baseline, butterfly, and cube, are defined as follow [2, 6, 37].

The perfect shuffle connection ρ^b is defined as

$$\rho^b(x_{n-1}x_{n-2}\dots x_1x_0) = x_{n-2}\dots x_1x_0x_{n-1}.$$

where $[x_i \in (0, 1, \dots, b-1)]$.

The butterfly connection β_i^b is defined as

$$\beta_i^b(x_{n-1}\dots x_{i+1}x_ix_{i-1}\dots x_1x_0) = x_{n-1}\dots x_{i+1}x_0x_{i-1}\dots x_1x_i,$$

where $[x_i \in (0, 1, \dots, b-1)]$.

The baseline connection δ_i^b is defined as

$$\delta_i^b(x_{n-1}\dots x_{i+1}x_ix_{i-1}\dots x_0) = x_{n-1}\dots x_{i+1}x_0x_ix_{i-1}\dots x_1,$$

where $[x_i \in (0, 1, \dots, b-1)]$.

The cube connection E_i is defined as

$$E_i(x_{n-1}\dots x_{i+1}x_ix_{i-1}\dots x_1x_0) = x_{n-1}\dots x_{i+1}\bar{x}_ix_{i-1}\dots x_0,$$

where $[x_i \in (0, 1, \dots, b-1)]$.

The node addresses are represented as $[a_{n-1}\dots a_1a_0]$ where $\{a_i \in (0, 1, \dots, b-1)\}$. The source and destination node addresses are represented by $[s_{n-1}\dots s_1s_0]$ and $[d_{n-1}\dots d_1d_0]$, respectively. Figure 2.7 (a) shows an 8-node unidirectional baseline MIN. A butterfly MIN with 8 nodes using 2×2 switches is shown in Figure 2.7 (b). Figure 2.7 (c) shows a 16 nodes butterfly MIN constructed using 4×4 switches. It has been shown in [49] that the structure of the BI-MIN is equivalent to a fat tree topology.

2.3 Switching Techniques

Switching techniques specify the connection activities performed by the switching elements when a message is received at the input port. The message enters the network from the source node and is forwarded through a series of switches towards its destination. Three major switching techniques used in multicomputers are circuit switching, packet switching, and cut-through/wormhole switching. Historically, circuit switching technique has been used to link between two end points in telecommunication networks. Packet switching was first developed in the area of data communication and had been implemented in the early generation multicomputers. Virtual cut-through and wormhole switching techniques compromise the advantages in both the circuit switching and the packet switching. In multicomputer communication networks, the cut-through switching approach is a promising technology because of its low latency and buffer requirements.

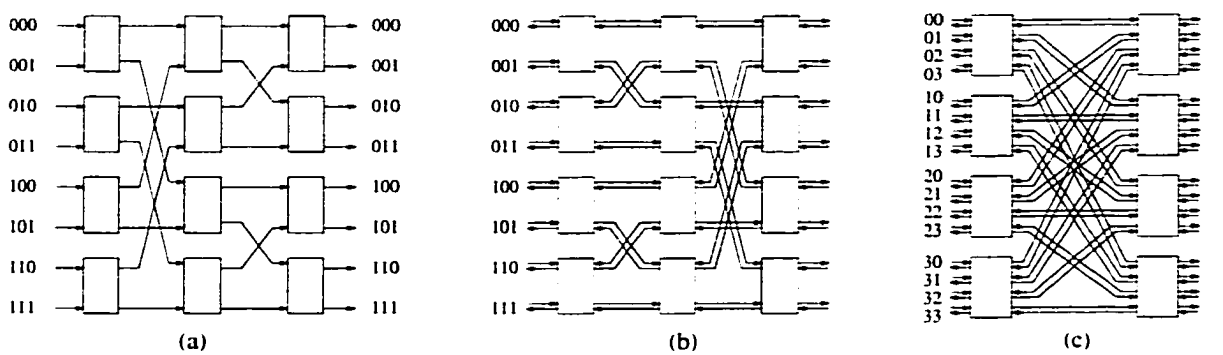


Figure 2.7 MIN topologies (a) 8-node baseline UNI-MIN using 2 × 2 switches (b) 8-node butterfly BI-MIN using 2 × 2 switches (c) 16-node butterfly BI-MIN using 4 × 4 switches.

Two application processes intercommunicate by passing messages. In systems where the maximum size of packets is limited, messages are broken down to smaller packets. A packet is defined as the smallest unit of information that contains routing and sequencing information. At the lowest level, the packet is divided into flow control units, called *flits*. A flit is the smallest data unit that the hardware communication unit will process at a time. The hierarchy of data units is shown in Figure 2.8.

2.3.1 Circuit Switching

In circuit switching, a set of physical channels is dedicated to the connection. Three phases are involved in the circuit switching communication. Prior to the transmission of any data,

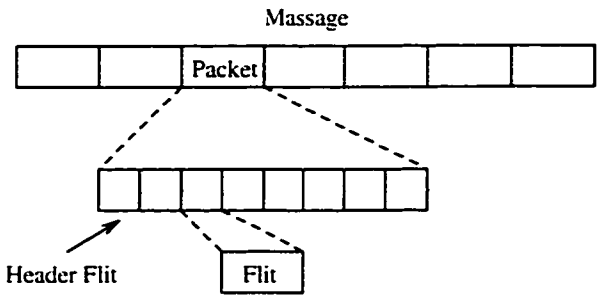


Figure 2.8 Message format.

a dedicated path from source to destination is created. Data transfer can be initiated after a dedicated circuit is established. Once the dedicated circuit is set up, the message transfer delay is only the combination of propagation delay through channels. Hence, the latency in the circuit switching is bounded after communication circuit is established. When the data transfer phase is completed, the circuit is terminated. No buffering is required in the communication path. The time for circuit establishment and termination is a major overhead in the circuit switching. The reserved channels may not be used during these processes which results in lower channel utilization. These disadvantages limit the acceptance of circuit switching as a candidate for implementation in multicomputer networks.

2.3.2 Packet Switching

In packet switching [8], a message is fragmented into smaller chunks of information, called *packets*. Each packet contains the routing information and is independently routed towards its destination. The entire packet is stored in every intermediate node before it is forwarded to the next node in its path. The main advantage of packet switching is that the channel resource is occupied only when a packet is actually transferred. Since the packet contains the routing information, different paths can be selected upon encountering the network congestion or the faulty nodes. On the other hand, additional overhead is required for encapsulating the routing information in every packet and reassembling the packets at the destination. The major drawback of the packet switching is the delay incurred in transmitting and receiving an entire packet from one node to other node. The network latency is thus directly proportional to the number of hops in the path, which results in higher latency as the distance increases. Since network latency is very sensitive to the number of hops in the packet switching, network topologies with small diameter such as hypercube is preferred over a large diameter network.

2.3.3 Cut-Through/Wormhole Switching

The virtual cut-through concept was introduced by Kermani and Kleinrock [9]. When a packet arrives at the intermediate node, a free outgoing channel is selected and a packet is forwarded before an entire packet is received at the current intermediate node. The communication latency is dramatically reduced because multiple communication links are utilized in parallel using the pipelining technique. If no free outgoing channel is available, the packet is stored at the intermediate node until an outgoing channel is freed. The virtual cut through router must have buffer space to store all incoming messages. As the number of incoming message cannot be predetermined, the buffer space required by virtual cut-through might be excessively high and may not be practical from an implementation viewpoint.

The wormhole switching technique was first proposed as a low latency switching technique in the Torus Routing Chip [11]. In wormhole switching [10, 50], a packet is divided into small flits. Only the header flit(s) contains the routing information. Using the same concept as that of virtual cut-through, the data flits follow the header flit in a pipelined fashion. If the header flit cannot be forwarded due to resource contention, all following flits are blocked along the network. The packet remains in the network until blocked resources become available. Similar to virtual cut-through, the pipelined flow control dramatically reduces the communication latency. Compared to virtual cut-through, only a small buffer is required at each input port. The small buffer requirement has a strong effect on the router speed since it is possible to implement a router on a single chip.

Experimental results in [10] show that the wormhole switching is almost independent from distance assuming no resource contention. The insensitivity to distance characteristic is not true in congested networks. A major drawback of wormhole routing arises from the network resource contentions. The probability of blocking is greater in the wormhole routing because blocked packets continue to remain in the network and therefore freezes up some network resources. Figure 2.9 shows the example of chained blocking. When packet A is blocked, packet B will wait for the channel occupied by packet A. A blocked packet can create a blocking chain to others packets along the network.

The blocking probability can be reduced by increasing the buffer size at the switch. The smallest buffer requirements of wormhole switching is one flit buffer for each direction. If the buffer space is equal to the packet size, the worst case performance of buffered wormhole switching is similar to the packet switching. No other packet is allowed to use any empty buffer in the same queue, if there are flit(s) belonging to one packet in the buffer queue. The router has no information to distinguish between data flits of different packets. A packet header flit must wait until all buffers in the queue of the next node are empty before it can be passed to

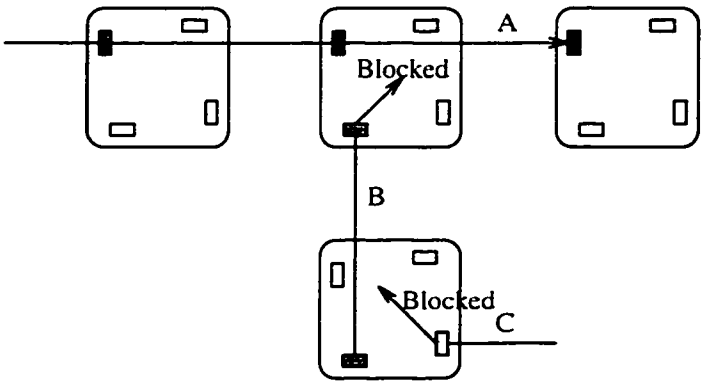


Figure 2.9 Blocking in wormhole routing.

that node. The impact of the packetization in wormhole-switched networks is reported in [51]

Wire density is a design constraint in VLSI implementations. The network wire bisection width reflects the idea of the required wire density needed to construct the interconnection network. With wormhole switching, Dally [48] shows that the low-dimension networks have lower latency than the high-dimension networks under the assumption of equal wire bisection width. Since the network latency is almost insensitive to the path distance in a wormhole switched network, it is better to have wider channel widths than smaller network diameters. In this dissertation, we will focus our discussions on the wormhole-switched networks.

Other switching techniques using cut-through concept include mad postman switching, pipelined circuit switching, and scouting switching. The details of these switching techniques are presented in [37]. The studies in incorporate multiple switching techniques in the same networks can be found in [52, 53].

2.4 Virtual Channels

Figure 2.10 (a) shows the conventional interconnection between two switches (without virtual channel). The physical channels are directly connected from the output ports to the input ports between neighboring nodes. Since each switch operates independently, associated control lines are required for the handshaking protocol. The performance of wormhole switched networks can be significantly improved using the concept of *virtual channel* [54]. The virtual channel is a flow control technique that allows packet flits to be time-multiplexed on the same physical channel. The virtual channel is implemented by adding extra flit buffers and control circuits associated with each physical channel of the node. These buffers act as small queues which decouple buffers from channels. The buffer allocation function can be implemented

using random, round-robin, FIFO or priority based schemes. The FIFO and priority schemes are commonly used to prevent starvation. Virtual channel allocation can be done randomly or may depend on some schemes determined by a routing function. Connection between two switches with two virtual channels per physical channel is shown in Figure 2.10 (b).

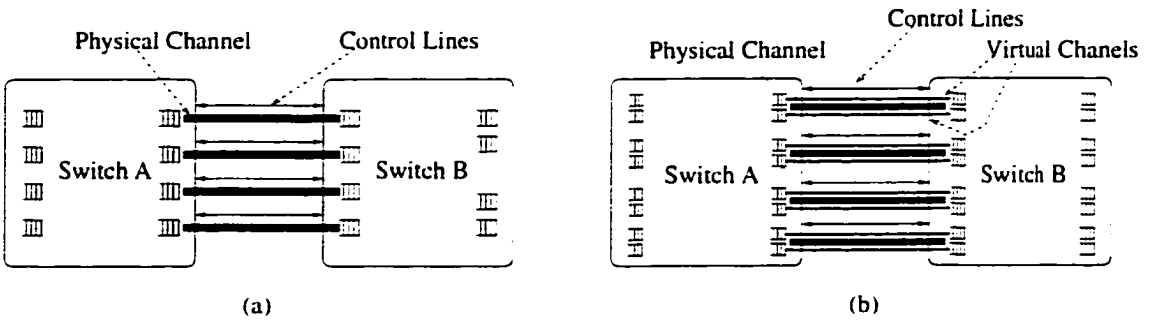


Figure 2.10 Physical connection between two switches (a) no virtual channel (b) two virtual channels sharing the same physical channel.

Virtual channels can improve the blocking behavior of the wormhole switched network. As shown in Figure 2.11, a single physical channel is shared by two virtual channels. Two flit buffers are associated with each physical channel. Packet A uses the upper virtual channel and is blocked at node 3. Blocked packet A then relinquishes the control of the channel from node 1 to node 2 so that packet B can still be forwarded. The virtual channel can be used to divide the physical network into logical abstraction of virtual networks. The concept of virtual networks has been used to facilitate several routing techniques such as multiple priority traffic in the network or prevent deadlock cycle.

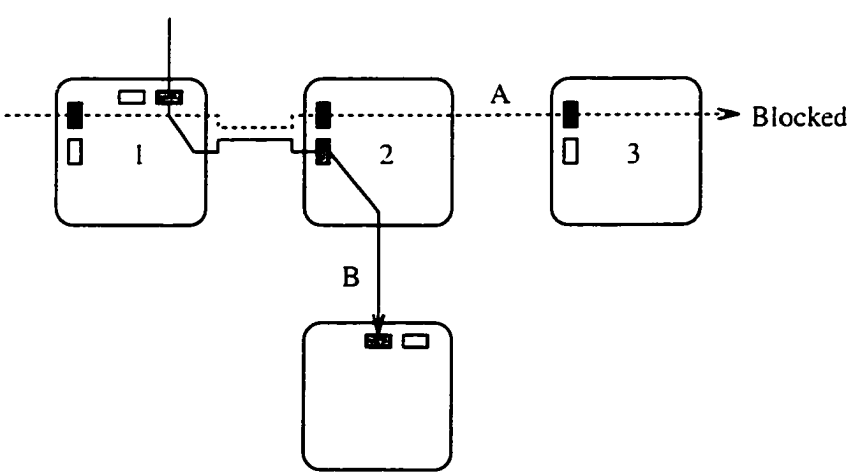


Figure 2.11 Avoid blocking using virtual channels.

The virtual channel concept is used in several aspects including performance improvement [54, 55], fault tolerance [56, 57, 58], and deadlock avoidance [57, 59, 60, 61, 62]. The implementation of the virtual channels increases the complexity of the switch router in term of additional storage space and logic. Chien [63, 64] showed that the implementation of virtual channels can considerably decrease the clock speed of the switch. Therefore the number of virtual channels should be kept small (two to four). Examples of commercial multicomputer systems implementing virtual channels include Cray T3D/T3E [5, 17].

2.5 Starvation, Livelock, and Deadlock Issues

In wormhole-switched networks, a message is transfered through a series of switches towards its destination. Buffers and communication channels are considered as common network resources that are shared among the processing nodes. Resource sharing could create starvation, livelock, and deadlock. Starvation is a situation in which access to the resources is never granted to a specific message. Starvation is usually resolved using network resource allocation policy. Round-robin and first-come first-serve policies are common techniques used to solve the starvation problem. Livelock is a situation in which a packet can be routed without ever arriving at the destination. One way to prevent livelock is limit the ratio of misroute steps to progress steps [59]. A probabilistic technique used to achieve livelock-free is proposed in [65] where the minimal path is given preference over non-minimal path. Livelock problem does not occur in the minimal routing because in every progress, a packet is forwarded towards the destination.

The major drawback of wormhole switching is that the messages occupy network resources along their paths. The request and hold of network resources could lead to a deadlock configuration. Deadlock in an interconnection network occurs when a set of packets cannot be forwarded because of the cyclic waiting of the network resources. A deadlock configuration in a 2-dimensional mesh network is shown in Figure 2.12. Packet A, B, C and D are waiting for the availability of buffers in circular fashion. None of the messages can progress in the waiting cycle. Deadlock is a disastrous state in multicomputers because the communication can never be completed. The deadlock configurations associated with multicast communication will be discussed in details later in Chapters 4 and 5 for mesh networks and MINs, respectively.

There are two major solutions for the deadlock problems, deadlock recovery and deadlock avoidance. The deadlock recovery approach consists of two steps of operations. First, the deadlock detection mechanism is required to identify deadlock configuration. After the deadlock configurations is detected, recovery is performed in the second step. Time-out mechanism

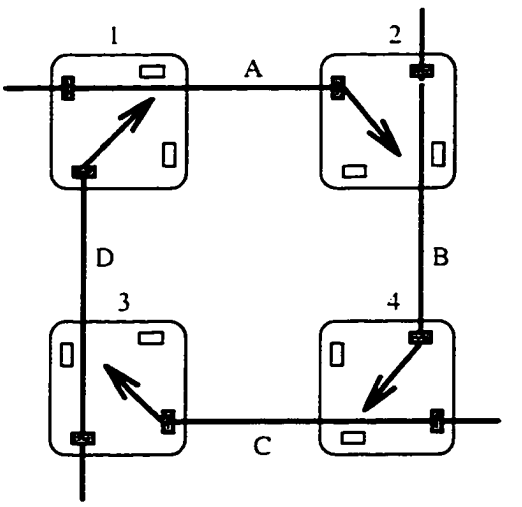


Figure 2.12 Deadlock situation in wormhole routing network.

can be used to identify the deadlock scenario [58, 66, 67]. A message is marked as a deadlock message after the blocking time exceed some threshold values. However, this approach is not very reliable since false detection can happen especially under the heavy traffic. In addition to the time-out mechanism, the chained blocking information can be used to reduce the probability of false detection, as proposed in [68]. A feasible way to recover from deadlock is the process of abort and retry [67]. In abort and retry, the messages detected as deadlock messages are discarded so that other packets can be forwarded. A discarded packet will be later retransmitted by the source node. The overhead in retransmission process rapidly increases under high traffic. The use of a deadlock recovery channel is another approach of deadlock recovery [66, 58]. Once the message is detected as a deadlock message, it is allowed to exclusively use the recovery channels to routed towards its destination. Since only one message is allowed to enter the recovery channels, the recovery channel is deadlock-free.

For deadlock avoidance, the network resources are allocated such that deadlock configurations cannot occur. In packet switching networks, a structure buffer pool method [69] is proposed to prevent the deadlock. This method divides the packet buffers in each node into classes and restricts the order of the buffer allocation. The structure buffer pool can not be used in the wormhole routing because no routing information or packet identification is contained in the data flits. One approach of deadlock avoidance in wormhole networks is by applying labels to the network resources. The network resources are allocated in a particular order to avoid the cyclic waiting. Incorporation of routing restrictions and virtual channel allocations are the most commonly used techniques to prevent deadlock in wormhole-switched

networks. Several research works has been accomplished in developing deadlock-free theory. The sufficient conditions for the deadlock-free networks in several environments were proposed in [50, 62, 70, 71]. Necessary and sufficient conditions for deadlock-free routing were reported in [72, 73, 74, 75]. The details of deadlock avoidance techniques for mesh networks and MINs will be discussed in Chapters 3 and 5, respectively.

2.6 Path Selection Techniques

Path selection technique concerns with selecting a path from the source node to the destination node. The term "routing algorithm" usually refers to the algorithm that is used to select the routing path. In this section, the overview of the path selection techniques for unicast and multicast communication are presented.

2.6.1 Unicast Path Selection Techniques

In unicast communications, the path from a source node to a destination node is chosen by the routing algorithm. Several ways to classify routing algorithms have been adopted. Routing algorithms can be classified by the location at which the routing decision is made, degree of freedom in routing, and the path distance. The routing algorithm can be classified as *source* routing or *distributed* routing. In source routing, the path for message routing is decided at the source node. The path information is encoded in the message header. Since the message has to carry information about the entire path, the message header could be excessively large. Several switch-based networks with a small path distance have adopted source routing, i.e., MINs [6] and Myrinet [25]. In the distributed routing, the intermediate router switches along the routing path examine the message header of the incoming message.

If the path between every pair of source and destination is fixed, the algorithm is called *deterministic*. For better system performance, it is preferable that the algorithm adapts itself to the traffic congestion by providing alternate paths. Adaptive routing algorithms are classified as *partially adaptive* or *fully adaptive*. Partially adaptive routing algorithms use only a subset of the available physical paths between the source and the destination. Turn model [76, 77, 78], direction restriction model [79], and planar-adaptive routing [80, 81] are examples of partially adaptive algorithms for mesh networks. Examples of fully adaptive algorithms include the routing schemes for mesh networks proposed by Linder and Harden [57], Su and Shin [60], Boura and Das [61], Duato [62], and Schwiebert and Jayasimha [82]. The formal definition of the routing algorithm is described as follow.

Definition 4: A *Routing Algorithm* $R : N \times N \rightarrow \rho(C)$, where $\rho(C)$ is the power set of C, given a set of output channels to send a message from current node s_i to destination node d_j. A routing algorithm is *connected* if and only if a path between any pair of s_i and d_j in the interconnection network can be created using the set of channels given by R.

Path distance is another way of classifying the routing algorithms. In *minimal routing*, only the shortest path is taken so that in every progress, a packet is moved closer to its destination. If there is no deadlock, the minimal routing algorithm guarantees that a packet will finally reach its destination. Figure 2.13 (a) shows the shortest path taken by a packet in the mesh network. In *non-minimal routing*, misrouting or derouting is allowed. The non-minimal paths can be used to detour around the congested area or the permanently faulty nodes. Since the non-minimal routing allows a packet to be moved away from its destination, livelock prevention is required to ensure that packet will finally reach its destination. An interesting research work on the non-minimal routing can be found in [83, 59]. The non-minimal routing is illustrated in Figure 2.13 (b).

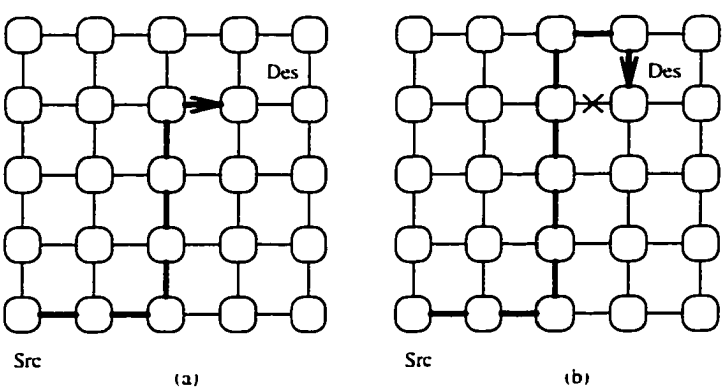


Figure 2.13 Routing algorithm (a) minimal (b) non-minimal.

The unicast routing algorithms for mesh and MINs will be revisited in Chapter 3 and 5, respectively. Comprehensive survey on the unicast routing algorithms in the direct networks is reported in [38]. The studies on routing in MINs can be found in [6, 54, 49].

2.6.2 Multicast Path Selection Techniques

The purpose of multicast communication is to distribute the same information to a group of processing nodes. The multicast path selection algorithm determines the path(s) from the source to all its destinations. Depending on the techniques used in multicasting, several paths can be selected to forward the multicast messages. If the communication involves a group of processors, as in collective communication, the communication latency is defined as the

message initiation time to the time until all operations are completed. The performance of multicast communications is measured in terms of its latency in delivering a message to all the destinations [42].

Multicast communication can be carried out by sending one message to each of the destinations. If the multicast operation involves K destination nodes, the unicast function, `send()`, is called K times for each destination. As shown in Figure 2.14 (a), the source node A send multicast messages to seven destination nodes $B - H$. The total of seven messages for each destination are sent by the source node. This method performs poorly not only because it requires a large amount of network resources but also because it involves several start-up phases. Efficient software-based multicast algorithms utilizing divide and conquer strategy have been proposed by several researchers [84, 85, 86, 87, 88, 89, 90, 91]. The software-based multicasting uses underling unicast communication and therefore does not require any hardware modification. These schemes try reduce the number of start-up phases by allowing some destinations to act like source nodes after they receive the message. The total number of start-up phases is reduced because more number of destinations can be covered as the multicast operations progress. Multicast operations based on binomial tree structure is shown in Figure 2.14 (b). The source node A sends the message to the destination node E in the first step. Nodes A and E distribute the multicast messages to node G and C in the second steps. The rest of the destination nodes are covered in the third communication steps. For the binomial multicast structure, $\lceil \log_2(K+1) \rceil$ communication phases is required for completing the multicast operation in K destination. Park *et al.* have proposed software-based multinomial multicast algorithm [92]. The number of communication phases is further reduced by fine tuning the multicast tree based on the interconnection network parameters.

As previously mentioned, the communication latency comprises of start-up latency and network latency. The start-up latency is the time required to start a message, which involves destination encoding and formating the message. The network latency is a combination of propagation delay, router delay, and contention delay. The communication latency is dominated by the start-up latency which is in order of 1 μs to 20 μs in the current generation systems [93]. For example, consider a 16 × 16 mesh network having the following parameters: 3.2 Gbit/sec link bandwidth, 64 bits/flit, 1024 bits messages. The average number of hops is equal to 14 using the uniform traffic pattern. The average network latency will be $(\frac{64}{(3.2\times10^9)}) \times (14+15) = 580ns$. If we assume $1\mu s$ start-up time, the start-up latency accounts for 63.29% of the total communication latency. In multicast communications, if an operation consists of several start-up phases, the overall communication latency may become significantly high.

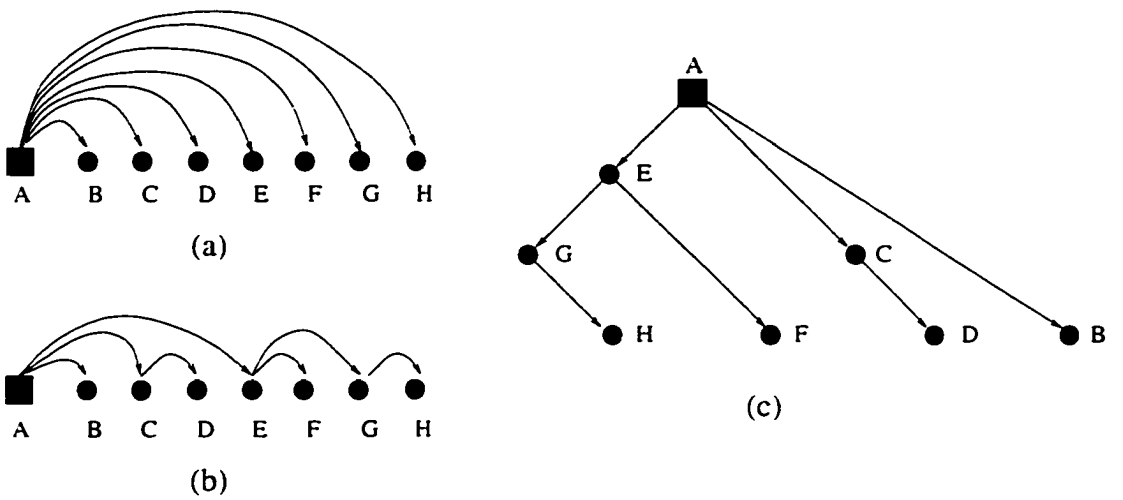


Figure 2.14 Software-based multicast (a) unicast-based multicast (b) efficient unicast-based multicast (c) unicast-based multicast tree.

To further reduce the communication latency, the multicast operations need to be supported by the hardware. The importance of hardware supported multicasting was reported by Ni [29]. As dedicated hardware usually performs better than software, the start-up delay can be significantly reduced. Furthermore, the hardware-supported multicast can be designed such that it is possible to utilize the common paths in forwarding a single multicast message to cover more than one destination. The number of start-up phases is therefore significantly reduced. Examples of useful hardware supports for multicast operations are *absorb-and-forward* and *replication.*

The absorb-and-forward operation is the additional function provided by a hardware router that allows a message to be concurrently forwarded and stored. Figure 2.15 (a) shows the multicast absorb-and-forward operations for the multicast message destined to nodes A, C, and D. The amount of traffic in the network is less compared to the software-based scheme since the destination nodes $[A, C, D]$ receive information from the same multicast message. When the message reaches an intermediate destination, i.e., node A and C, the destination addresses in the header are updated by the switch router. The header flit is then forwarded to the next node. The router keeps absorbing and forwarding the data flits in a pipelined fashion. Finally, the message is eventually consumed at the last destination in its multicast path. In replication, the switch has a capability to replicate the incoming flit and forward them to different directions. The replicate mechanism is shown in Figure 2.15 (b). Nodes E and F replicate an incoming multicast message and forward to different destinations. The multicast message that cover more than one destination is usually referred to as multidestination messages [41]. To support

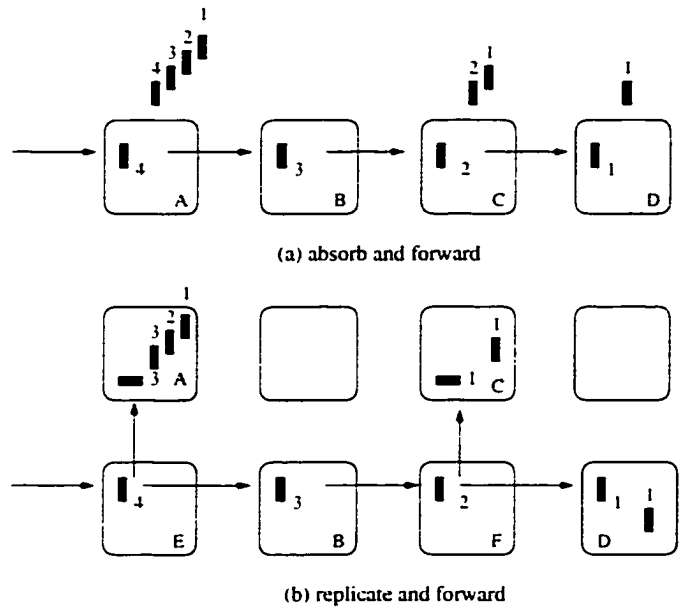


Figure 2.15 Hardware multicast operations.

the multidestination message, the multicast path information are encoded and appended to the message header. The multicast encoding schemes for multicomputer systems were proposed in [94].

The hardware-based multicast schemes can be classified as *path-based* and *tree-based* schemes. In the path-based schemes, the primary problem for multicasting is finding the shortest path that covers all multicast destinations [37]. After the multicast path is selected, the intermediate destinations perform absorb-and-forward operations along the path. The path-based approach is more suitable for direct networks since all switch routers connects to at least one processing node. The path-based multicasting algorithms [34, 41, 95, 96, 97] that have been proposed for the direct networks are focused on the design of deadlock-free multicast algorithms. The detail of this deadlock configuration will be discussed later in Chapter 4. The path-based approach is not convenient for indirect networks because some intermediate switches are not connected to any processing node. Furthermore, it has been shown in [98] that the path-based multicasting in MINs could cause deadlocks.

In tree-based schemes, the multicast problem is the finding of Steiner tree with a minimal total length to cover all multicast destinations [37]. The tree operations introduce additional network resources dependencies which could result in deadlock situation. It is very difficult to avoid the deadlock if the global information is not available. Hence, in wormhole-switched direct networks, the tree-based multicasting is usually undesirable. In [99], a tree-based multicast algorithm for meshes was proposed which supports only small size messages and the

deadlock problem is solved using buffers. MINs inherit the tree topology which can be effectively exploited to support tree-based multicast communication. Several tree-based multicasting algorithms have been proposed in th literature [35, 36, 98, 100, 101]. These algorithms use different deadlock avoidance techniques. In chapter 5, issues in multicasting in MINs will be discussed in detail.

3 UNICAST COMMUNICATION IN MESH NETWORKS

In multicomputers, tasks are executed by a set of intercommunicating nodes or processors. This communication is usually carried out by means of passing messages from one node to another over the interconnection network. Since direct networks can utilize the locality of the message references more efficiently, most of the existing systems use direct networks such as k-ary n-cube or n-dimensional meshes. Examples of such systems include DASH [3], ALEWIFE [12], J-Machine [13], Intel Paragon [16], Cray T3D/T3E [17, 5], etc. In [48], Dally has shown that lower dimensional networks offer lower latency and higher throughput than higher dimensional networks.

Message passing in multicomputer systems is implemented based on certain routing algorithm that determines the path a message follows to reach its destination. Guaranteed delivery of the message requires that the algorithm be deadlock and livelock free. If the path between every pair of source and destination is fixed, the algorithm is called a *deterministic algorithm*. For better system performance, it is however preferable that the algorithm adapt itself to the network faults and traffic congestion and allow alternate paths. This leads to the development of adaptive algorithms.

The adaptive algorithms presented in [60, 61, 62, 82] try to achieve more adaptivity by allowing more number of alternate paths for message routing. In order to achieve high adaptivity, these algorithms often favor some messages or some paths over the others, which in turn, cause an uneven traffic distribution in the network. As a result, a part of the network is heavily loaded whereas other regions may be sparsely utilized. This uneven network utilization often results in an early saturation of the network and limits the system performance. The system performance, thus depends not only on adaptivity of the algorithm, but also on how evenly the network traffic is distributed. In fact, our simulation results clearly indicate that the traffic distribution created by the algorithm has a significant impact on the system performance. In [102], Boppana and Chalasani have also shown that more adaptivity does not necessarily mean better performance. Thus, the main motivating factor behind our work is to develop a new routing algorithm that is not only more adaptive but also creates a balanced traffic distribution.

In this chapter, we propose a fully adaptive minimal routing scheme for two-dimensional (2D) meshes. The algorithm uses only two virtual channels [54] per physical channel creating two virtual networks. Messages are routed positive-first in one virtual network and negative-first in the other. Because of the way the algorithm uses two distinct virtual networks, we call it *Positive-First-Negative-First* (PFNF) algorithm. The proposed routing scheme is described in two phases. First, we introduce a new concept called the *region of adaptivity*, the region where messages can be routed using all the available paths. Using this concept, we show how various algorithms cause an uneven traffic distribution in the network and their effect on the system performance. Second, we present the details of the PFNF routing algorithm. The results indicate that the PFNF routing algorithm outperforms the previously proposed adaptive routing algorithms in terms of the network latency and throughput. Using the concept of region of adaptivity and the simulation results we also show that the PFNF algorithm creates a balanced traffic load in the network. 3P [60], mesh_route [61], and opt-y [82] algorithms are considered for performance comparison because of their high adaptivity using the same amount of hardware resources.

The rest of the chapter is organized as follows. Section 3.1 presents the required terminologies used in this chapters. The previously proposed routing algorithms in meshes are summarized in Section 3.2. Section 3.3 discusses the motivation behind our works. The PFNF routing algorithm is described in Section 3.4. Simulation results are presented in Section 3.5.

3.1 Terminologies

In this section, we define the terminologies associated with the adaptive routing scheme. Some of these definitions are reiterated from previous works [50, 62, 72] for the sake of completeness.

Definition 3.1: A *physical interconnection network*, PN, is a strongly connected graph, $PN(PV, PC)$, where PV represents the set of processing nodes and PC represents the set of physical channels connecting the nodes.

Definition 3.2: A *virtual interconnection network*, VN, is a strongly connected graph, $VN(PV, VC)$, where PV represents the set of processing nodes and VC represents the set of virtual channels, that are mapped to the set of physical channels PC.

Definition 3.3: A *2-dimensional mesh* is defined as an interconnection structure that has $K_0 \times K_1$ nodes with K_i as number of nodes in the ith dimension. Each node in the mesh is identified by an 2-coordinate vector (x_0, x_1). Two nodes, (x_0, x_1) and (y_0, y_1), are connected if and only if either ($x_0 = y_0 \pm 1$, and $x_1 = y_1$) or ($x_0 = y_0$, and $x_1 = y_1 \pm 1$) conditions are

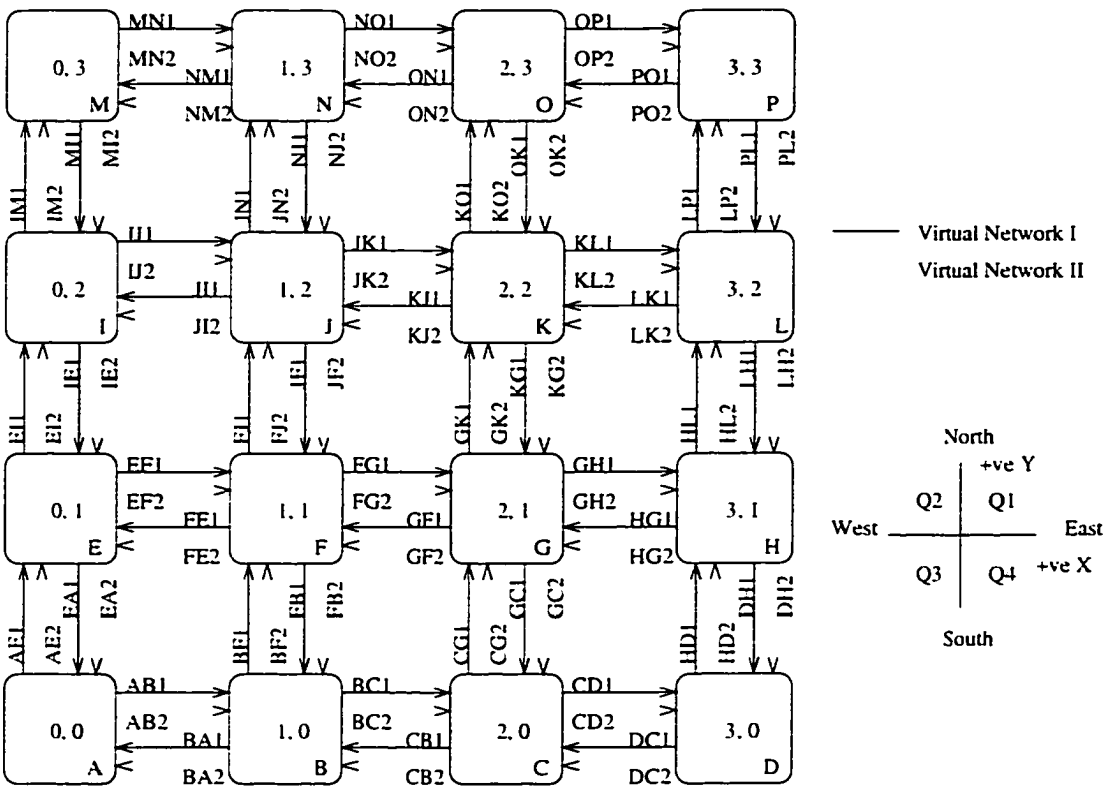


Figure 3.1 A 4 × 4 two-dimensional mesh.

satisfied.

Figure 3.1 shows a 4 × 4 2D mesh with two virtual channels per physical channel. The two virtual networks are shown as $VN1$ and $VN2$. Each node in the mesh is identified by an 2-coordinate vector (x_0, x_1). The figure also shows the directional notations, the quadrants, and labels used in this chapter.

Definition 3.4: A *routing function* $R : N \times N \rightarrow \rho(C)$, where $\rho(C)$ is the power set of VC, supplies a set of alternative output channels to send a message from current the node x to the destination node d. $R(x, d) = (c_1, c_2, \ldots, c_p)$.

Definition 3.5: A routing function for a given interconnection network is connected iff, for any pair of nodes x, $y \in N$, it is possible to establish a path $P(x, y) \subset \rho(C)$ between them using channels supplied by R.

Definition 3.6: A *routing subfunction* R_1 for a given routing function R is a routing function that supplies a subset of channels supplied by R. Thus, R_1 restricts the routing options supplied by R. The set of all the channels supplied by R_1 is $C_1 = \cup_{\forall x,y \in N} R_1(x, y)$.

Definition 3.7: Given an interconnection network I, a routing function R, a routing subfunction R_1, and a pair of channels c_i and c_j supplied by R_1 for some destinations, there is a *direct*

dependency from c_i to c_j iff c_j can be used immediately after c_i by messages destined for some node x.

Definition 3.8: Given an interconnection network I, a routing function R, a routing subfunction R_1, and a pair of channels c_i and c_j supplied by R_1 for some destinations, there is an *indirect dependency* from c_i to c_j iff it is possible to establish a path from s_i to d_j for messages destined for some node x. The source and the destination nodes of channel c_i are denoted as s_i and d_i, respectively. c_i and c_j are the first and last channels in that path and the only ones supplied by R_1. Thus, c_j can be used after c_i by some messages. As c_i and c_j are not adjacent, some other channels not supplied by R_1 are used between them.

Definition 3.9: Given an interconnection network I, a routing function R, a routing subfunction R_1, and a pair of channels c_i, and c_j, there is a *direct cross dependency* from c_i to c_j iff c_j can be used immediately after c_i by messages destined for some node y, c_j is supplied by R_1 and c_i cannot be supplied by R_1 for that destination. However, c_i can be supplied by R_1 for some other destination(s).

Definition 3.10: Given an interconnection network I, a routing function R, a routing subfunction R_1, and a pair of channels c_i, and c_j, there is an *indirect cross dependency* from c_i to c_j iff it is possible to establish a path from s_i to d_j for messages destined for some node y. c_i and c_j are the first and last channels in that path. c_j is the only channels supplied by R_1. c_i can not be supplied by R_1 for that destination. However, c_i is supplied by R_1 for some other destination(s). Thus, c_j can be used after c_i by some other messages. As c_i and c_j are not adjacent, some other channels not supplied by R_1 are used between them.

Definition 3.11: A *channel dependency graph* D for a given interconnection network I and routing function R, is a directed graph, $D = G(C, E)$. The vertices of D are the channels of I. The arcs of D are the pairs of channels (c_i, c_j) such that there is a direct dependency from c_i to c_j.

Definition 3.12: An *extended channel dependency graph* D_E for a given interconnection network I, and routing subfunction R_1 of routing function R, is a directed graph. $D_E = G(C_1, E_E)$. The vertices of D_E are the channels supplied by the routing subfunction R_1 for some destinations. The arcs of D_E are the pairs of channels (c_i, c_j) such that there exists either a direct, indirect, direct cross or indirect cross dependency from c_i to c_j.

Definition 3.13: *Region of Adaptivity* is the area in which a message can route fully adaptively using all the available virtual channels. Quantitatively, the region of adaptivity of a source node is defined as a region constituting a set of contiguous nodes through which a message sent by the source can be routed fully adaptively using the underlying routing algorithm (i.e., using all of the physical paths or all of the virtual paths, if virtual channels are used). The

region of adaptivity of a routing algorithm is the region of adaptivity of a source node at the center of the network.

Consider a 2D mesh network as shown in Figure 3.2(a). With node (3, 1) as the source node, under the East-First algorithm [77], all the messages directed towards any of the nodes in the shaded region can be routed fully adaptively, while messages to any node outside this region would be routed deterministically. We call the shaded region as the region of adaptivity of the node (3, 1) under the East-First algorithm. Figure 3.2(b) represents the region of adaptivity for the node (1, 2). As a whole, the region of adaptivity of the East-First algorithm can be represented as the shaded region shown in Figure 3.2(c). If virtual channels are used, the region of adaptivity can be obtained by considering adaptive regions of all the virtual networks.

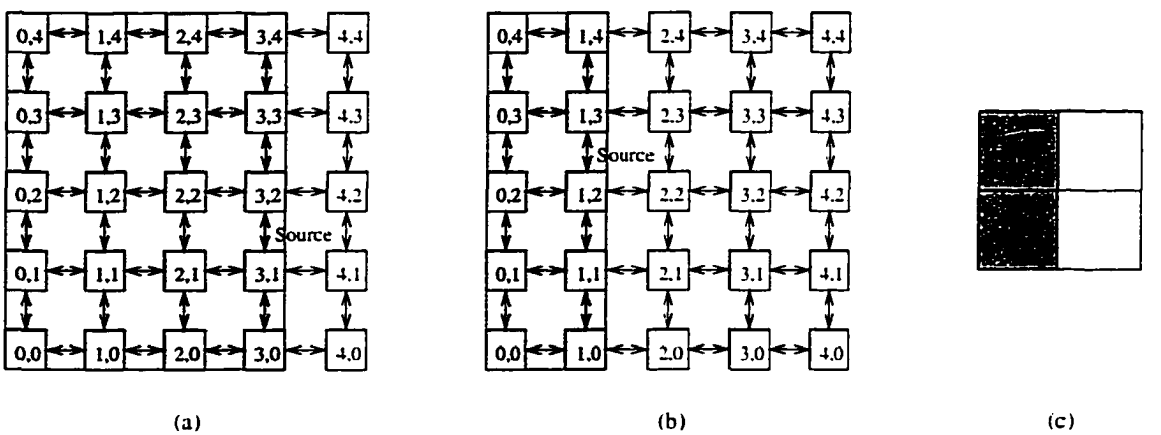


Figure 3.2 Region of adaptivity of (a) node (3, 1) (b) node (1, 2) (c) the East-First algorithm.

The concept of region of adaptivity helps us understand the behavior of the algorithms, particularly on how evenly the algorithm distributes the network load. Since the East-First algorithm has one-half of the mesh as the fully adaptive region, all the messages have more number of alternate paths to route in this region than in the other half of the mesh. Since this region is not symmetric in the mesh, assuming uniform traffic generation at all nodes, it causes more traffic congestion in one part of the network and hence lead to early saturation. Saturation in only one part of the network saturates the rest of the network and hence degrades the system performance. Previous results have indeed shown that the symmetric Negative-First algorithm performs better than the partially East-First algorithm under uniform traffic distribution [77]. Thus, the region of adaptivity closely relates to the performance of the algorithm.

3.2 Unicast Routing Algorithms in Meshes

The path from the source to the destination is determined by routing information from the source address and the destination address. For the same pair of source and destination, all packets will follow the same path in deterministic routing. The shortest path can be easily found in the mesh network. The next hop is decided to the adjacent node that make a packet closer to the destination in one dimension.

In dimension ordered routing [50], the routing algorithm selects the shortest path that traverse network dimensions in sequence. Term *e-cube routing* is used for dimension order routing in hypercube networks. A packet traverse channels in the lowest dimension with non-zero displacement until that dimension have displacement of zero. Figure 3.3 shows the deterministic x-y dimension-order routing in the 16 × 16 two-dimensional mesh network. A packet first will routes in x-dimension. After a packet finishes the x-dimension, y-dimension is traversed. The dimension-order routing is straightforward and simple to implement. However, in asymmetric workload, some channels in network may be overloaded because only one fixed path is allowed.

Figure 3.4 (a) shows channel labels of the four nodes mesh network. The dimension ordered routing is deadlock-free because the cyclic waiting for channel resources does not exist. The direct dependencies exist for channel (1,2), (3,4), (5.6), and (7.8). These dependencies do not form any cycle in the channel dependency graph, as presented in Figure 3.4 (b).

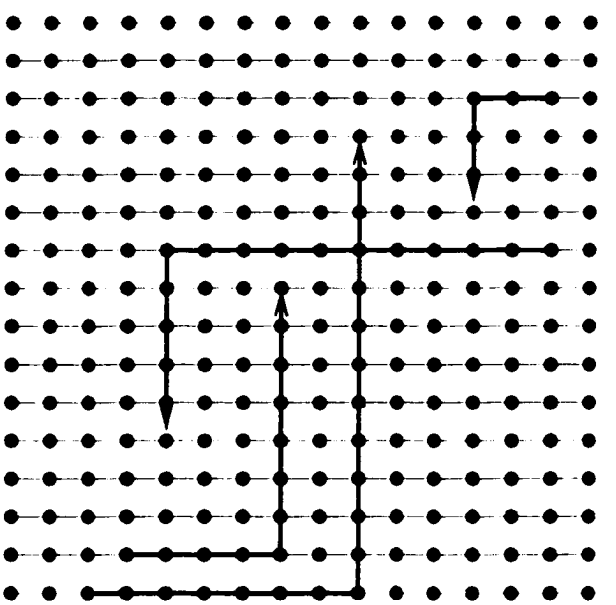


Figure 3.3 Dimension-order routing.

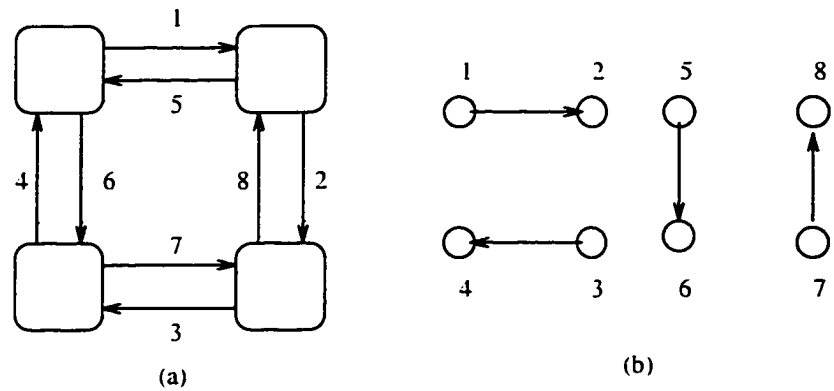


Figure 3.4 Channel dependency graph for dimension-order routing.

Turn Models for adaptive routing [76, 77, 78] have been proposed to represent the abstract turns in two-dimensional mesh networks. The turn models can be used to design turn prohibitions. The more number of turns supplied by the routing algorithm increase the degree of freedom in routing. To design the deadlock free routing algorithms, some turns are prohibited to brake the cycle in channel dependency graph. Based on the prohibited turns, the routing algorithm is defined. Figure 3.5 (a) shows all possible turns in the 2-dimensional mesh networks. In dimension ordered routing, half of all possible turns are prohibited, as depicted in Figure 3.5 (b). Only the turns from the x-dimension to the y-dimension are allowed. Notice that only two turn prohibitions is adequate to brake cycles represented in Figure 3.5 (a).

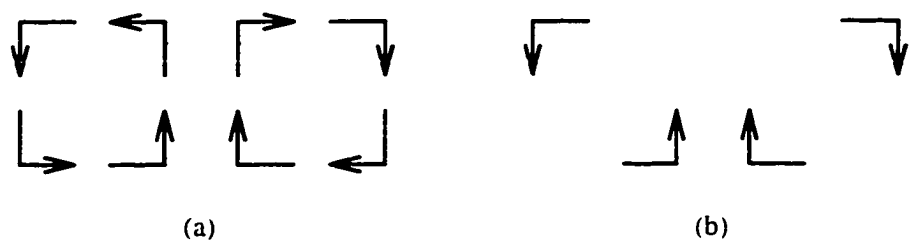


Figure 3.5 The turn model. (a) all possible turn. (b) X-Y dimension-order routing.

To design the efficient adaptive routing algorithms, the number of turn prohibitions should be minimized. The cycles in the channel dependency graph are broken by eliminating one turn in the clockwise cycle and one turn in counter clockwise cycle. Figure 3.6 represents the sixteen possible combination of the turn models in two-dimensional meshes [103]. Four out of these models cannot be used because the routing algorithms is not connected and deadlock situation is possible.

CCW / CW				
	Possitive First	West Last	Not Connected	North First
	East First	Pos Y Neg X First	North Last	Not Connected
	Not Connected	South First	Negative First	East Last
	South Last	Not Connected	West First	Pos X NegY First

Figure 3.6 The turn models for 2-dimensional mesh network.

The routing algorithms can be directly defined using turn models. The turns prohibitions limit the number of paths to forward a packet in some directions. The positive y direction and the positive x direction is defined as the *north* and the *east* direction, respectively. In *East First* routing algorithm, the turns from positive y and negative y to positive x are prohibited. If the destination node is located at positive x direction from the source node, x-y dimension-order routing is used, otherwise fully adaptive. Another example of adaptive routing algorithm is the *Positive-First* routing algorithm. A packet can be routed using any shortest path to the $(+x, +y)$ and $(-x, -y)$ directions. The $y - x$ dimension-order routing and $x - y$ dimension-order routing are used to forwarded a packet to the $(-x, +y)$ and $(+x, -y)$ respectively. In Positive First routing algorithm, the positive displacement must be finished before any other directions. Using the turn model concept, a family of partially adaptive routing algorithms in the mesh network were presented by Glass and Ni [77].

The virtual channel concept is commonly used to improve the degree of adaptivity and the network performance. By the additional flit buffers associated with each physical channel, the number of virtual paths in the network are increased. Glass and Ni have proposed the adaptive routing algorithms for 2-dimensional meshes that require two virtual channels only in y-dimension [104]. Figure 3.7 shows a switch router with two channels in y-dimension. The additional channels can be implemented as the physical channels or the virtual channels. The

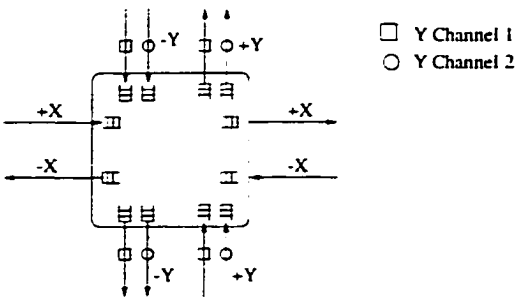


Figure 3.7 A router with double-y channels.

turn prohibitions for fully adaptive *double-y* routing algorithm are shown in Figure 3.8 (a). By removing some routing restrictions while preserving deadlock freedom, maximal adaptive double-y algorithm was proposed by the same researchers. The turns allowed by the mad-y are shown Figure 3.8 (b). The mad-y scheme provides better performance than the double-y scheme with the same cost. An optimal fully adaptive routing algorithm *opt-y* [82] has been proposed by Schwiebert and Jayasimha. The cycles are allowed in the channel dependency graph with escape paths. Dash lines, in Figure 3.8 (c), represent turn prohibitions in opt-y algorithm and the dotted lines denotes the conditionally restrictions. No turn is allowed if the turn is prohibited. Only a message that has more than one dimension to traverse is not allowed to turn when the conditionally restriction is imposed. The opt-y scheme is optimal with the number of virtual channel in the router.

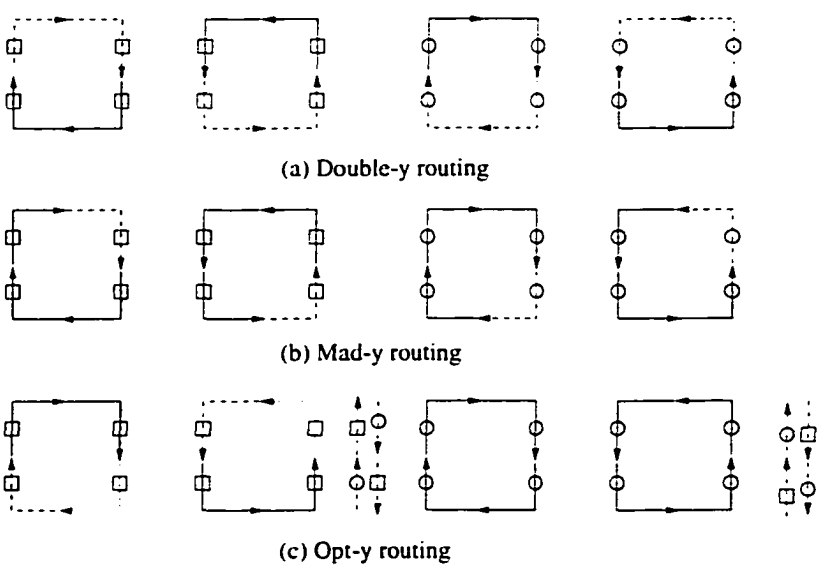


Figure 3.8 Adaptive routing algorithms using double-y channels (a) double-y routing (b) mad-y routing (c) opt-y routing.

The adaptive routing algorithms with virtual channels require only the additional flit buffers to enhance the algorithm adaptivity. Figure 3.1 shows the 2-dimensional mesh network, implemented with two virtual channels per physical channel. Two unidirectional channels are used as connections between two adjacent nodes. Most of the previously proposed adaptive routing algorithms consider an interconnection network with virtual channels as layer of virtual networks. A packet is allowed to route freely in one virtual network, called fully adaptive layer [1]. If the network resource contentions occur, a packet has to wait for the other layer and is routed with some turn restrictions to prevent deadlock.

Dally [59] proposed the dimension reversal routing. A packet is allocated to virtual channels using a count number of dimension reversals (DR). All packets are started with a DR of zero. Each time that a packet is changed to a lower dimension, the DR of a packet is incremented. Two allocation algorithms, static and dynamic, were proposed. The static algorithm separates the virtual channels into classes numbered zero to r where r is the maximum number of dimension reversal permitted. Packets with a $DR < r$ are allowed to route freely in only a virtual channel of class DR. If a packet has a $DR = r$, it must dimension-order be routed in the virtual channel of class r. The dynamic algorithm allows packets to route in any direction with no limit on the number of dimension reversal. The virtual channels are divided into two classes, adaptive and deterministic. Packets are first routed on the adaptive channel. A packet with DR cannot wait for a channel labeled with $q \leq DR$. If all channels with equal or lower DR are occupied, a packet must change to deterministic channel and is not allowed to use the adaptive channel again.

The 3P routing algorithm was introduced by Su and Shin [60]. A similar approach was also independently proposed by Duato for hypercube networks [62]. The 3P routing algorithm is the fully adaptive minimal routing. This algorithm requires only one extra virtual channel for the mesh network. The network is logically divided into two virtual networks, namely, the fully adaptive layer and the deterministic layer. No routing restriction is applied in the fully adaptive layer and therefore cycles are exist in the channel dependency graph. The dimension-order routing algorithm is implemented in the deterministic layer. A packet will first be routed in any direction that it can make progress towards the destination in the fully adaptive layer. If all selected channels in the fully adaptive layer is not available, a packet have to wait for the deterministic channel. The 3P routing is guaranteed to be deadlock-free by providing escape paths to a destination in the deterministic layer.

Mesh_route is an efficient fully adaptive wormhole routing for n-dimensional mesh proposed by Boura and Das [61]. Similar to 3P routing, the virtual channels are classified as the waiting

[1] The fully adaptive layer is called non-waiting layer in some literatures.

and the non-waiting channel. The routing algorithm consists of two phases. A packet is first routed along any dimension using a free non-waiting channel that can forward a packet closer to its destination. Unless non waiting channel is available, a packet must be routed on the lowest positive dimension. The major distinction between 3P routing and the mesh route is that if all elements in the routing tag is less than zero, the mesh route allows a packet to route freely to the negative direction using the waiting channels. Accordingly, the waiting channels are more utilized in the mesh_route algorithm.

3.3 Motivation

The motivating factor for development of our algorithm is driven from the observation that most of the fully adaptive algorithms presented in the literature either have more routing restrictions or their adaptivity is improved at the expense of uneven traffic distribution in the network. We illustrate this point by comparing two recently proposed algorithms - 3P [60] and mesh_route [61]. Both 3P and mesh_route algorithms divide the virtual channels into two separate sets - waiting channels and non-waiting channels. Using 3P algorithm, a message can travel using any of the non-waiting virtual channels. If it gets blocked, it travels through the waiting channels using dimension order routing [50]. Similarly, in mesh_route algorithm, a message can travel using any available non-waiting channel. If the non-waiting channels at a node are not available, then the messages are restricted to the dimension order routing in waiting channels, except those who have to go negative in both x and y directions. These messages can use all the waiting channels without any restriction.

The average buffer utilization of 3P and mesh_route algorithms for uniform traffic are shown in Figures 3.9 (a) and (b), respectively. While 3P algorithm distributes the traffic very evenly, the traffic is concentrated in one quadrant of the mesh for the mesh_route algorithm. This uneven load distribution becomes a bottleneck as the high traffic areas saturate early and in turn saturate the whole network. The 3P algorithm has more routing restrictions due to the dimension order routing in the waiting channels. On the other hand, mesh_route has relatively less routing restrictions and it favors messages going in the negative directions and hence creates an uneven traffic distribution in the network.

The recently proposed opt-y algorithm is proven to be optimal in terms of adaptivity and the number of required virtual channels [82]. But the traffic distribution created by the algorithm is not symmetric when the messages are uniformly generated, as shown in Figure 3.9 (c). Thus the goal of our work was to design an algorithm that is more adaptive as well as produces a balanced and symmetric network traffic load and thereby improves the system

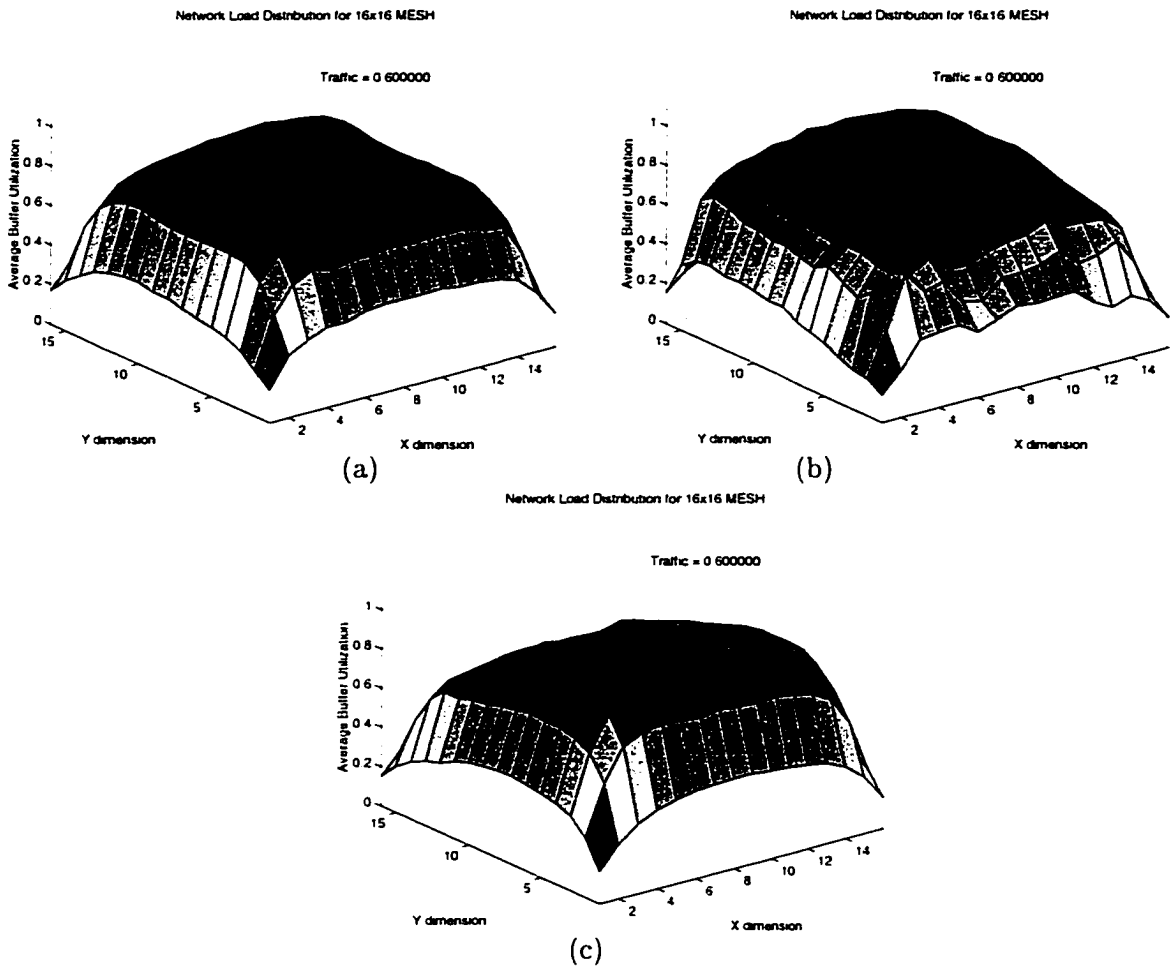


Figure 3.9 Traffic distribution (a) 3P algorithm (b) mesh_route algorithm (c) opt-y algorithm.

performance.

3.4 PFNF Routing Algorithm

The basic concept behind our algorithm is as follows. The physical interconnection network PN is logically divided into two virtual networks, $VN1$ and $VN2$, such that two virtual channels associated with the same physical channel are in different virtual networks. A different routing algorithm is used in each of the two virtual networks, $VN1$ and $VN2$. At each step, a set of virtual channels are chosen from the two virtual networks depending upon the routing tag and the routing function for that particular virtual network. The selection function then selects the channel through which the message is routed.

Routing algorithm(message_header)
/* Let the current node be (x_0, x_1) and destination be (d_0, d_1). */

1. Routing_tag(message_header). /* form the routing tag. */
2. If all elements in routing_tag = 0, store the current message and return.
3. VC = Routing_function(routing_tag). /* determine the set of virtual channels, VC, for the next step routing. */
4. vc = Selection_function(VC). /* select an appropriate vc from VC. */
5. If $vc \neq \emptyset$, forward the message along virtual channel, vc.

Figure 3.10 PFNF routing algorithm.

The PFNF algorithm for 2D mesh can be described as follows. A virtual channel directed from node (x_0, x_1) to (d_0, d_1) is denoted by $vc_{VN}((x_0, x_1), (d_0, d_1))$, where VN is the virtual interconnection network to which the virtual channel vc belongs. The routing algorithm is defined as shown in Figure 3.10. The Routing_tag() and Selection_function() are described in Figure 3.11.

When the message header arrives at an intermediate node, the routing_tag is calculated. This routing_tag is used to determine the routing dimension to be completed. The incoming message is consumed if all elements in the routing_tag is equal to zero. The function, Routing_function (described later), returns all virtual channels that are allowed by the routing algorithm. The specific virtual channel taken by a message is chosen by Selection_function. Selection_function will first check the availability of all virtual channels in the physical channels. If the number of available virtual channels is more than one, a selection policy is applied. The selection policy can be random, turn biased, or multiplex-turn biased. In random selection policy, the virtual channel is chosen randomly from the free set. The network contention is reduced if messages avoid making turns. The turn bias policy selects the virtual channel in the same direction if possible. The performance degradation due to the virtual channel is attributed to the delay in multiplexing the physical channel. The multiplex-turn bias first avoids sharing the physical channel with other message if possible, and then avoids making a turn as a second priority. The effect of these selection policies have been studied in [59, 76]. The simulation results show that the turn bias and multiplex-turn bias perform better than the random selection policy [59, 76, 103].

Figure 3.12 shows the PFNF routing function. In the procedure of routing function, PF routing algorithm is implemented in $VN1$ and NF routing algorithm is implemented in $VN2$. In both PF and NF routing algorithms, a message can route without restriction to destinations

Procedure Routing_tag(message_header)

1. For $i = 0$ to 1 do
   If $d_i - x_i > 0$, routing_tag[i] = 1
   If $d_i - x_i < 0$, routing_tag[i] = -1
   If $d_i - x_i = 0$, routing_tag[i] = 0 /* Finish routing in dimension i */
   end.
2. return routing_tag.

Procedure Selection_function(VC)

1. If the number of members in $VC = 1$, then select it.
2. Otherwise, use multiplex-turn bias (explanation follows) to select a vc.

Figure 3.11 Routing tag function and selection function.

in $< +x, +y >$ and $< -x, -y >$ directions from the source. When the destination is located in the directions $< -x, +y >$ or $< +x, -y >$ of the source node, the routing restrictions of PF and NF algorithms are applied. To implement the PF algorithm in one virtual network and NF routing algorithm in another, the routing function is divided into three cases. The first two cases return virtual channels in both $VN1$ and $VN2$ for the message destined to $< +x, +y >$ and $< -x, -y >$ directions. The third case returns virtual channels in the $VN1$ for the positive direction and the $VN2$ for the negative direction. The routing restrictions for PF and NF are applied in this case.

The deadlock freedom of the algorithm can be proved by using Duato's theorem stated as follows [62, 72]. The proof of the theorem uses several terminologies associated with the channel dependency graph. These terminologies are defined earlier in [50, 62, 72].

Theorem 3.1: For a given interconnection network IN, a routing function R is deadlock-free iff there exists a routing subfunction R_1 which is connected and has no cycles in its extended channel dependency graph.

To prove that the PFNF algorithm is deadlock free, we first analyze the routing restrictions. The turn models for the PF and NF routing algorithms [77] are depicted in Figure 3.13 (a) and (b), respectively. The two sets of channels belonging to the virtual networks are distinguished by | and ||, respectively. Dotted lines represent restricted turns. Glass and Ni [76] showed that without any extra virtual channel, the routing algorithm is deadlock free if there is no cycle formed in turn model. When we consider the virtual networks individually,

Procedure Routing_function(routing_tag)

$VC = \emptyset$
1. If all elements in routing_tag ≤ 0
   If routing_tag[0] < 0
   add $vc_{VN1}((x_0, x_1), (x_0 - 1, x_1))$ and $vc_{VN2}((x_0, x_1), (x_0 - 1, x_1))$ to VC.
   If routing_tag[1] < 0
   add $vc_{VN1}((x_0, x_1), (x_0, x_1 - 1))$ and $vc_{VN2}((x_0, x_1), (x_0, x_1 - 1))$ to VC.
   return virtual channel set VC.
2. If all elements in routing_tag ≥ 0
   If routing_tag[0] > 0
   add $vc_{VN1}((x_0, x_1), (x_0 + 1, x_1))$ and $vc_{VN2}((x_0 + 1, x_1), (x_0, x_1))$ to VC.
   If routing_tag[1] > 0
   add $vc_{VN1}((x_0, x_1), (x_0, x_1 + 1))$ and $vc_{VN2}((x_0, x_1), (x_0, x_1 + 1))$ to VC.
   return VC.
3. If routing_tag[0] > 0
   add $vc_{VN1}((x_0, x_1), (x_0 + 1, x_1))$ to VC.
   If routing_tag[0] < 0
   add $vc_{VN2}((x_0, x_1), (x_0 - 1, x_1))$ to VC.
   If routing_tag[1] > 0
   add $vc_{VN1}((x_0, x_1), (x_0, x_1 + 1))$ to VC.
   If routing_tag[1] < 0
   add $vc_{VN2}((x_0, x_1), (x_0, x_1 - 1))$ to VC.
   return VC.

Figure 3.12 PFNF routing function.

the routing algorithms in both the virtual networks (PF and NF) are deadlock-free. Since the PFNF algorithm allows a message to change from one virtual network to another, the channel dependencies between virtual networks also need to be considered. The turns involved between the two virtual networks are shown in Figures 3.13 (c) and (d). Dashed lines represent the conditionally restricted turns. These turns are restricted if a message has more than one dimension to traverse; otherwise, these conditionally restricted turns are allowed.
The turn restrictions for the PFNF algorithm are summarized as follows:

- Within $VN1$: South-East and West-North turns are restricted. (PF restrictions)
- Within $VN2$: North-West and East-South turns are restricted. (NF restrictions)
- From $VN1$ to $VN2$: South-East and West-North turns are restricted. A message can use $VN1$ in South(West) direction iff it has finished routing in the East(North) direction using the PF restrictions imposed in $VN1$. Hence, the South-East(West-North) turns will not occur.

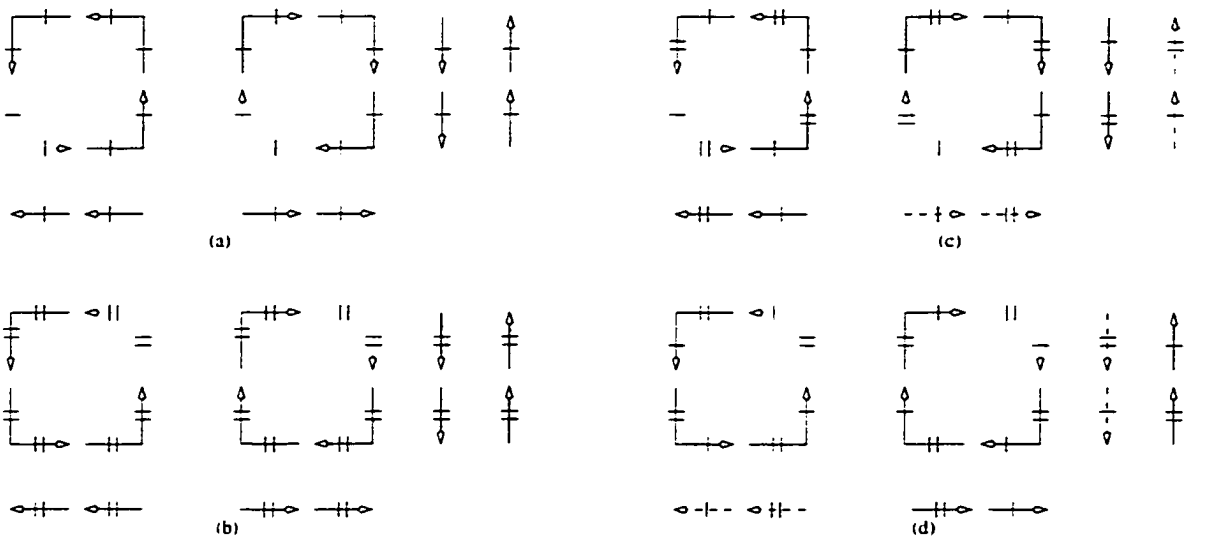


Figure 3.13 Routing restrictions in PF, NF, and PFNF algorithms.

- From $VN2$ to $VN1$: North-West and East-South turns are restricted. A message can use $VN2$ in North(East) direction iff it has finished routing in the West(South) direction using the NF restrictions imposed in $VN2$. Hence, the North-West(East-South) turns will not occur.

- From $VN1$ to $VN2$: While routing in the North(East) direction, messages traveling from $VN1$ to $VN2$ are conditionally restricted. From NF restrictions in $VN2$, a message in $VN1$ can route further in the North(East) direction using $VN2$, only when the routing is not needed in any other negative direction. Hence, changing form $VN1$ to $VN2$ in North(East) direction is prohibited if a message has to route in the West(South) direction at a later time.

- From $VN2$ to $VN1$: While routing in the South(West) direction, messages traveling from $VN2$ to $VN1$ are conditionally restricted. From PF restrictions in $VN1$, a message in $VN1$ can route further in the South(West) direction using $VN2$, only when the routing is not needed in any other positive direction. Hence, changing form $VN2$ to $VN1$ in the South(West) direction is prohibited if a message has to route in the East(North) direction at a later time.

The routing restrictions, described above, may form cycles in the channel dependency graph. However, Duato [62] has shown that a routing algorithm can be deadlock-free in the presence of cycles in the channel dependency graph provided that escape path(s) without the cyclic dependency exists in the extended channel dependency graph.

A routing subfunction R_1 is defined to show that an escape path without cyclic dependency always exists. Using R_1, a message is routed in dimension-order in $VN2$ in the North direction, and in $VN1$ in the South direction from the source node. R_1 is stated as follows: If the destination node (d_0, d_1) is equal to the current node (x_0, x_1), the message is consumed. If $d_1 < x_1$, the message is forwarded using dimension-order routing in $VN1$. If $d_1 > x_1$ then the message is forwarded using dimension-order routing in $VN2$. In other words, R_1 is conditionally assigned to both of the virtual networks. To forward a message in North(South) direction, R_1 defines dimension-order routing using $VN2(VN1)$.

Lemma 3.1. The routing subfunction R_1 is connected.

Proof: It is known that the dimension-order routing is connected. Therefore, the routing algorithm that assigns one virtual network to a message headed toward North direction and another virtual network for a message headed toward South direction is also connected. Q.E.D

The dependency cycles can be classified as intra-dependency cycles and inter-dependency cycles. Virtual channels in the same virtual network are involved in the intra-dependency cycles. The inter-dependency cycles consist of virtual channels from different virtual networks. To prove that there is no cycle in the extended channel dependency graph, we need to show that there are no intra-dependency or inter-dependency cycles in the graph.

Lemma 3.2. Using the routing subfunction R_1, there is no cycle formed due to the channel dependencies in the same virtual network (intra-dependency cycle).

Proof: To form a counter-clockwise cycle within the same virtual network, virtual channels in the directions $(-x, -y, +x, +y)$ in sequence are required. Similarly, virtual channels in the directions $(+x, +y, -x, -y)$ in sequence are required for the formation of a clockwise cycle. The virtual channels directed to North(South) that belong to $VN1(VN2)$ cannot be supplied by R_1 (Since R_1 supplies virtual channels in $VN2$ for the North bound messages, the $+y$ virtual channels in $VN1$ is not supplied by R_1). Thus, the $+y$ virtual channels in $VN1$ are not involved in the extended channel dependency graph and no intra-dependency cycle can be formed in $VN1$. The same scenario is applicable in $VN2$, where $-y$ virtual channels are not supplied by R_1. Q.E.D

Lemma 3.3. Using the routing subfunction R_1, there is no cycle created from the channel dependencies in one virtual network combined with the channel dependencies in the other virtual network (inter-dependency cycle).

Proof: The virtual channels that are involved in the direct and indirect dependencies are supplied by R_1. Since R_1 defines different virtual networks for messages headed in North and South directions, the direct and the indirect dependencies are within the same virtual network. However, the dependency between virtual channels of two different networks can be

created from cross dependencies. Using R_1, the virtual channels that are assigned to restricted destinations are only in the x dimension. For example, $-x$ virtual channels in $VN2$ cannot be assigned using R_1 to route a South bound message, but it can be assigned to a North bound message. However, $-x$ virtual channels in $VN2$ can be supplied by routing function R for a South bound message. Because of the PFNF routing restrictions, there is no inter-cross-dependency starting from $-x$ in $VN1$ and $+x$ in $VN2$. The inter-cross dependencies[2] from $VN2$ to $VN1$ are only from $-x$ virtual channels of $VN2$ to $-y$ or $-x$ virtual channels of $VN1$. Similarly, the inter-cross dependencies from $VN1$ to $VN2$ are only from $+x$ virtual channels of $VN1$ to $+y$ or $+x$ virtual channels of $VN2$. These dependencies are shown in Figure 3.14. The only possible way of forming a cycle is the channel dependencies in sequence: ([$-x$ in $VN2$], [$-y$ in $VN1$], [$+x$ in $VN1$], [$+y$ in $VN2$], [$-x$ in $VN2$]). But, there is no dependency from [$-y$ in $VN1$] to [$+x$ in $VN1$] using PF restrictions. Similarly, there is no dependency from [$+y$ in $VN2$] to [$-x$ in $VN2$] using NF restrictions. Hence, there is no inter-dependency cycle in the extended channel dependency graph. Q.E.D.

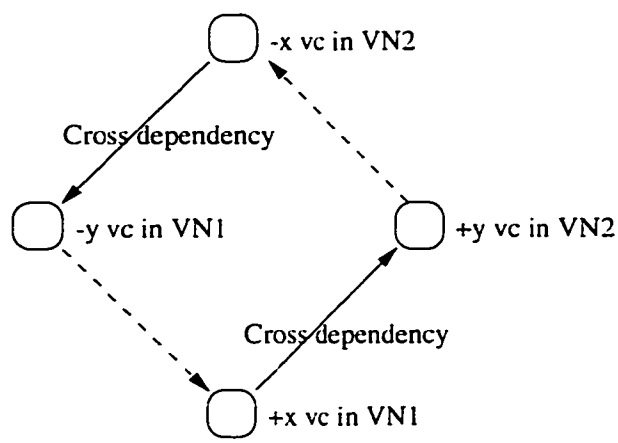


Figure 3.14 Possible cycle that can be formed between virtual networks.

Figure 3.15 shows an example of the extended channel dependency graph for the routing subfunction R_1. The channel labels are defined in Figure 3.1. Vertices, solid lines, dashed lines, and dotted lines represent virtual channels, direct dependency, indirect dependency, and direct cross dependency, respectively. For clarity, indirect dependencies and indirect cross dependencies that are redundant to other direct dependencies and direct cross dependencies are not shown in the figure. A detailed description of the construction of the extended graph is explained in [37, 62, 72].

[2]The inter-cross dependencies include both direct cross dependencies and indirect cross dependencies that connect channels in different virtual networks.

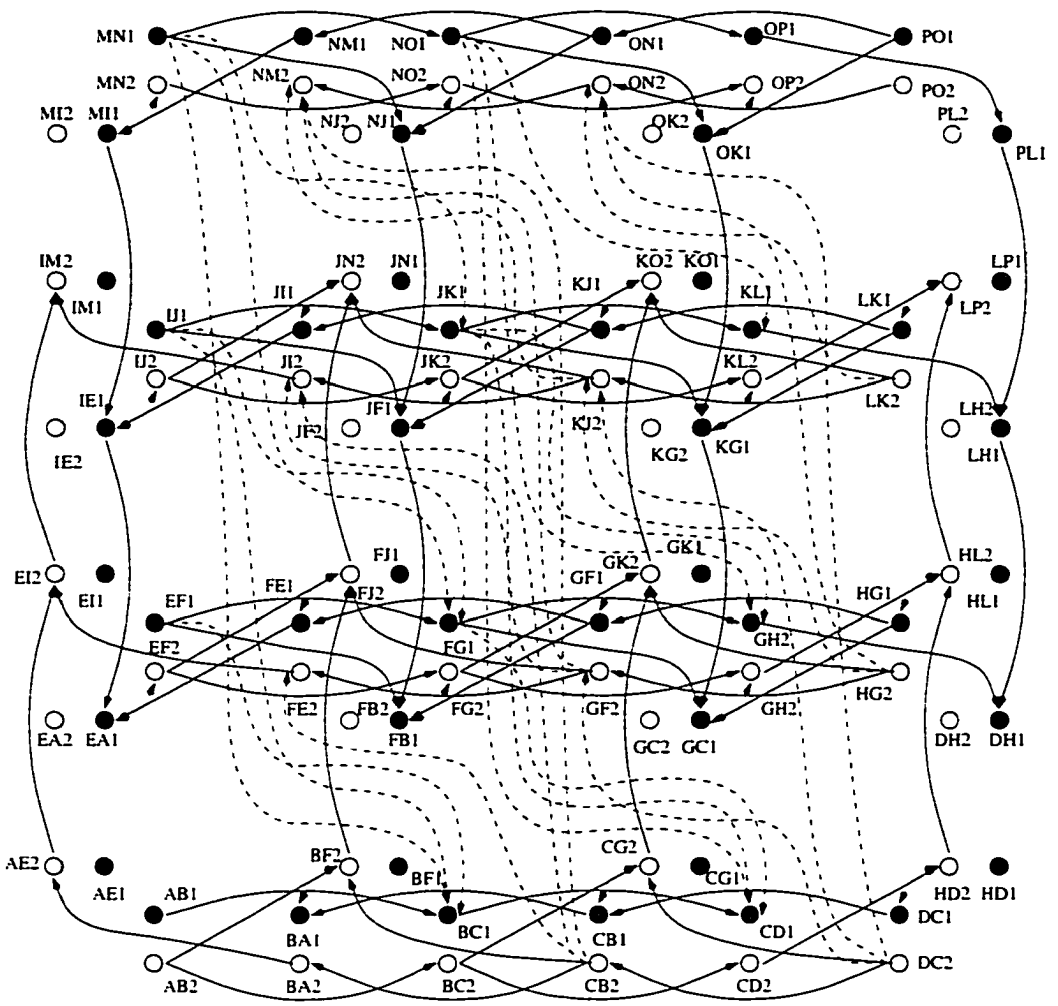


Figure 3.15 Extended channel dependency graph of the mesh in Figure 3.1.

Theorem 3.2. PFNF routing algorithm is deadlock free.

Proof: From Lemma 1 through 3, and using Theorem 1, PFNF routing algorithm is deadlock-free. Q.E.D.

Next, we compare the region of adaptivity of all the four algorithms discussed earlier. As all these algorithms use at most one additional virtual channel per physical channel, we consider the whole network as consisting of two separate virtual networks. Region of adaptivity for each virtual network is(are) the quadrant(s) in which full adaptivity is offered by the algorithm in that particular network. The region of adaptivity of the actual network is then obtained by superimposing the adaptive regions of the two virtual network. The adaptive regions for each virtual network and the total region of adaptivity for the four algorithms are shown in Figure 3.16. The intensity of the shaded areas represent the adaptivity in the regions of the network

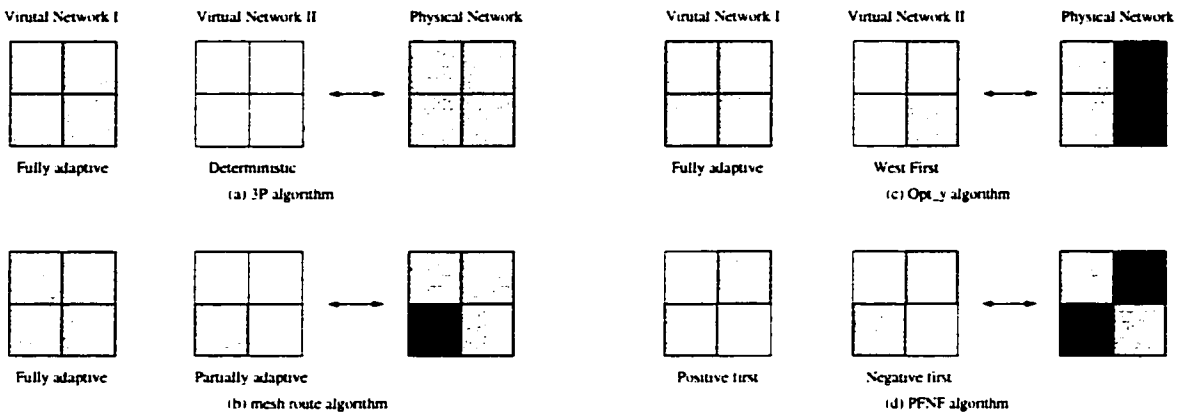


Figure 3.16 Region of adaptivity for (a) 3P (b)mesh_route (c) opt-y (d) PFNF algorithms.

as explained in Section 3.1. Note that, in Figure 3.16(d) the combination of deterministic algorithms in the second and fourth quadrants of the two virtual networks give full adaptivity in the actual physical network. For example, in the second quadrant the PF virtual network allows YX routing and the NF virtual network allows XY routing, thus providing full adaptivity. The figure demonstrate the comparison of the adaptive regions of the algorithms. The symmetry of the adaptive regions are also illustrated. It can be inferred from Figure 3.16 that since the region of adaptivity for the PFNF algorithm is symmetric and more adaptive, it would perform better than the other algorithms. The fault tolerant PFNF algorithm is described in [105]. In [106], the PFNF algorithm is modified for the two-dimensional torus networks.

3.5 Performance Evaluation

In this section, we present results for various traffic patterns and network loads. We have compared the performance of the PFNF algorithm with that of the 3P, mesh_route, and opt-y algorithms. These algorithms are shown in the literature to have better performance or adaptivity than other existing adaptive routing algorithms.

3.5.1 Simulation Environment

The simulations were conducted on a 16 × 16 mesh. We have assumed 20 flits per packet. Each virtual channel is assumed to have only one flit buffer associated with it. Packet generation rate is assumed to have an exponential distribution of inter-arrival time. We have used multiplex-turn biased selection policy for all the algorithms. We have considered a source buffer size of 20 which stores the generated messages if they cannot propagate through the

network. The generation is halted when the source buffer becomes full and is resumed when there is at least one empty space. The simulation was carried out for 150,000 packets. The effect of the first 40,000 to 60,000 delivered packets are not included in the results in order to reduce the transient effects in the simulations. The results were reproduced several times and were observed to be consistent with a maximum deviation of only 1%.

Uniform, hotspot, and transpose traffic patterns were considered in our study. Under the uniform traffic pattern, a node sends messages to every other node with equal probability. Under hotspot traffic, one particular node receives some additional traffic besides its normal traffic. Using the parameters from [102] we have considered only one hotspot node with the hotspot percentage of four, i.e., in a 16×16 mesh a message is directed to the hotspot node with a probability of 0.0438 and to each of the other nodes with a probability of 0.0038. We have chosen node $(5, 5)$ as the hotspot node. Under transpose traffic, a message from node (i, j) is directed to node (j, i) if $i \neq j$. If $i = j$ node (i, i) sends messages to node $(K-i, K-i)$, where K is the network radix.

We have studied the average communication latency, the average throughput of the network and the network load distribution in the network. The communication latency is defined as the average time from the message generation to the time when the tail reaches the destination. The throughput is the average number of messages that finish routing per unit time. The network traffic distribution is measured by finding the average flit buffer utilization at each node. All the above parameters are studied against the network traffic. The network traffic is defined as the ratio of the average traffic generated by a node to the average bandwidth available per node.

3.5.2 Results and Discussions

Figures 3.17(a) and 3.17(b) plot the average latency and average throughput of the network against the network traffic under the uniform traffic pattern. In the low traffic region, all the four algorithms result in almost the same average latency. However, as the traffic is increased, opt-y and mesh_route saturate first and their latencies increase rapidly. The PFNF algorithm performs better than all the schemes. The same trend is also observed in the throughput results. All the algorithms give the same throughput for lower traffic rates (< 0.3), however at higher traffic, the throughputs of the opt-y and mesh_route algorithms drop abruptly. Throughput using 3P and PFNF schemes do not drop abruptly but instead saturate close to their maximum values.

The opt-y and mesh_route algorithms have less routing restrictions than the 3P routing. However, they create uneven traffic distribution in the network. The fact that 3P shows

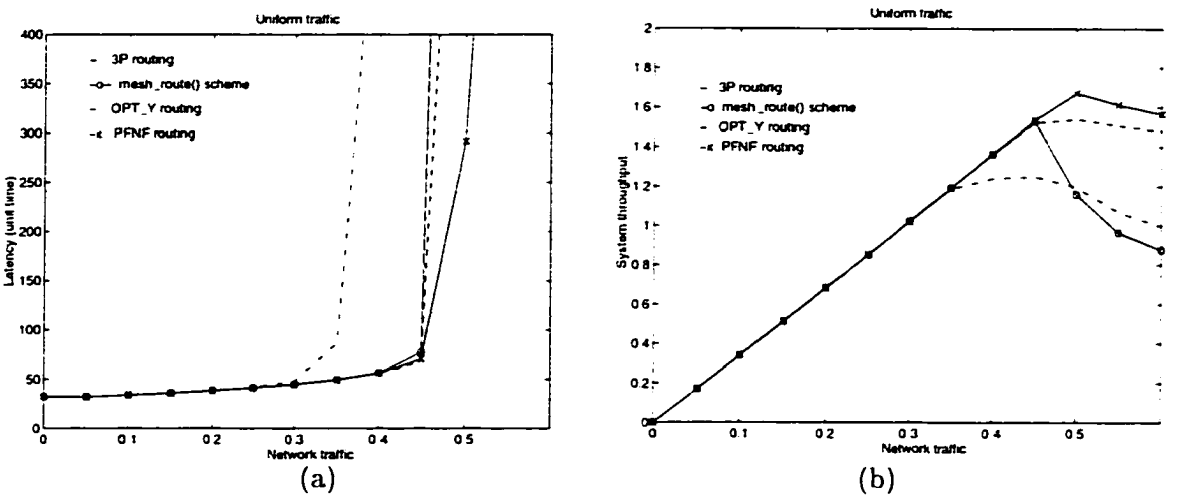


Figure 3.17 Performance results for the uniform traffic pattern.

higher performance compared to opt-y and mesh_route, confirms our claim that the system performance depends significantly upon how evenly the traffic is distributed. The PFNF algorithm is highly adaptive and it also distributes the network load symmetrically, and hence demonstrates better performance.

To demonstrate that the PFNF algorithm indeed distributes the network load symmetrically we have plotted the traffic distribution for the PFNF algorithm in Figure 3.18. The results are in accordance with what was expected from the region of adaptivity analysis and clearly illustrate that the traffic distribution is more balanced using the PFNF algorithm than the mesh_route (refer to Figure 3.9 (b)).

Figure 3.19 shows the latency and throughput results under the hotspot traffic pattern. The opt-y and mesh_route algorithm perform poorly and saturate early. The PFNF algorithm outperforms all the three schemes in both the latency and throughput results. The same trend is observed under the transpose traffic pattern shown in Figure 3.20.

It should be noted that the opt-y algorithm uses a total of six virtual channels per router compared to eight used by the other three algorithms in a 2-dimensional mesh. Providing the same resources to opt-y will enhance the performance and it becomes almost equal to that of the 3P algorithm [103].

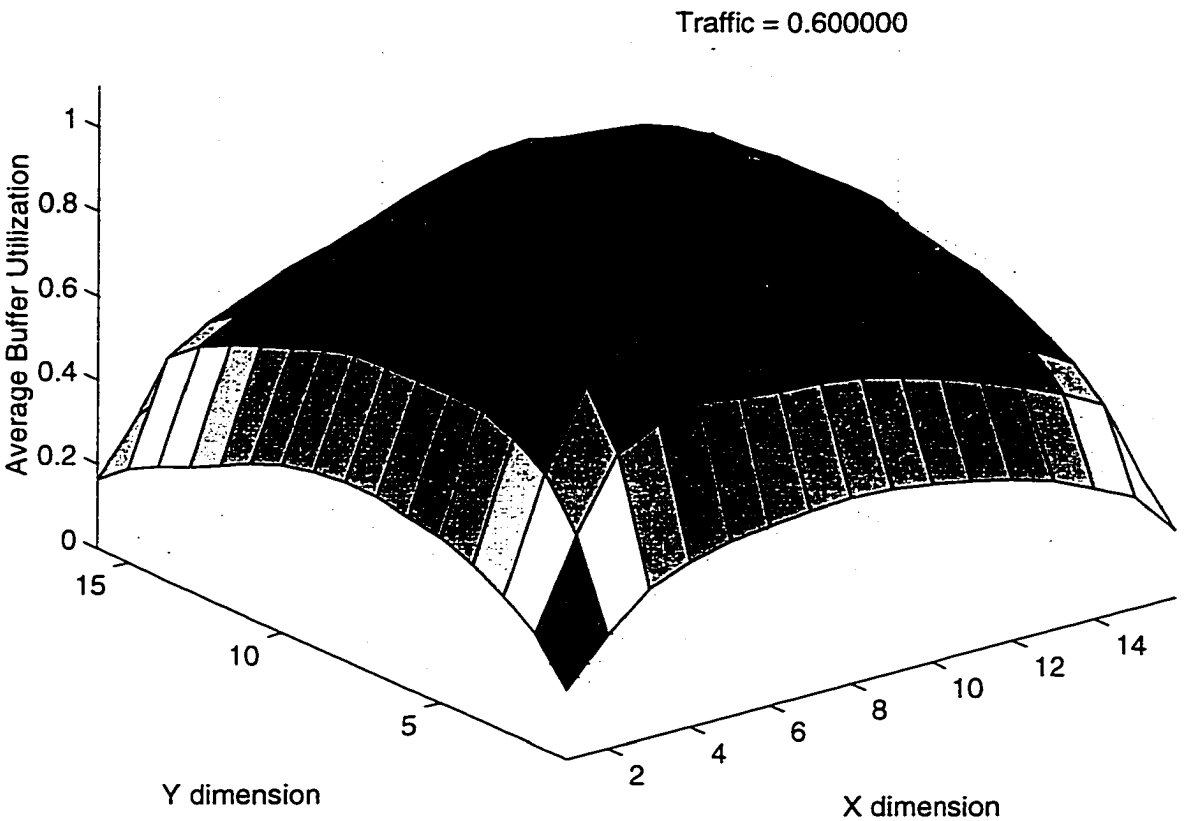


Figure 3.18 Traffic load distribution produced by the PFNF algorithm.

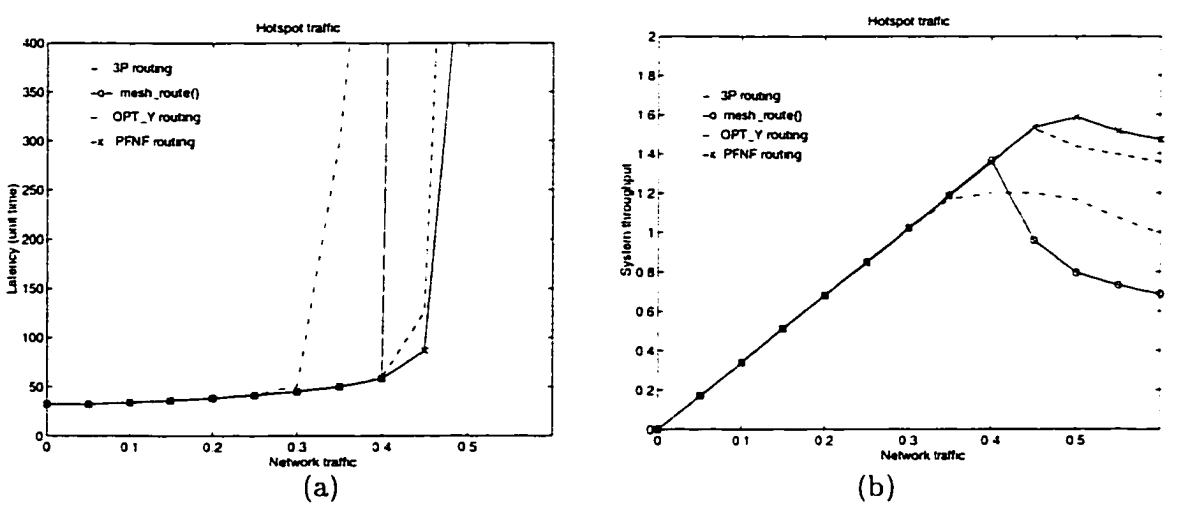


Figure 3.19 Performance results for the hotspot traffic pattern.

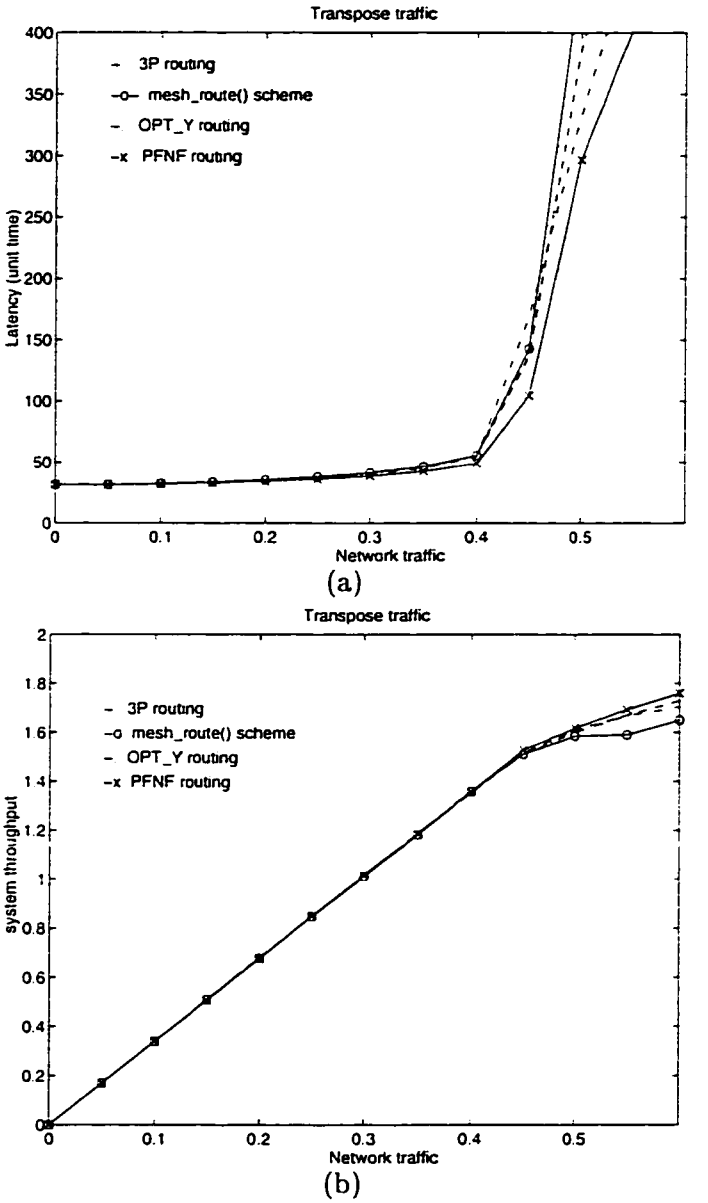


Figure 3.20 Performance results for the transpose traffic pattern.

4 MULTICASTING IN MESH NETWORKS

Multiprocessor systems often require coordination and synchronization between processing elements through interprocessor communication which can be either one to one (unicast communication) or could involve a group of processors (collective communication). Collective communication [26] involves a group of processing nodes that intercommunicate in a specific manner. Examples of collective communication primitives are barrier synchronization, broadcast, gather, scatter, all-gather, all to all, global reduction, and scan. Because of the nature of parallel programming, which requires a group of collaborating processors to complete a single task, efficient support of collective communications is a critical issue in the design of high performance parallel systems [27]. The inclusion of collective communications in the Message Passing Interface (MPI) standard [107] further justifies the research in this area.

An important communication primitive among collective operations is the multicast communication. Multicast communication is a generalization of the broadcast operation and is concerned with sending a message from a source node to a set of destination nodes. The multicast services can be considered as basic services for other collective operations such as broadcast and barrier synchronization. In barrier synchronization [31], a multicast operation is performed to distribute signal to resume the application. It is also used for distributing data to processes. In shared memory systems, multicasting is used for cache invalidations.

Multicast communication can be carried out by sending one message to each of the destinations. This method performs poorly not only because it requires a large amount of network resources but also involves too many communication steps. An efficient software multicasting technique called Umesh is proposed for meshes that do not require any hardware modification [84]. The Umesh scheme reduces the number of communication steps by allowing some destinations to act like source nodes after they receive the message. In [90], an extended dominating node approach was proposed for the broadcast algorithm in meshes. A set of nodes is selected as the dominated nodes. These nodes are used as intermediate sources to cover all destinations. These techniques involve a large number of communication start-up steps and software overheads. To further reduce the communication latency, the multicast operations need to be supported by the hardware. The importance of hardware supported multicasting was reported

by Ni [29]. In [95], the Hamiltonian path based algorithms were proposed to reduce the amount of traffic created from the multicast operations. The router is modified to support an *absorb and forward* operation. This operation allows a message to be forwarded and stored concurrently at an intermediate destination node. The multiple destination addresses are encoded and appended to the message header [94]. Such multicast messages are usually referred to as multidestination messages [41]. The Hamiltonian path guarantees that the message will cover all the destinations. However, in this algorithm, the path length becomes a dominant factor, which may lead to high latency. Furthermore, the Hamiltonian path-based scheme does not conform to any base routing scheme proposed for unicast communications.

Leader-based [41] and column-path [96] multicasting schemes are two similar algorithms developed independently and conform to the base unicast routing algorithm. In [41], it was reported that the number of start-up phases for a single-source multicast for a destination set DS in a $k \times k$ mesh is less than or equal to $\lceil \log_t(2k) \rceil + 2$ if $t > 1$ and $\lceil \log_t(k) \rceil + 2$ if $t \leq 1$, where $kt \leq |DS| \leq k(t+1)$ and t is an integer. We have attempted to further reduce the number of start-up phases and thereby lower the multicast latency. In this chapter, we propose a new hardware multicast routing algorithm that uses at the most two start-up phases to reach any number of destinations in a two-dimensional mesh while conforming to the base unicast routing algorithm. The methodology is called two-phase multicast (TPM) algorithm.

To fully utilize the multicast paths supplied by the base routing algorithm, some intermediate nodes that are not destinations are allowed to perform multicast operations. This feature increases flexibility in distributing messages to the destinations and thereby improves the performance. First, a multicast zone is created that defines the network resources that can be utilized for each multicast operation. The multicast message is sent by the source node to a specific corner node of the multicast zone in the first phase of communication. Next, a set of nodes that received the message during the first phase sends the message to the remaining destinations. The routing paths are selected in such a way that they cover all destinations in at most two phases while conforming to the base routing algorithm. We have presented examples of both deterministic as well as adaptive multicasting schemes. The performance of the multicast algorithms are evaluated through simulations. The simulation results show that the proposed algorithm performs better than the dualpath [95] and column-path [96] algorithms.

This chapter is organized as follows. Section 4.1 presents preliminaries for the multicast routing models. The proposed algorithm along with the hardware support and implementation details are described in Section 4.2. A deterministic TPM algorithm based on the dimension order routing is reported in Section 4.3, followed by the development of the adaptive TPM algorithms in Section 4.4. The simulation results are presented in Section 4.5.

4.1 Preliminaries

We develop multicast routing algorithms for two-dimensional (2-D) mesh networks. As mentioned in Chapter 2, a straightforward way to support multicast communication is to send one message to each destination. Although this method does not require any additional hardware, the multicast operation consists of several communication phases and the number of messages in the network is increased considerably. The number of communication phases can be minimized using the divide and conquer technique purposed in [84]. However, the number of communication phases is of the order of $\lceil \log_2 K \rceil$ to send a multicast message to $(K-1)$ destinations. To further reduce the latency by reducing the number of communication phases and the number of messages, the multicast operations need to be supported by the hardware. Hardware multicasting requires some additional functionality to be implemented in the router along with the capabilities for unicast communication. Some basic communication services required in wormhole routed networks for supporting multicasting in hardware are defined as follows.

- Forward (FWD): When the router accepts a message from a neighboring node and the current node is not one of the destinations, the message is forwarded to the next node toward its destination(s).
- Absorb (ABS): If the current node is the destination, the message is absorbed and passed on to the local processor.
- Retransmit (RTM): For deadlock avoidance, in some cases, the multicast message must be temporarily stored and retransmitted to eliminate the cyclic dependencies. The message is stored in the local memory. Thus the probability of deadlock from the depletion of storage space is negligible. The overhead of retransmission operation is high because it involves both software (operating system) and hardware operations.

Efficient multicast algorithms can be adopted by using a combination of the basic communication services. In the Umesh algorithm [84], the retransmit operations are used in several phases of the multicast operations. The Hamiltonian path-based algorithms [95] are supported by implementing the absorb and forward mechanism. The destination addresses are encoded in the message header. The ordering of the destinations in the header implies the sequence of destinations through which the message traverses. When the message reaches an intermediate destination, the destination addresses in the header are updated by the router. The current destination address is removed from the header and appropriate header processing functions are performed. After sending the message header to the next node, the router keeps absorbing

and forwarding the data flits in a pipelined fashion. The message is eventually consumed at the last destination in its multicast path.

The absorb and forward capability introduces additional resource dependencies that can lead to deadlock situations. Since multiple consumption channels may be occupied by one message along the multicast path, the deadlock configuration can stem from the cyclic waiting for consumption channels. An example of consumption channel deadlock in a linear-array is shown in Figure 4.1. Node 1 generates message A destined to nodes 2 and 3. At the same time, message B is generated by node 4 destined to the same set of destinations. Assuming an one-port model, after two steps, message A occupies consumption channel $c2$ at node 2 and requests for consumption channel $c3$ at node 3. The consumption channel $c3$ is occupied by the message B which also requests for $c2$. This cyclic wait creates a deadlock. The deadlock due to consumption channel contention can involve several nodes. The upper bound of the number of consumption channels required to avoid such deadlocks is equal to nv where n is the network dimension and v is the number of virtual channels per direction [96]. The detailed treatments of the consumption channels deadlock can be found in [70, 108].

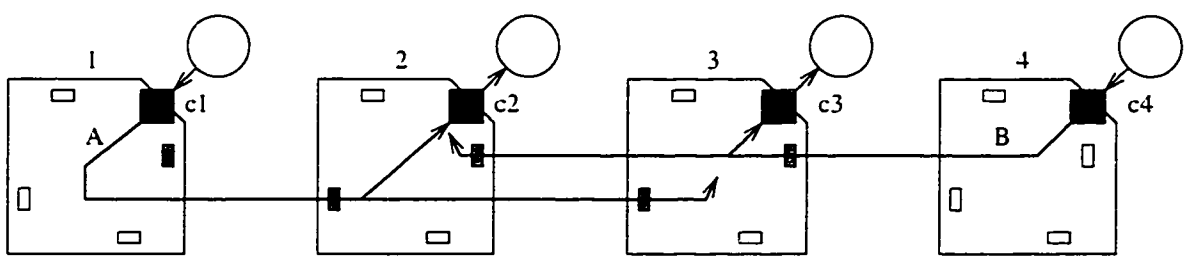


Figure 4.1 Deadlock due to consumption channels.

4.2 A Two-Phase Multicast Algorithm

In this section, we present a new hardware multicast routing algorithm that uses at most two start-up phases to communicate with any number of destinations; hence the name *two phase multicast* (TPM) algorithm. We first outline the basic characteristics of the algorithm followed by the listing of the required hardware supports. The destination preprocessing mechanisms and creation of multicast zone are also discussed. Finally, we provide a detail description of the framework of the TPM algorithm followed by some discussions on the implementation issues.

4.2.1 Basic Characteristics

There are two essential characteristics of the proposed TPM algorithm. First, the algorithm uses the same routing algorithm as that employed for the unicast communication. It conforms to the base routing algorithm and complies to the routing restrictions, and thereby avoids deadlock configurations (assuming that the base routing algorithm is deadlock-free). The deadlock configurations due to the dependencies in the consumption channels can be avoided by using the required number of consumption channels as discussed later in Section 4.3. The second characteristic of the TPM algorithm differentiates it from all the previously proposed algorithms. In the previously proposed multicast algorithms, only the multicast destination nodes participate in the multicast operations. This restriction increases the number of start-up phases and therefore degrades the performance. In the TPM scheme, we allow some additional nodes that are not in the destination set to perform multicast operations. This mechanism can be supported by introducing some additional communication primitives. The absorb operation is modified to support two operations - *temporary absorb* and *permanent absorb*. The temporary absorb operation is defined for the multicast operation performed by an intermediate node that is not in the multicast destination set. The message is temporarily stored in the local memory. In the permanent absorb, the message is passed onto the application after the tail flit is received. Because of the involvement of nodes that are not destinations, it may appear that the performance of these nodes will degrade. However, the reduction of multicast latency overcomes this degradation. More discussions on these issues are detailed in Section 4.5.

4.2.2 Hardware Support

The multicast communication services used by the TPM algorithm are extended from the basic communication services (listed in the previous section) and are summarized as follows:

- Permanent absorb and forward (PAF): This operation is required in the intermediate destination nodes. The message is absorbed in the node while being forwarded to the next node.

- Permanent absorb and retransmit (PAR): An intermediate node performs a PAR operation if it is the last node in the first phase of communication and is also one of the multicast destinations.

- Permanent absorb, forward, and retransmit ($PAFR$): In the TPM scheme, some intermediate nodes need to absorb and retransmit a message to another set of destinations.

If an intermediate node that performs such an operation is also a multicast destination, then the $PAFR$ operation is executed.

- Temporary absorb and retransmit (TAR): In some cases, while using the TPM scheme, we might need to retransmit at an intermediate node that is not one of the destinations. Such intermediate nodes will temporarily store a message and retransmit in the next phase.

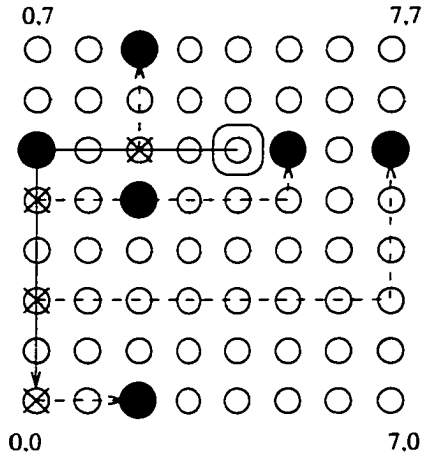
- Temporary absorb, forward, and retransmit ($TAFR$): $TAFR$ is similar to TAR except that the message is also forwarded to the next node while it is being temporarily stored at the current node. The intermediates nodes that are not destinations do the TAFR operation while the last such node of the first communication phase does a TAR operation.

The implementation of some of these communication primitives have been also addressed by previous researchers [41, 95]. The hardware multicast services, discussed above, provide flexibility in path selection to cover more number of destinations in less number of communication steps. The TPM algorithm is designed based on these multicast communication services. Even though some of these operations involve additional overhead at the nodes that are not destinations, a significant performance improvement is obtained due to the reduction of the number of communication phases.

4.2.3 Destination Preprocessing and Multicast Zone

The destination addresses along with the nodes that need to perform multicast operations are sorted in order and encapsulated in the message header. Along with the addresses, the required multicast services are also stored in the header in the form of flags. An example of a message structure showing ten destinations is shown in Figure 4.2. The required multicast operations are also tagged along with the intermediate destination addresses. For example, the node $(0,4)$, which is not a destination, needs to perform the TAFR operation to retransmit the multicast message to the node (2,4) and (5,5). The node (2,4) flag is tagged as PAF indicating to absorb the message while forwarding it to node (5,5). Since the node (5,5) is the last destination in the branch, the ABS operation is assigned.

The multicast zone is defined as the smallest two-dimensional array that includes the source node and all the destinations. The purpose of zoning is to confine a boundary of network resources that can be utilized for the multicast operations. Let the coordinates of the lower left corner be denoted by (l_x, l_y) and the upper right corner be denoted by (u_x, u_y), as depicted in Figure 4.3.



Source node

Destination node

Intermediate multicast node

Des Addr	Flag
2, 5	TAFR
2, 7	ABS
0, 5	PAF
0, 4	TAFR
2, 4	PAF
5, 5	ABS
0, 2	TAFR
7, 5	ABS
0, 0	TAR
2, 0	ABS
DATA	
TAIL	

Figure 4.2 Encapsulation of destination addresses in the header.

An algorithm for defining the multicast zone is shown in Figure 4.4. In step 1, the minimum of all x-coordinates and y-coordinates are calculated and are marked as l_x and l_y, respectively. Similarly, the maximum of x and y coordinates corresponds to u_x and u_y, respectively.

Based on the restrictions of the base routing algorithm, some boundary shapes may need to be modified. The Adjust_shape(l_x, l_y, u_x, u_y) function modifies the invalid periphery and depends on the base routing algorithm. An example corresponding to the XY routing algorithm is described in Section 4.3.

For the ease of explanation of the TPM algorithm, we logically divide the multicast zone into four quadrants. The quadrants are labeled as Q_1, Q_2, Q_3, and Q_4 as shown in Figure 4.3. The source quadrant can be obtained using the algorithm shown in Figure 4.5. To determine the source quadrant, the source coordinates (S_x, S_y) are compared to the center of the multicast zone. The TPM algorithm uses different destination ordering functions based on the source quadrant. The relative positions N_h, F_h, N_v, and F_v are defined for the ease of writing the destination preprocessing algorithm. N_h, F_h, N_v, and F_v denote near horizontal periphery line, far horizontal peripheral line, near vertical peripheral line, and far vertical peripheral line, respectively. These parameters are relative to the location of the source and the multicast zone. After the calculation of multicast zone and source quadrant, the next step of destination preprocessing is the destination ordering and header encapsulation. The destination ordering depends on the underlying routing algorithm, the multicast periphery, and the source quadrant.

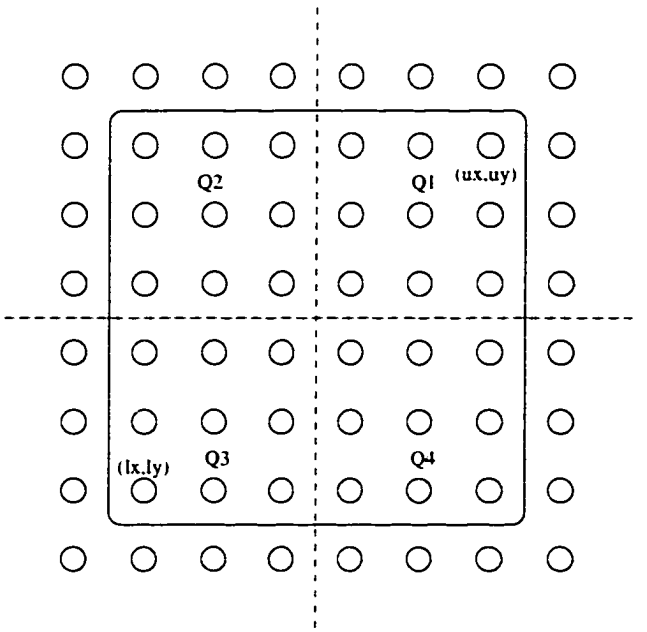


Figure 4.3 Quadrants in the multicasting zone.

4.2.4 Framework of the TPM Algorithm

The formal description of the TPM routing algorithm is given in Figure 4.6. A message is forwarded to the next node if the current node is not a destination node. If the current node is a destination, the associated multicast_flag is used to determine the multicast operation to be performed. The routine for the multicast operation is shown in Figure 4.7. For the TPM algorithm, the implementation of three extra bits are adequate to represent all multicast operations that we have listed in Section 4.2.2. Each bit in the multicast flag represents a unique multicast service: absorb, forward, or retransmit. If the absorb flag is set, the message is copied onto the local node. If the forward flag is set then the message is sent to the neighboring node in the direction of the next destination node. For retransmission, the destination list in the message header needs to be modified. The destination addresses starting from the next destination to first absorb-only destination are removed from the message header. The destination that performs only absorb operation implies the last destination in its branch. The destination addresses removed from the message header are passed on to the retransmission routine along with the data to be retransmitted.

The basic idea behind the proposed algorithm is that, during the first phase, the message is sent to a set of nodes such that all the destinations can be reached in the first or second phases of communication. First, the TPM algorithm defines the *multicast zone* that contains the source node and all the destinations. Next, it defines a path that is taken in the first

Algorithm 4.1: Zone calculation for 2-D ($N \times N$) mesh network.
Input: Destination set $DS[(d_{ix}, d_{iy})]$, source node (S_x, S_y).
Output: The coordinates of the upper and lower corner of the multicast zone (l_x, l_y) and (u_x, u_y).
Procedure:
begin
1. Find the coordinates that cover the source and all the destinations
   $l_x = \min\{DS[d_{ix}], S_x\}$, $l_y = \min\{DS[d_{iy}], S_y\}$
   $u_x = \max\{DS[d_{ix}], S_x\}$, $u_y = \max\{DS[d_{iy}], S_y\}$
2. Adjust the periphery according to the base routing algorithm
   Adjust_shape(l_x, l_y, u_x, u_y)
end.

Figure 4.4 Multicast zoning algorithm.

phase to a specific corner node of the multicast zone. The specific corner node depends on the base routing scheme as is described in Sections 4.3 and 4.4. This path is called the *main path*. After the first phase is completed, some of the nodes along the main path retransmit messages to rest of the multicast destinations. The main path is selected such that it conforms to the base routing algorithm and covers as many destinations as possible. This feature allows the multicast operations to be completed in at most two start-up phases in 2-D meshes. As mentioned earlier, the communication latency in multicomputers is dominated by the start-up latency. Hence, minimization of the number of start-up phases is an important design issue for multicast routing algorithms.

4.2.5 Implementation of the TPM Algorithm

The TPM algorithm may include a few nodes in the multicasting process that are not in the destination set. Thus the performance and execution of processes in these nodes could be affected if they are busy executing any task. It is very difficult to characterize or quantify the exact amount of additional overhead that are incurred by these nodes. Note that the additional nodes are involved only during the first phase. The number of such nodes is very few in most cases. It is quite possible that some of these additional nodes may not be busy and can help in collaborating and speeding up the multicast process. The inference on the additional nodes will be very little compared to the execution time of the job. The involvement of the additional nodes may affect (to some extent) the performance of the specific jobs runing on these nodes. These specific jobs may also be able to utilize the help of other nodes in case they need multicasting operations. Thus, collaboratively, the overall throughput of the system is likely to be enhanced by using the TPM scheme. In most cases, the performance

Algorithm 4.2: Algorithm for determination of the source quadrant.
Input: Multicast zone (l_x, l_y, u_x, u_y), source node (S_x, S_y).
Output: The source quadrant SQ, relative positions of periphery lines (N_h, F_h, N_v, F_v).
Procedure:

Let $midx = u_x - l_x/2$; $midy = u_y - l_y/2$
begin
  if($S_x > midx$) and ($S_y > midy$)
    $SQ = 1, N_h = u_y, F_h = l_y, N_v = u_x, F_v = l_x$
  if($S_x < midx$) and ($S_y > midy$)
    $SQ = 2, N_h = u_y, F_h = l_y, N_v = l_x, F_v = u_x$
  if($S_x < midx$) and ($S_y < midy$)
    $SQ = 3, N_h = l_y, F_h = u_y, N_v = l_x, F_v = u_x$
  if($S_x > midx$) and ($S_y < midy$)
    $SQ = 4, N_h = l_y, F_h = u_y, N_v = u_x, F_v = l_x$
end.

Figure 4.5 Multicast source quadrant algorithm.

improvement obtained by the TPM algorithm may overcome the penalty associated with with the inclusion of the additional nodes.

The TPM scheme will be very suitable for partitionable multiprocessors systems. The above mentioned problems will not be a big factor in case of partitionable systems where tasks are allocated in terms of non-overlapping subsystems like submeshes or subcubes. Support for such partioning mechanisms are provided in contemporary systems like Intel Paragon and Cray's T3D. As we involve nodes that are in the multicast zone, the few additional nodes that are forced to be involved in a multicast operation always belong to the same task. So the inclusion of these additional nodes for multicasting does not affect any other tasks. Above all, the TPM algorithm always tries to minimize the involvement of nodes that are not destinations. The performance improvement obtained using TPM algorithm justifies the temporary involvement of nodes that are not destinations.

The TPM algorithm can be implemented on the MPI standards. The system call arguments in the collective communication provided by the MPI standard consist of the group of destinations. The MPI library receives the list of multicast destinations and performs the destination preprocessing. The destinations are reordered and some additional intermediate destinations based on the TPM scheme are inserted (if necessary). In other words, the destination preprocessing is done at the compilation and the run time. In this approach, users do not have to be aware of the TPM mechanism in developing the application. The additional intermediate destination nodes may be involved in some other operations. If the operations are non-premptive (atomic) then the multicast message waits for the completion of such oper-

Algorithm 4.3: Message routing operation.
Input: Current node (x_1, x_2), Message header(contain the destination set DS and the multicast flags in order of traversal).
Procedure:
begin
1. if the current node address is the head of the destination in the header do
   multicast_operation(multicast_flag, Message header)
2. if the current node is not in the DS.
   forward the message to the next node using the underlying routing algorithm.

end.

Figure 4.6 The TPM algorithm.

ations. Otherwise, the operations, if any, at these nodes are temporarily interrupted and again resumed after the multicast message is routed through.

4.3 Dimension-Order TPM Algorithm

The TPM algorithm can be employed along with several algorithms used for the unicast communications. In this section, we explain the detailed mechanism of the algorithm using a deterministic (dimension-order) routing scheme.

In the first phase of multicast, the multicast message is sent to the farthest corner from the source node. The dimension order XY routing scheme [50] supplies a unique minimal path to the corner node of the diagonally opposite quadrant. The last destination node of the first multicast phase can be (l_x, l_y), (u_x, l_y), (u_x, u_y), or (l_x, u_y) corresponding to the source quadrant Q_1, Q_2, Q_3, or Q_4, respectively. In the second phase, some of the intermediate nodes along the main path retransmit messages to other multicast destinations that are not covered during the first phase. Figure 4.8 shows the dimension order TPM patterns for XY routing on an (8×8) mesh network. The solid lines and dashed lines represent communication in the first and second phases, respectively. In the first step, the source node uses XY routing to travel to the farthest corner node, i.e., to the farthest node of the diagonally opposite quadrant. In the second phase, some of the nodes that have received the message during the first phase send the message to the remaining destinations.

After the farthest corner node has been reached using the main path, all other destination nodes are guaranteed to be covered by the intermediate nodes during the second phase. However, some shape restrictions of the multicast zone need to be maintained for the deterministic routing. With XY routing, the TPM algorithm cannot cover all destinations if the X dimen-

Routine: Multicast_operation().
Input: multicast_flag, Message header(contain the destination set DS and the multicast flags in order of traversed).
Procedure:
begin
1. remove the current destination from the message header.
2. if the multicast_flag includes absorb operation
   -send copy of the received message to the application.
3. if the multicast_flag includes retransmit operation
   -remove the destination addresses from the header (starting from the next destination in DS to the first destination with only absorb operation).
   -pass the removed destination addresses and the content of the message to the retransmission routine.
4. if the multicast_flag includes forward operation
   compute the next direction and forward the message to the next node.

end.

Figure 4.7 Multicast operation routine.

sion is larger than the Y dimension. For example, in the (8×4) mesh shown in Figure 4.9, the last column cannot be reached in two communication phases using the TPM algorithm. To avoid such problems, the Adjust_shape() procedure for XY base routing extends the multicast zone such that $|S_y - F_h| \geq |S_x - N_v|$. We can use a different routing algorithm to route in non-square meshes. For example, the YX routing algorithm can implemented in Figure 4.9. For rectangular meshes, XY routing is used if $X \leq Y$, else YX routing is employed for both unicast and multicast communications. Thus, the dimension-order TPM algorithm can be used for any shape of 2D meshes.

In the TPM destination preprocessing algorithm, selected nodes in the main path are assigned the multicast destinations it needs to cover in the second phase. The formal description of this operation is shown in Figure 4.10. The multicast destinations are divided into four subzones with respect to the source node, as shown in Figure 4.8. The algorithm in Figure 4.11 outlines the subzone labeling scheme with respect to the source node. To explain the algorithm, we use quadrant 1 as an example. For destination nodes in subzone A, $[(l_x \leq D_x \leq S_x), (S_y \leq D_y \leq u_y)]$, the corresponding main path nodes have coordinates: (D_x, S_y). Similarly, for destination nodes in subzone B, $[(l_x \leq D_x \leq S_x), (ly \leq D_y < S_y)]$, the corresponding main path nodes have coordinate (F_v, D_y). The main path nodes for destinations in subzone D can be obtained by mapping the addresses to the F_v column. Some subzone C destinations also need mapping if it overlaps with the subzone D routing paths.

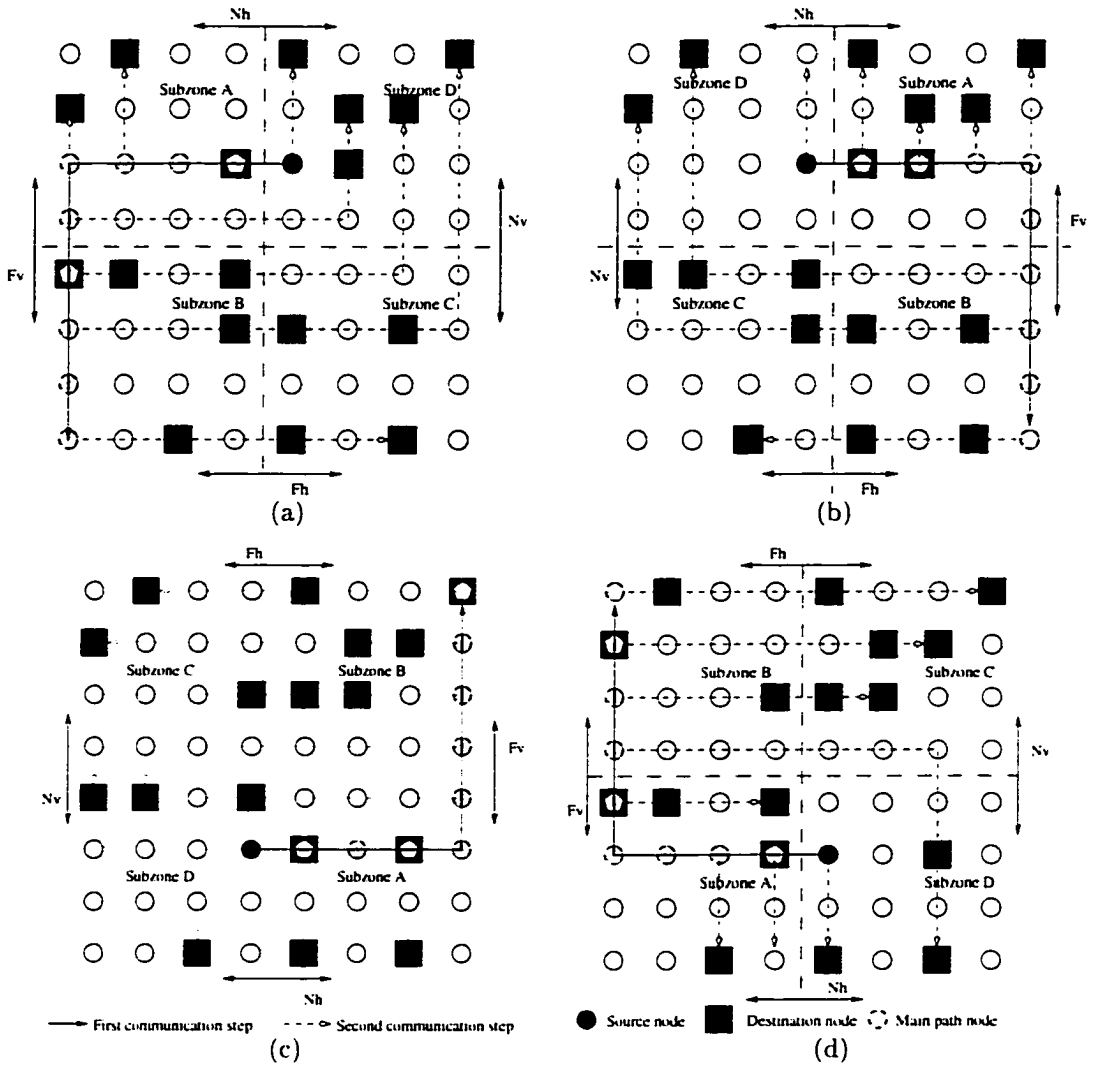


Figure 4.8 Deterministic TPM pattern (a) source quadrant 1, (b) source quadrant 2, (c) source quadrant 3, (d) source quadrant 4.

Lemma 4.1: The deterministic TPM algorithm requires only two communication start-up phases for 2-D mesh networks.

Proof: The message is sent to the opposite diagonal corner of the multicast zone in the first phase. If the multicast zone is defined such that $|S_y - F_h| \geq |S_x - N_v|$, all other destinations in the multicast zone can be reached using the XY paths in the second phase.

Lemma 4.2: The deadlock-free TPM requires four consumption channels in 2-D mesh networks.

Proof: The TPM algorithm allows the absorb and forward operations to be performed in either only X dimension, or only Y dimension, or both X and Y dimensions. In this configuration, at most four consumption channels are required at each node [41].

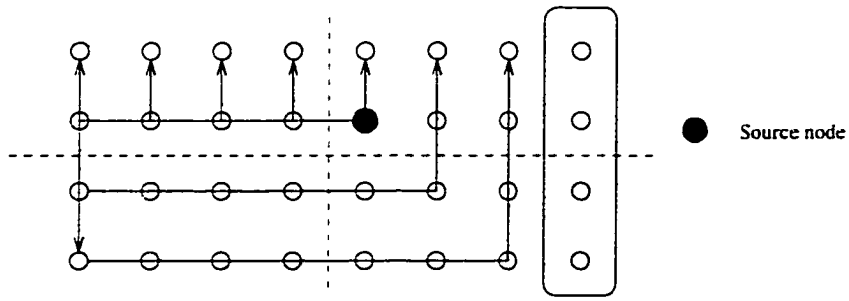


Figure 4.9 Invalid multicast zone for deterministic TPM routing.

4.4 Adaptive TPM Algorithm

The basic framework of the TPM algorithm can be used in conjunction with some adaptive routing schemes (developed primarily for unicast communication). In this section, we show the implementation of TPM on two adaptive routing algorithms, 3P algorithm [60] and PFNF algorithm [32]. Both algorithms belong to the class of fully adaptive algorithms and have been shown to perform better than several other adaptive algorithms proposed in the literature. We must emphasize that many other adaptive routing schemes can be also used for two-phase multicasting. We have discuss the 3P-TPM and PFNF-TPM algorithms as examples to show the effectiveness of adaptive TPM schemes.

4.4.1 3P-TPM Algorithm

The 3P algorithm uses two virtual channels per physical channel. The network is divided into two virtual networks [54]. Fully adaptive routing (no routing restrictions) is implemented in one virtual network, and the dimension order XY routing is implemented in the other virtual network. The XY virtual network provides escape paths without any cyclic dependency to prevent deadlock. Since the TPM algorithm allows the traversal of the destinations conformed to XY paths, it can use the 3P routing algorithm without any modification in the destination preprocessing. In addition, with 3P routing, the TPM algorithm takes advantage of the adaptivity between the destinations along the adaptive routing paths.

An example of alternative paths that can be utilized by the multicast messages under 3P routing algorithm is shown in Figure 4.12. Figure 4.12(a) shows the location of the source and the destinations. Figure 4.12(b) shows the path taken by the deterministic TPM scheme. All the possible paths that can be taken by adaptive 3P-TPM are shown in Figure 4.12(c). It can be observed that the 3P-TPM scheme provides more degree of freedom in forwarding messages to the multicast destinations. The alternative paths allows the multicast messages to avoid

Algorithm 4.4: TPM destinations ordering and header encapsulation.
Input: Destination set $DS[(d_{0x}, d_{0y}), (d_{1x}, d_{1y}), \ldots, (d_{nx}, d_{ny})]$, Multicast periphery (l_x, l_y, u_x, u_y), source node (S_x, S_y), relative position (N_h, F_h, N_v, F_v).
Output: Encapsulated destination message header.
Procedure: /* Let $MS[(m_{0x}, m_{0y}), (m_{1x}, m_{1y}), \ldots, (m_{nx}, m_{ny})]$ be the set of main path nodes associated with the DS. DM array contains destinations associated with the main path nodes. $MS[< (d_{0x}, d_{0y}), (m_{0x}, m_{0y}) >, < (d_{1x}, d_{1y}), (m_{1x}, m_{1y}) >, \ldots, < (d_{nx}, d_{ny}), (m_{nx}, m_{ny}) >]$. */
begin

1. for all elements in the DS
   if Destination_subzone(d_{ix}, d_{iy}) = subzone A
      $m_{ix} = d_{ix}$, $m_{iy} = S_y$
   if Destination_subzone(d_{ix}, d_{iy}) = subzone B
      $m_{ix} = F_v$, $m_{iy} = D_{iy}$
   if Destination_subzone(d_{ix}, d_{iy}) = subzone C
      $m_{ix} = F_v$
      if($| d_{iy} - S_y | < | N_v - S_x |$)
         if $N_h > S_y$, $m_{iy} = d_{iy} - max(| S_x - d_{ix} |, | S_y - d_{iy} |)$
         else $m_{iy} = d_{iy} + max(| S_x - d_{ix} |, | S_y - d_{iy} |)$
      else $m_{iy} = d_{iy}$
   if Destination_subzone(d_{ix}, d_{iy}) = subzone D
      $m_{ix} = F_v$
      if $N_h > S_y$, $m_{iy} = D_{iy} - | D_{ix} - S_x |$
      else $m_{iy} = D_{iy} + | D_{ix} - S_x |$
2. $DM = [< DS, MS >]$
3. Sort(DM) according to main path in each quadrant using MS and DS as the sorting keys.
4. Encapsulate the destination addresses along with the associated main path nodes that are not in the destination set into the header in their order of traversal.
5. Assign multicast_flags to the destination addresses in the header.

end.

Figure 4.10 Header encapsulation process in deterministic TPM algorithm.

congestions in the network. Multicasting operations can create a temporary hot spot area and an adaptive base routing scheme can be employed to reduce the congestions associated with the hot spot. It should be noted that all the destinations are reached within two start-up phases using the 3P-TPM algorithm.

4.4.2 PFNF-TPM Algorithm

Similar to the 3P algorithm, the PFNF algorithm [32] requires two virtual channels and logically divides the network into two virtual networks. Different turn models [77] are assigned to each virtual network. A specific combination of the turn model is selected such that there is an equal probability for selecting one of the two alternative channels. The PFNF scheme

Algorithm 4.5: Destination_Subzone (): Calculate subzone label for each destinations.

Input: Destination (d_{ix}, d_{iy}). Multicast periphery (l_x, l_y, u_x, u_y), source node (S_x, S_y). relative position (N_h, F_h, N_v, F_v).
Output: subzone label of the destination.
Procedure:

```
begin
    if( S_x ≤ d_ix ≤ F_v or S_x ≥ d_ix ≥ F_v)
        if(S_y ≤ d_iy ≤ N_h or S_y ≥ d_iy ≥ N_h)
            return subzone A
        if(S_y < d_iy ≤ F_h or S_y > d_iy ≥ F_h)
            return subzone B
    if( S_x < d_ix ≤ N_v or S_x > d_ix ≥ N_v)
        if(S_y < d_iy ≤ F_h or S_y > d_iy ≥ F_h)
            return subzone C
        if(S_y < d_iy ≤ F_h or S_y ≥ d_iy ≥ F_h)
            return subzone D
end.
```

Figure 4.11 Labeling subzones with respect to the source node.

employs one such combination where the Positive-First algorithm is used in one virtual network and the Negative-First algorithm in the other. This combination of the turn model is deadlock-free and have been shown to demonstrate better performance than the other routing algorithms for 2-D meshes [32]. The PFNF algorithm provides routing paths with no routing restriction to $(+X, +Y)$ and $(-X, -Y)$ directions in both virtual networks. This routing property is well suited to the TPM approach. It reduces the number of messages and the number of additional nodes required for the multicast operations.

To utilize the adaptivity provided by the PFNF algorithm, during the first phase, the multicast message is sent to (l_x, l_y) or (u_x, u_y) corner of the multicast zone based on the relative distance from the source node. The main path is selected such that it covers the maximum number of multicast destinations while traveling to the farthest corner node ((l_x, l_y) or (u_x, u_y)), called multicast corner. The XY conform paths are used if the Y dimension of the multicast zone is larger than the X dimension of the multicast zone. Similarly, the YX conform paths are used if the X dimension of the multicast zone is larger than the Y dimension of the multicast zone. As shown in the example in Figure 4.13(a), the source nodes that reside in the upper tridiagonal area send a message to the lower left corner of the multicast zone and vice-versa.

We propose a greedy algorithm to group the multicast destinations based on the adaptive paths. The greedy scheme searches for the multicast destinations using the shortest paths from

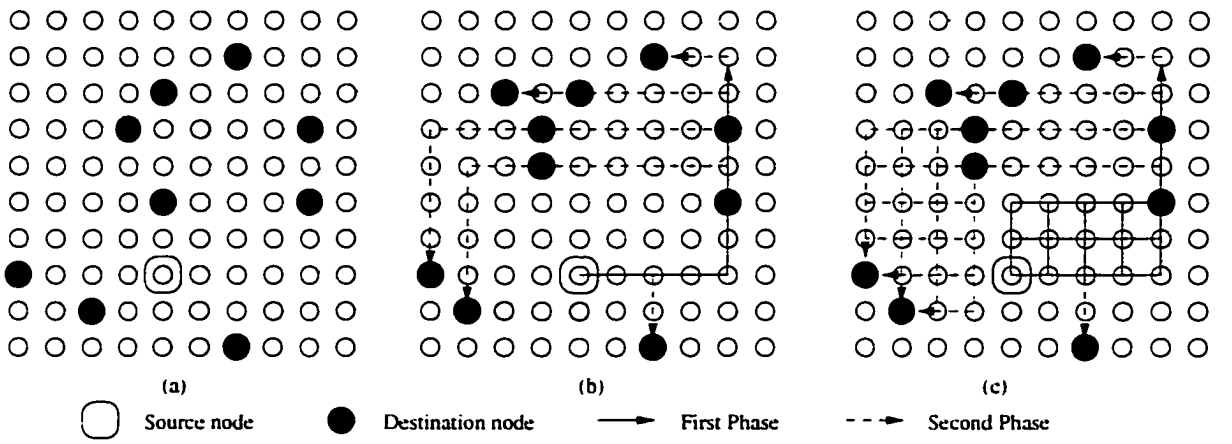


Figure 4.12 Adaptive path supplied by TPM algorithm using the 3P routing algorithm (a) destination locations. (b) deterministic TPM paths. (c) Adaptive TPM paths.

the multicast corner to the opposite corner. An example of the multicast path is shown in Figure 4.13 (b). The second phase multicast paths are first formed based on the minimal path using the greedy scheme. The main path is then traced through the starting nodes of these second phase paths. Note that some new starting nodes in the second phase paths need to be added to conform to the minimal path. If the number of multicast destinations is equal to the number of nodes in the multicast zone, the multicast path is the same as the dimension-order TPM.

Using the PFNF-TPM scheme, the performance is enhanced by two ways. First, we exploit the adaptivity in sending multicast messages. Second, we can utilize the common multicast paths that can cover maximum number of destination. Using adaptive routing, the multicast paths can cover the multicast destinations that conform to the adaptive path. If the underlying routing algorithm restricts some messages to the deterministic paths, the multicast paths are also restricted to those paths as in the case of dimension order or 3P routing algorithms. As mentioned earlier, the PFNF algorithm have no restriction in the $(+X, +Y)$ and $(-X, -Y)$ directions. We can create the multicast paths based on any minimal routing paths in these directions. The number of messages in multicast operations is reduced for sparse distribution of destinations since the destinations which are not in the $x - y$ path can be group together in the PFNF-TPM algorithm. For dense distribution of destinations, the PFNF-TPM will have similar performance to that of the dimension order TPM.

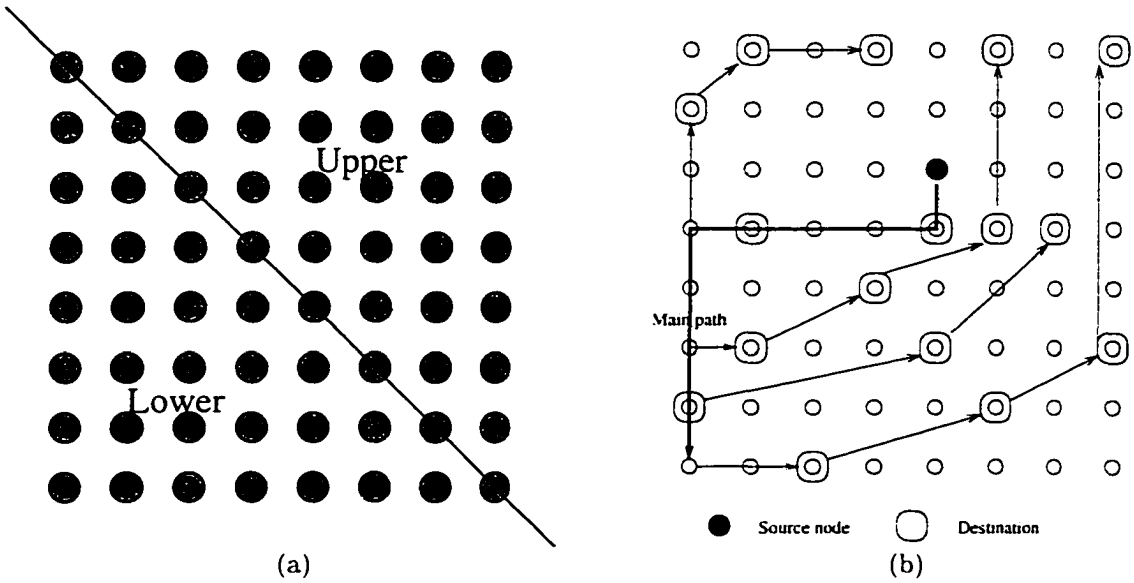


Figure 4.13 (a) PFNF multicast zone (b) greedy algorithm for the multicast destination grouping.

4.5 Performance Evaluation

To investigate the performance of the TPM routing algorithm, the simulator was customized to support multicasting wormhole routed mesh networks using the CSIM event driven simulation platform [109]. The simulation environment and the performance results of the deterministic as well as adaptive TPM routing schemes are described in this section.

4.5.1 Simulation Model

A simulation model was developed for a 16 × 16 mesh. Fixed size packets of 20 flits per packet was assumed. To improve the traffic flow, we have considered two virtual channels per physical channel with one flit buffer per virtual channel. The time required to transfer one flit from a node to its neighbor is assumed to be equal to $30ns$ which is mapped to one time unit of the simulator. The start-up latency is set to $1\mu s$ (33 time units). In multicast operations, multiple messages can be generated in the same communication phase. To model such situations, the succeeding message start-up time is set to $240ns$ (8 time units) because the generation of succeeding messages takes less time than the generation of the first message. If a multicast algorithm requires absorb and retransmit operation, additional start-up latency is used to capture the retransmission overhead for fair comparison.

Some multicast operations require header processing at the intermediate nodes where an

additional overhead needs to be taken into account. The routing decision usually takes 1.5 to 2 times the time required for transferring the data flit [56]. Using this data, the routing decision is assumed to take $60ns$ (2 time units) without the header modification. None of the earlier reported works have considered the overhead associated with header modification in multicast communications. We assumed $90ns$ (3 time units) routing delay for the routing decision and header modification.

For our experiments, the source node is randomly chosen for each operation. The multicast destinations are assumed to be uniformly distributed over the network. We have considered multicast latency, average additional traffic [110], average number of hops, and average maximum number of hops as multicast performance metrics. The multicast latency is defined as the time between the initiation of multicast operation and the time when the tail of the multicast message reaches all the destinations. The additional traffic reflects the amount of network resources that are used to complete the multicast operation. A physical channel occupied for one time unit is counted as one traffic unit. The average number of hops is defined as the mean value of the number of hops required to reach the destinations. The average number of hops will reflect the length of paths used by the multicast messages. Similarly, the maximum number of hops is defined as the average value of the maximum number of hops in each multicast communication. The maximum hop distance could have a large impact on the multicast latency as the last destination may be the one that may incur the largest number of hops.

4.5.2 Performance under Contention-Free Communication

We first present the performance results under the contention-free condition to show the effectiveness of the proposed TPM algorithm for reducing the multicast latency through a reduction of the start-up phases. The average performance metrics are calculated from 1000 multicast operations. The number of destinations is varied from 20 to 250 nodes. The performance of the proposed TPM algorithm is compared with the dualpath [95], column-path [96], and unicast-based multicast algorithms. In the unicast-based multicast algorithm, the source node sends one message to each destination in the multicast destination set. This algorithm represents an implementation without any additional hardware or software requirements.

4.5.2.1 Performance Results of the Deterministic TPM Algorithm

We obtained simulation results for both one-port and all-port models. In an one-port model, there is only one consumption channel and only one injection channel. Since the simulations were run in a contention-free environment, there is no deadlock problem. Figure

4.14(a) shows the multicast latency versus the number of destinations. The TPM algorithm has the lowest latency and is nearly constant with the increase in the number of destinations. The multicast latency using dualpath algorithm increases when the number of destinations increase from 20 to 100 and thereafter flattens out. The column-path and the unicast based multicast perform poorly because they involve more number of communication phases when the number of destinations increase.

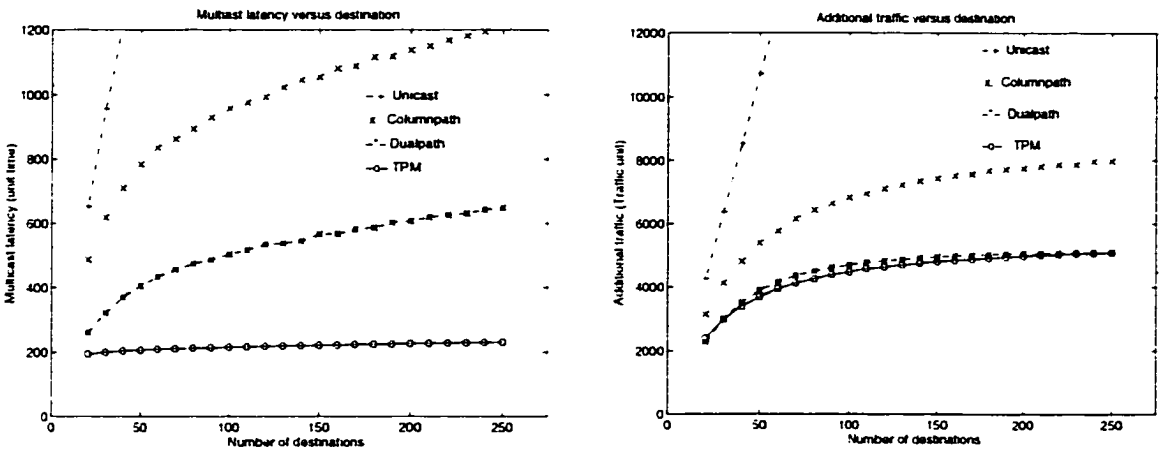


Figure 4.14 Performance comparison of multicast algorithms for an one-port model (a) communication latency, (b) additional traffic.

The results of additional traffic with respect to the number of destinations are shown in Figure 4.14 (b). The TPM algorithm requires a little less network resources than the dualpath algorithm. More network resources are required to complete the same multicast operation using unicast operations or the column-path algorithms. The difference is attributed to the fact that, with the same number of destinations, the dualpath and TPM send the multicast messages in paths that cover more number of destinations.

Figure 4.15 (a) shows the average number of hops versus the number of destinations. The column-path and unicast-based multicast algorithms have the smallest average number of hops since the algorithms follow the minimal deterministic paths. The TPM algorithm requires approximately 10 extra hops compared to the minimal paths. However, the number of hops for the TPM algorithm is constant as the number of destinations increase. For dualpath, the number of hops increases with the number of destinations.

In an all-port model, the number of consumption channels and the number of injection channels are equal to the total number of virtual channels in the router. The performance

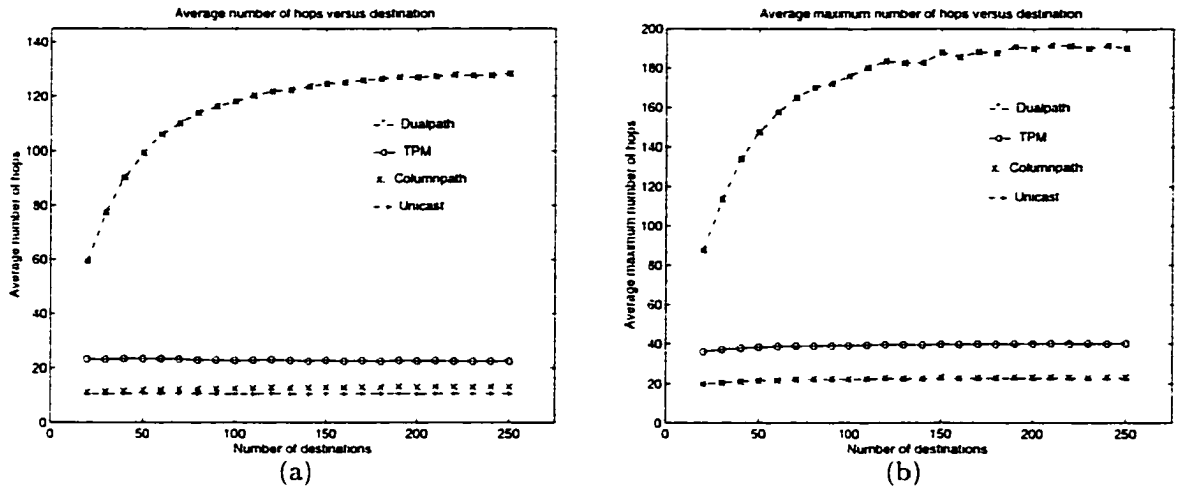


Figure 4.15 Performance comparison of multicast algorithm for an one-port model (a) average number of hops. (b) average maximum number of hops.

comparisons for an all-port model are shown in Figure 4.16. The trends are observed to be the same as that of the one-port model. However, the performance of the column path and unicast-based multicast performance are improved due to more number of messages being sent concurrently. The column-path algorithm performs as good as the dualpath algorithm. These results also show that the number of injection channels have a strong impact on the performance for the class of multicast algorithms that generate several messages at the same time.

4.5.2.2 Performance Results of Adaptive TPM Algorithms

The performance results of adaptive TPM algorithms in a contention-free environment is shown in Figure 4.17. Figure 4.17(a) displays the multicast latency comparison of the PFNF and 3P based adaptive TPM schemes to that of the deterministic XY-TPM scheme. It is observed that the 3P and the XY based TPM algorithms incurs lower latency compared to the PFNF-TPM algorithm in a contention-free case. Figure 4.17(b) shows the comparison of the additional traffic for the same algorithms as that of Figure 4.17(a). The additional traffic generated by using the PFNF-TPM schemes is lower than that of the XY-TPM or the 3P-TPM schemes. The PFNF-TPM uses the adaptivity provided by the base routing algorithm to cover more multicast destinations. Therefore the amount of additional traffic is less than all other algorithms. However, as the number of destinations increases, the difference is diminished. Higher traffic may increase the average message latency in a network environment when the

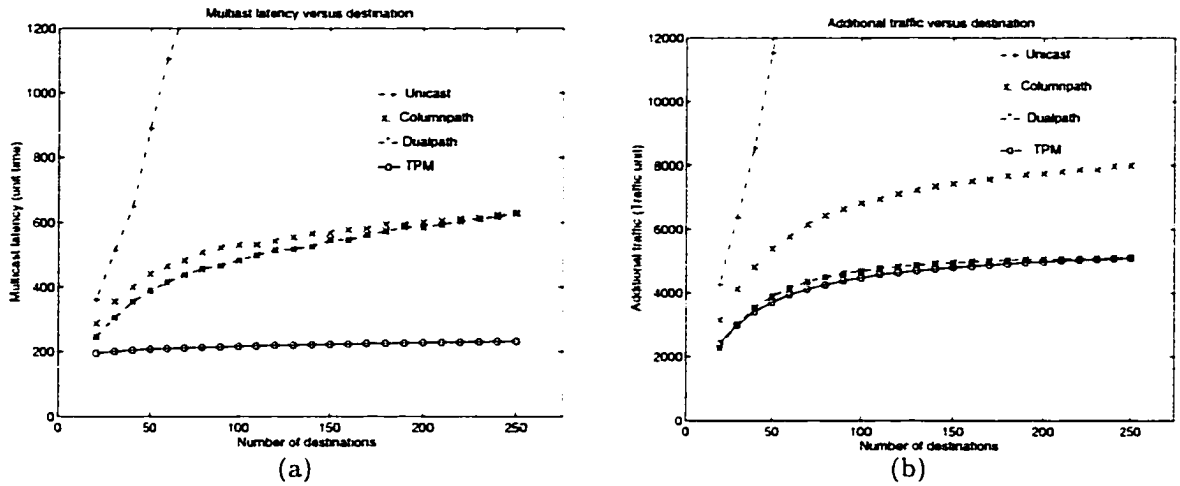


Figure 4.16 Performance comparison of multicast algorithms for an all-port model (a) communication latency, (b) additional traffic.

effect of contention is factored in. In fact, this speculation is validated in the next subsection.

4.5.3 Performance of Unicast and Multicast Traffic with Contention

In real applications, the multicast message might compete for network resources with the unicast messages or other multicast messages in the network. To examine the performance of the TPM algorithms in such situations, we have simulated the multicast traffic together with the unicast traffic. We consider 90 percent unicast messages and 10 percent multicast messages. The destinations are assumed to be uniformly distributed. The average number of destinations (D) in the multicast operation is set to 32 with a standard deviation of 15. The minimum and maximum number of destinations is 2 and 250, respectively. As described in [108], the unicast inter-arrival time for each node is equal to $\frac{N}{X \times 0.9}$ where N is the number of nodes in the system and X is the system throughput. The multicast inter-arrival time is defined as $\frac{(N \times D)}{(X \times 0.1)}$. Our model assumed one injection channel and eight consumption channels implemented in the router. The results are computed by averaging the measures after the delivery of 140,000 messages. The performance measures of the first 40,000 messages are ignored to avoid the transient effects.

The performance results under mixed traffic(unicast and multicast) are given in Figure 4.18. The unicast communication latency are shown in Figure 4.18 (a). Among all algorithms, the TPM algorithms demonstrate the best performance followed by the column-path, unicast-based multicast, and dualpath algorithms, respectively. The multicast latency curves are shown in Figure 4.18 (b). The TPM scheme with XY, 3P, and PFNF algorithms exhibits low latency.

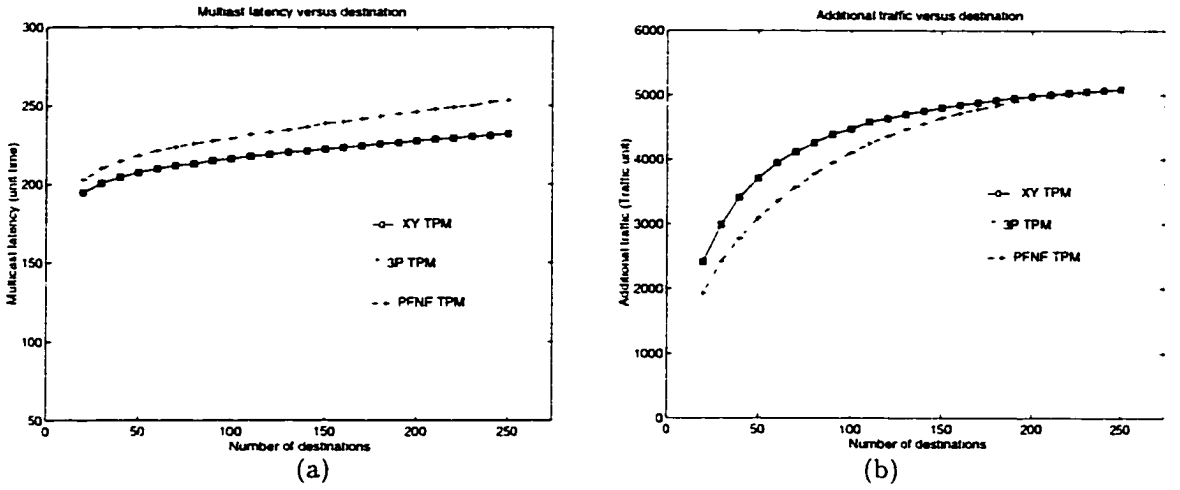


Figure 4.17 The effect of adaptivity for an all-port model (a) multicast latency, (b) additional Traffic.

The dualpath algorithm performs well in the low throughput but the latency increases rapidly in higher throughput. The communication latency increase slowly for column-path but it is severely degraded by the start-up latency and queuing delay at the source node.

We have simulated the TPM algorithm under three base routing algorithms, XY dimension order, 3P, and PFNF. The latency curves shown in Figure 4.19 indicate that the adaptivity does improve the performance for both unicast and multicast messages. Note that the number of multicast messages spawned for each multicast operation is the same for both XY and 3P routing algorithms. With less restrictions in PFNF adaptive routing algorithm, the destination grouping can be designed such that less number of multidestination messages are spawned.

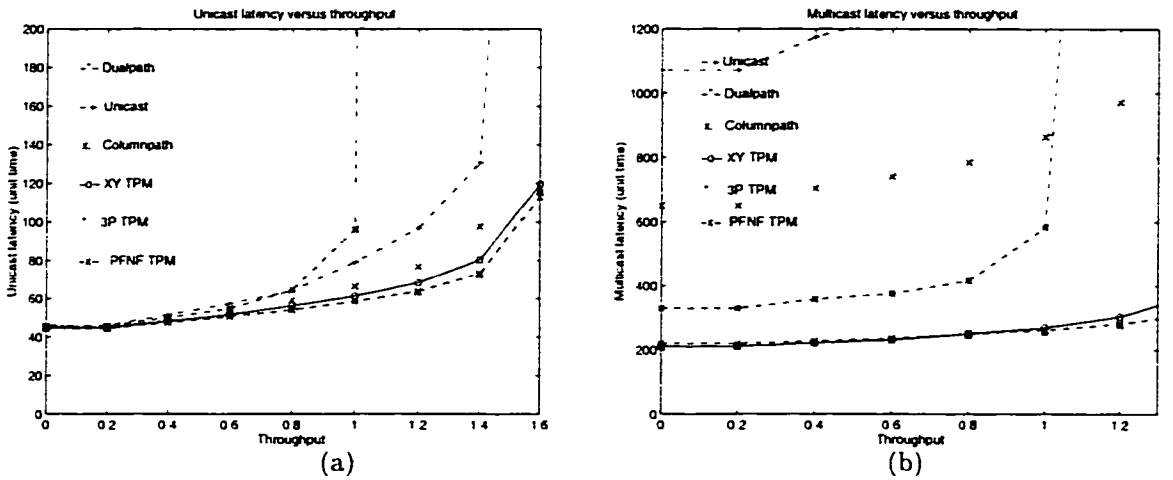


Figure 4.18 Performance results for mixed uniform and multicast traffic a) unicast communication latency, b) multicast communication latency.

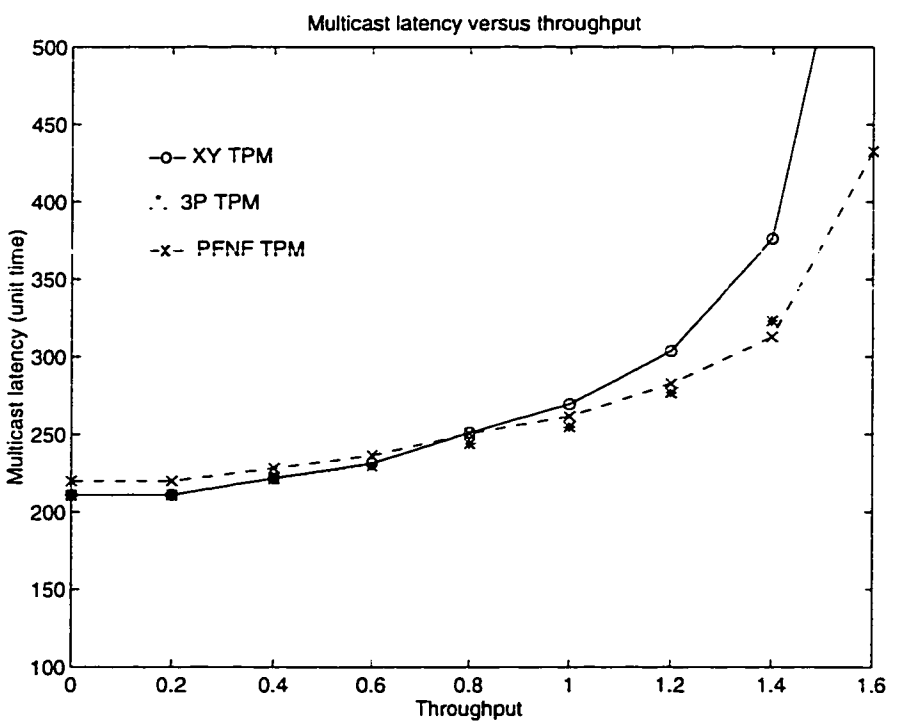


Figure 4.19 Effect of adaptivity for the mixed uniform and multicast traffic.

5 MULTICASTING IN MULTISTAGE INTERCONNECTION NETWORKS

Most of the contemporary parallel systems are designed to efficiently support unicast communication. These systems support multicast communication by sending one unicast message to each of the multicast destinations. A detailed treatment of the performance of wormhole switching in MINs can be found in [49]. Efficient software multicast algorithms have been proposed for bidirectional [86] and unidirectional MINs [87]. These algorithms rely only on the underlying unicast communication and do not need any hardware modification. The source node sends unicast messages to one multicast destination in the first phase. A divide and conquer strategy is used to improve the communication latency. In subsequent phases, some destinations are assigned to act like source nodes in addition to the original source node to send messages to destinations that are yet to be reached according to a predefined multicast tree structure. The number of communication phases required in binomial software multicasting schemes for d destinations is $\lceil \log_2(d+1) \rceil$. Each phase incurs a startup latency which is a major proportion of the total communication latency.

In [92], a multinomial tree structure for the source and the destinations for multicasting is proposed. The proportions of startup latency, network latency and reception latency that comprises the communication latency is different from system to system. The multinomial multicasting steps are tuned based on the communication network parameters to further optimize the communication latency. However, the software-based approaches have a higher latency as they involve several communication start-up phases and do not exploit the concept of sending the message concurrently to cover as many destinations as possible. To further improve multicasting performance, the multicast operations need to be supported at the lower level [29, 30]. The lower level multicast supports can be implemented as additional functions in network interface units and/or dedicated hardware in the switching elements. Hardware multicasting allows sharing of network resources to cover multiple destinations which reduces both network traffic and the number of communication phases. Hardware-based multicasting algorithms can be classified as path-based or tree-based. In the path-based approach, improvement in performance is exploited from the destinations that can share a common path. Several path-based

multicasting algorithms have been proposed for the direct networks [34, 41, 95, 96]. Most of these works focus on designing deadlock-free multicast algorithms for strict wormhole switching[1]. The path-based approach is not convenient for MINs because the messages pass through intermediate switches that are not connected to any processing node. Hence, the path-based multicasting scheme in MINs does not reduce the amount of traffic injected into the network and has only small differences to to the scenario of sending one message to each destination, without interfering with the processors. Also, it has been shown in [98] that the path-based multicasting approach is not only inefficient but also can create deadlocks in MINs.

MINs inherit the tree topology which can be effectively exploited to support multicast communication. Unfortunately, the tree branching operations create additional resource dependencies that make the wormhole networks more susceptible to deadlock configurations. In tree-based multicasting, a multihead message is sent out of the switch and the multiple headers are forwarded either synchronously or asynchronously. The tree-based multicasting scheme proposed by Chiang and Ni [98] requires the multicast headers in different branches to be forwarded synchronously from one stage to the next (called synchronous worm). Synchronous movement is required to prevent deadlocks. Priority schemes are required to prevent deadlock and starvation at the switches. To synchronize all multicast branches, a feedback mechanism that traverses the whole multicast tree is required. The synchronous multicasting method not only requires a considerable amount of hardware modifications but also suffers from the synchronization overhead in terms of additional latency.

Another approach of tree-based multicasting is the asynchronous replication scheme. In asynchronous tree-based multicasting, the asynchronous worm allows multiple headers to be forwarded independent of each other. This approach matches well with the real implementation of switching elements. If a branch of multicast message is blocked, the other branches can be forwarded asynchronously by introducing bubbles in their paths. The asynchronous worm is preferred over the synchronous worm because of its ease of implementation. However, it is more prone to deadlocks. As observed in [98], deadlock prevention in asynchronous multicasting approaches is very difficult while using wormhole routing without large buffers at each of the switching elements [98]. An approach to implement the asynchronous multicasting through the use of large buffers at each switch to prevent deadlocks is reported in [99, 100]. In this work, we propose an efficient and simple deadlock prevention technique for asynchronous multicasting MIN systems. The methods for calculating the size of buffer required to prevent deadlocks in bidirectional MINs are reported in [100].

The multicast message header for an arbitrary destination set needs to contain a lot of

[1]Strict wormhole switching allows only a small amount of buffer space at the input ports.

information for selecting the multicast paths. This factor can not only increase the message length but also can make the header decoding and routing complex which, in turn, degrades the switch performance. To minimize routing operations at the intermediate switches, the multiport encoding scheme [101] uses the same routing tag for different multicast branches at the same stage. This method allows limited tree operations to a set of destinations. Thus, completion of multicast operation may require several communication phases depending on the location and spread of the destinations. As in the case of software multicasting technique, multiple communication phases can increase the communication latency. To prevent deadlocks, the multiport encoding scheme requires buffers at the switches corresponding to the maximum packet size at each input port to break the branch dependencies.

We first investigate the deadlock problems associated with the tree-based multicasting in MINs. The deadlock configurations in the tree-based multicast in MINs are analyzed. A switch grouping technique is developed to analyze the behavior of the deadlocks. Based on the study, an asynchronous tree-based multicasting (ATBM) scheme is proposed for multicasting in MINs using wormhole switching technique. To prevent deadlocks, the switches are grouped using a grouping algorithm, and multiple tree operations are serialized within the groups. The delay incurred due to the serialization is overcomed due to the reduction in overall network latency using the proposed scheme. The serialized deadlock prevention is simpler than the synchronous replication approach [98]. The hardware modifications required for the proposed ATBM scheme is associated with the coordination of the switches in the same group. We have discussed the implementation of the ATBM algorithm for both unidirectional and bidirectional MINs.

The proposed scheme is different from the previously proposed asynchronous schemes [101] in the sense that the multicast communications can be completed in a single start-up phase and only small buffers are required at each of the switches. The performance evaluation is carried out through simulation experiments. The results show that our approach performs significantly better than the software multicasting schemes while incurring low hardware overheads compared to the previously proposed multicasting schemes.

This chapter is organized as follows. Section 5.1, presents the system model for the wormhole routed MINs. The deadlock problems in MINs and the switch grouping algorithm are explained in Section 5.2. The proposed ATBM algorithms are presented in Section 5.3. The simulation results are given in Sections 5.4.

5.1 Preliminaries

Switching elements are the main components of a MIN. Switch architecture in Figure 2.5 is shown again in Figure 5.1. The switch comprises of a set of input channels with the associated buffer(s), a crossbar interconnection, a central buffer space, and a set of output channels. There are a few alternatives for the location of buffers in the switch. If the buffer size is small compared to the average message size, the performance difference because of the location of buffers is not significant. Upon reception of the header, the control circuit examines the routing information and performs the routing operations accordingly. If the next buffer is empty, the header is forwarded to the next stage. The data flits follow the same path. After the tail flit is forwarded, the buffer is released. In this work, UNI-MIN and BI-MIN are used as a basis of our discussion.

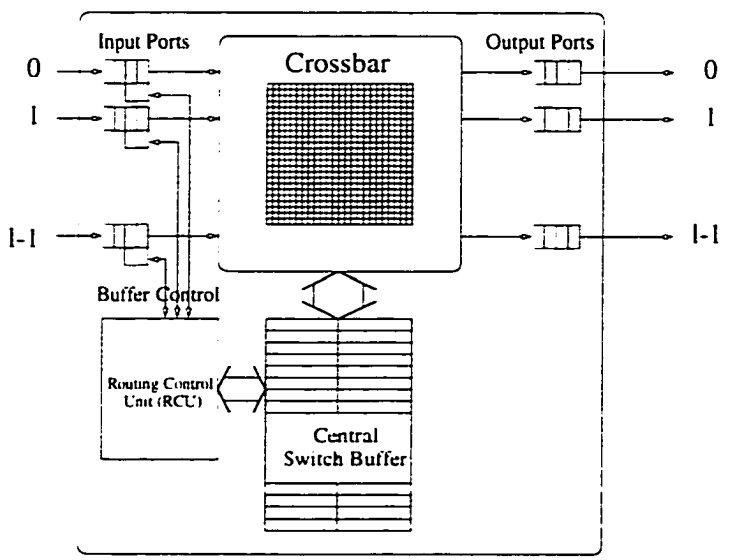


Figure 5.1 Switch architecture.

A broad class of unidirectional MINs that belongs to the family of *Delta* networks has a self routing property by which the routing path is decided on the basis of the destination address tag. Baseline and butterfly topologies belong to this class. Figure 5.2 (a) shows examples of destination-tag routing from [0010] ($S1$) to [1010] ($D1$), and [1100] ($S2$) to [1000] ($D2$) for the unidirectional baseline network. The next output port is decided on the basis of the destination bit corresponding to the current stage.

The routing algorithm for the unicast messages in butterfly BI-MIN uses the turnaround routing [49]. The stage T at which the message turns around is first calculated using the least common ancestor of the source and destination addresses ($[s_{n-1} \ldots s_1 s_0]$ and $[d_{n-1} \ldots d_1 d_0]$).

The turnaround stage T is defined as the first bit position where the s_i and d_i are different from the left hand side. The routing operation is divided into two phases. In the first phase, a message is routed freely to the stage T. After reaching stage T, the message turns around and is routed to the destination using the destination address. Figure 5.2 (b) shows two examples of turnaround routing from [0010] ($S1$) to [1010] ($D1$), and [1100] ($S2$) to [1000] ($D2$). From $S1$ to $D1$, the least common ancestor of the source and destination addresses is equal to 3. The message is forwarded to stage 3 and then turns back to reach its destination. Similarly, the message turns around at stage 2 for the communication between $S2$ and $D2$. Since the unicast messages request for the network resources acyclically, the turnaround routing is deadlock-free.

To support tree-based multicast operations, a replication mechanism is implemented. The incoming flit is replicated and forwarded to multiple output ports in the branching operation. A flit can be copied to multiple output ports in parallel or in sequence. The switch architectures that support tree operations have been studied in [111]. Multiple destinations can be accomplished by performing the tree operations at the appropriate switches. Multiple headers created from the tree operation can be forwarded synchronously (synchronous worm) or asynchronously (asynchronous worm) [98]. All headers in the multicast operation are advanced concurrently at the same stage in a synchronous worm. If one header is blocked, all headers in the other branches are stalled. In contrast, the asynchronous worm allows all headers to be forwarded independently to the nodes of the multicast destination set. If one branch is blocked, the back-pressure stops the message flow at the switch that performs the replication and branch operation. Since the movement of the message stops at the tree stem, bubbles (holes) are introduced in the branches in which the headers keep moving toward their destinations. The asynchronous approach may consume more network resources due to the bubbles. However, the asynchronous approach is easier to implement compared to the synchronous approach because the synchronous worm requires feedback mechanism that traverses the entire multicast tree.

Figure 5.3 shows possible tree operations in MINs. The switch broadcast operation for UNI-MINs is shown in Figure 5.3 (a). The degree of tree operations ranges from 2 to b. In BI-MINs, the message routing operations involve three steps: routing freely to the turnaround stage, turning around, and routing toward destination. The basic routing operations in BI-MINs consist of forward and turnaround operations. The tree operation can be performed in the forward phase as shown in Figure 5.3 (b), or in the backward phase as shown in Figure 5.3 (c). Turnaround to multiple ports is also considered as tree operations. Figure 5.3 (d) shows the tree turnaround with forwarding, and Figure 5.3 (e) shows tree turnaround without forwarding.

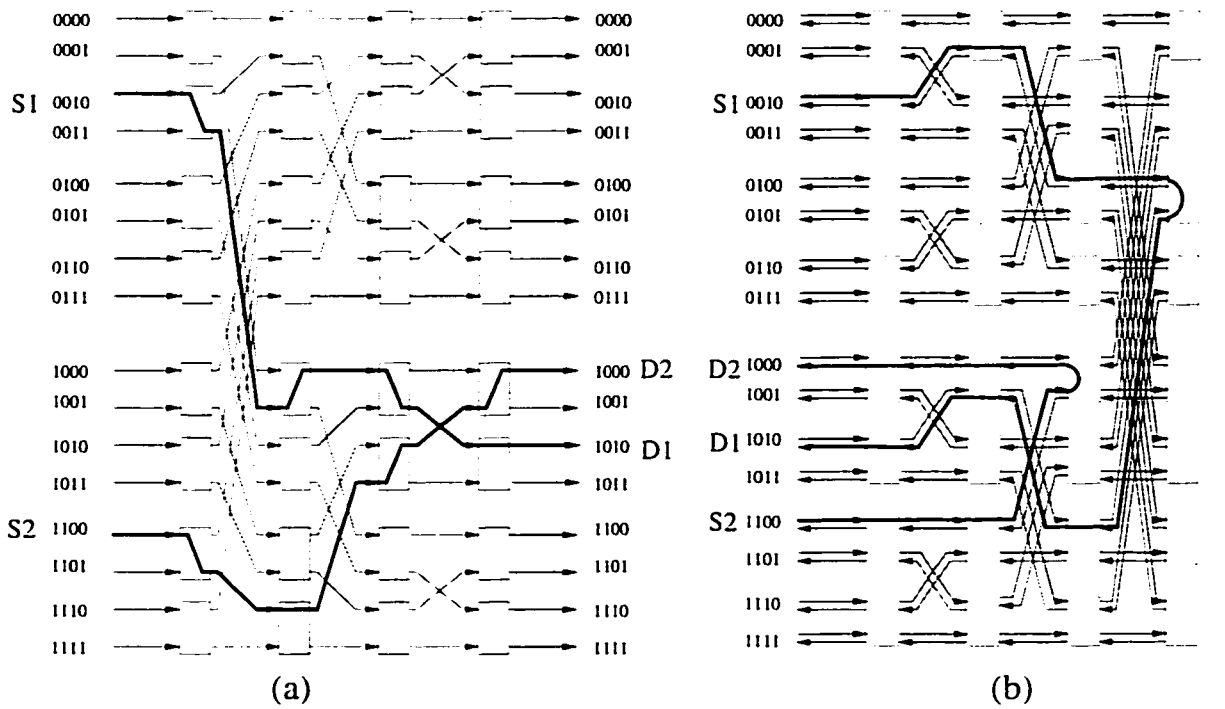


Figure 5.2 Routing in MINs (a) unidirectional unicast routing (b) bidirectional unicast turnaround routing.

The tree operations are carried out as necessary at the appropriate switches. The tree-based multicasting in UNI-MINs is straightforward since it involves transmission only in one direction. The tree-based multicasting mechanism in BI-MINs can be classified based on the turnaround points [100]. Figure 5.4 (a) shows the tree-based multicast operation from the source (S) to a set of destinations (D) where multiple turnaround points are involved. The multiple turnaround approach uses the turnaround with forwarding to complete the multicast operation. Another method of implementation of the tree-based multicasting is shown in Figure 5.4 (b). In this case, a multicast message is routed to the turnaround stage and performs the tree operations in the last step of the forward routing operation. In other words, only one turnaround is allowed and tree operations are allowed only after the turnaround. This approach requires only the backward tree operations and the tree turnaround without forwarding. The amount of traffic used in the multiple turnaround scheme is lower than the single turnaround scheme. The impacts of these approaches on the deadlock prevention are discussed later in this chapter.

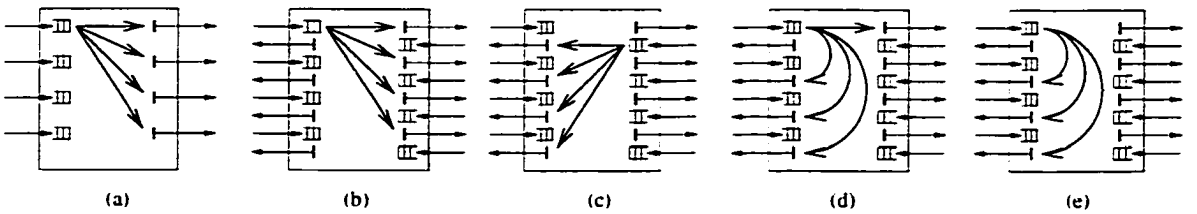


Figure 5.3 Multicast tree operations in MIN switches.

5.1.1 Message Format and Routing Tag

Implementation of a routing algorithm requires the messages to be formated in a specific manner. The system should support several classes of communication operations including unicast and multicast. These classes are distinguished by the message type field. For unicast messages, the switching elements can utilize the destination address to determine the next output port. The routing information in this case is the destination address for UNI-MINs and turnaround stage and the destination address for BI-MINs which directly translates to one of the output ports of the switch.

The routers at the switches require additional routing information to perform arbitrary tree-based multicast operations. Several multicast header encoding techniques are presented in [94]. A simple encoding scheme is the *all-destinations* encoding. All multicast destination addresses are encapsulated in the multicast header. The message preparation and the start-up phases are simple. When the header is received, the control unit decodes it and supplies a set of output ports to the current switch. The message header has to contain all multicast destination addresses which can be very long for the large systems. Therefore, the complexity of decoding activities can degrade the system performance. Another extreme of the encoding technique is tag routing. The set of output ports to be used at each switch is encoded in the tag routing scheme. The routing decision is made at the source node. The routing decision at the switch is simple; an incoming message is forwarded to the specified output port(s). However, the tag routing is not convenient for arbitrary multicasting since the number of switches involved may be excessively large. The message header processing could be overly complex.

A hierarchical routing tag encoding can also be used to reduce the decoding complexity. In tree-based multicasting, the switch has to support an operation that replicates and forwards the flits to the different output ports concurrently. The switch uses the routing tag to specify the next output ports for the message. To support multicast, the routing tag is represented by b bits for a $b \times b$ switch. Figure 5.5 (a) shows a broadcast tree for a MIN that uses 2×2 switches. The format of the routing tag associated with the switches is shown in Figure 5.5 (b). The

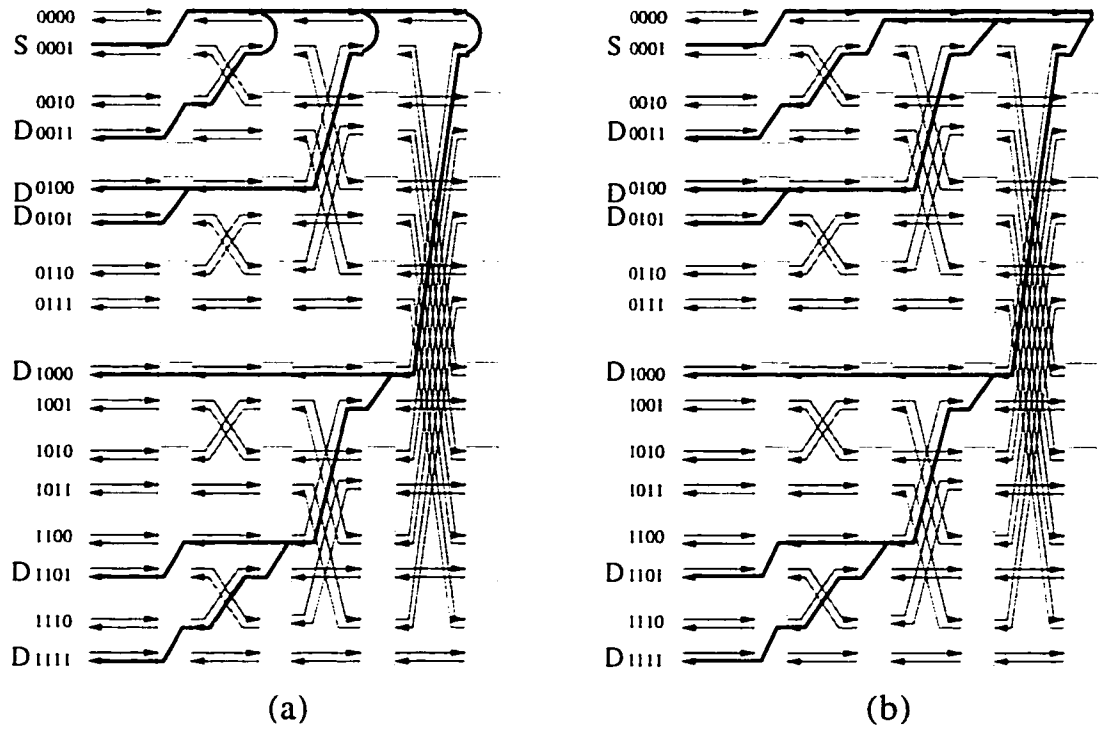


Figure 5.4 Multicasting in BI-MINs (a) multiple turnaround multicast (b) single turnaround multicast.

switch needs to determine the position of its routing information in the header. This position can be easily derived from the switch address. The fixed format of the routing tag reduces the overheads at the intermediate switches. A stripping mechanism can also be employed such that the routing tag for the current stage is discarded before forwarding the message to the next stage. In the worst case, the number of routing tags required at the s^{th} stage is equal to b^s, for a $b \times b$ switch. The broadcast operation involves b^s switches at the b^{th} stage.

We have adopted the *bit string encoding scheme* in which the routing information are represented by N bits, where N is equal to the number of nodes in the system. The bits corresponding to destinations that belong to the multicast group are set to one otherwise they are set to zero. For example, if the multicast destinations of an 8 nodes system are $\{0, 2, 5, 7\}$, the bit string for the multicast message is represented as $\{10100101\}$. The bit string decoder can be supported in form of a routing table. The routing table will consist of b entries with each entry having N bits. A bit in the i^{th} entry is set to "1" on the basis of the destinations that can be reached using the i^{th} output port. The routing decision can be made by performing an AND operation between the bit string in the header and the routing table. The output port i will be used if the result of the AND operation is one. The routing table can be set during

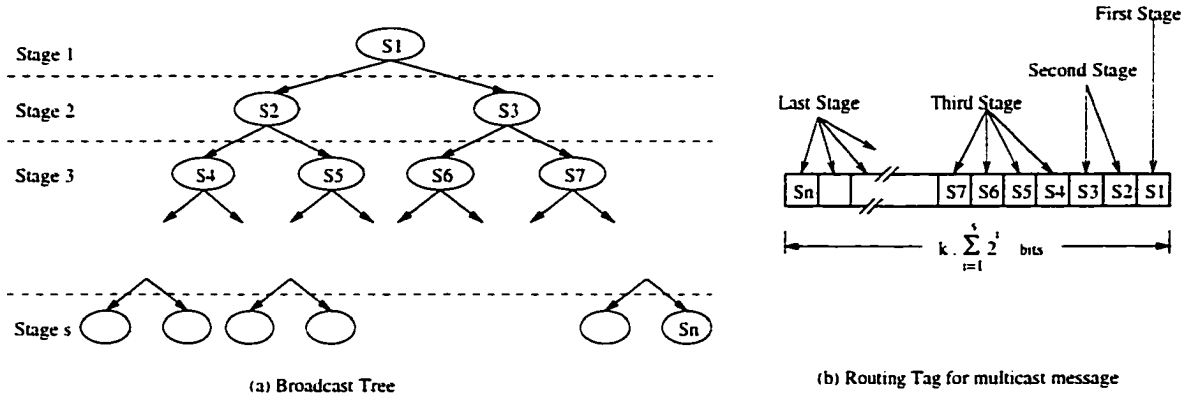


Figure 5.5 Hierarchical multicast routing tags.

the system initialization process. The bit string encoding scheme is suitable for supporting the multicast operation in medium sized systems ($N < 512$). The bit string encoding is also adopted for the tree-based multicasting schemes proposed in [100].

5.2 Deadlocks in Tree-Based Multicasting

In wormhole switching networks, the tree operations create additional channel/buffer dependencies that can form a deadlock cycle. Although tree-based multicasting is a generic approach to distribute information in the network, deadlock is a major problem associated with it. The unicast routing scheme in MINs forwards the unicast message to follow the shortest path from the source to the destination. The requests for the network resources are in one direction. Therefore, no deadlock cycle can be formed. However, the unicast message can be a part of the deadlock configuration. The cause of deadlock in tree-based multicasting is the multiple tree operations. In this section, a detailed treatment of deadlock issues in tree-based multicasting is presented.

5.2.1 Deadlock Configurations

The simplest form of deadlock configuration in tree-based multicasting is the single switch deadlock where the deadlock cycle involves messages at the same switch. Figure 5.6 (a) shows an example of the single switch deadlock configuration. Messages A and B arrive at the input ports of the switch 1. Both messages perform multicast tree operations at switch 1 at stage i. The branches request the same two buffers at switches 2 and 3 at stage $i + 1$. If each branch gains an access to one buffer and requests another, a deadlock cycle is formed. If the switch 1

is in the last stage, similar deadlock cycle involving consumption channels is formed. Figure 5.6 (b) shows this deadlock configuration.

Another form of deadlock is created from the tree operations of the multicast messages at different switches. We term this type of configuration as multi-switch deadlock. An example of this type of deadlock is shown in Figure 5.6 (c). The tree operation of message A is carried out at switch 1. At a later stage, message A occupies the upper port of switch 2 and requests the lower port of the switch 4. At the same time, message B is replicated and forwarded to both the ports at the switch 3. Message B occupies the lower port of switch 4 and requests the upper port of the switch 2. Since both messages cannot proceed further due to the request-and-hold cycle of the network resources, a deadlock occurs. The single switch and multi-switch types of deadlock configurations are the basis of all deadlocks in MINs. Note that both of the deadlock configurations need to be comprised of at least two tree-operations from different multicast messages.

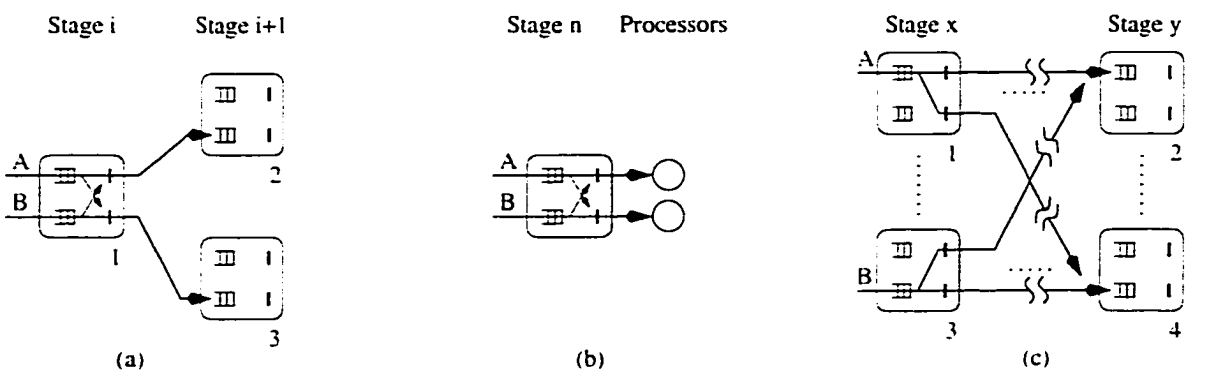


Figure 5.6 Deadlock configurations (a) single switch deadlock (b) single switch deadlock at the last stage (c) multi-switch deadlock.

To illustrate the general concept behind the formation of deadlock cycles, the abstract deadlock configurations are shown in Figure 5.7. In Figure 5.7 (a), two messages perform a switch broadcast operation at the stage zero. These two messages request the same buffers at stage three. Deadlock occurs when messages acquire common buffers in one branch and request the common buffers in the other branch. Figure 5.7 (b) show a possible deadlock that involves more than two messages. This condition may be prevalent in the asynchronous multicast operations.

Virtual channels [54] or dilated physical channels can be used to lower the probability of the occurrence of deadlock configurations. In Figure 5.8 (a), the virtual channels provide alternative paths to the destinations in which the particular deadlock situation is resolved. However, if there are other multicast messages at the input buffers of switch 3, a deadlock is still

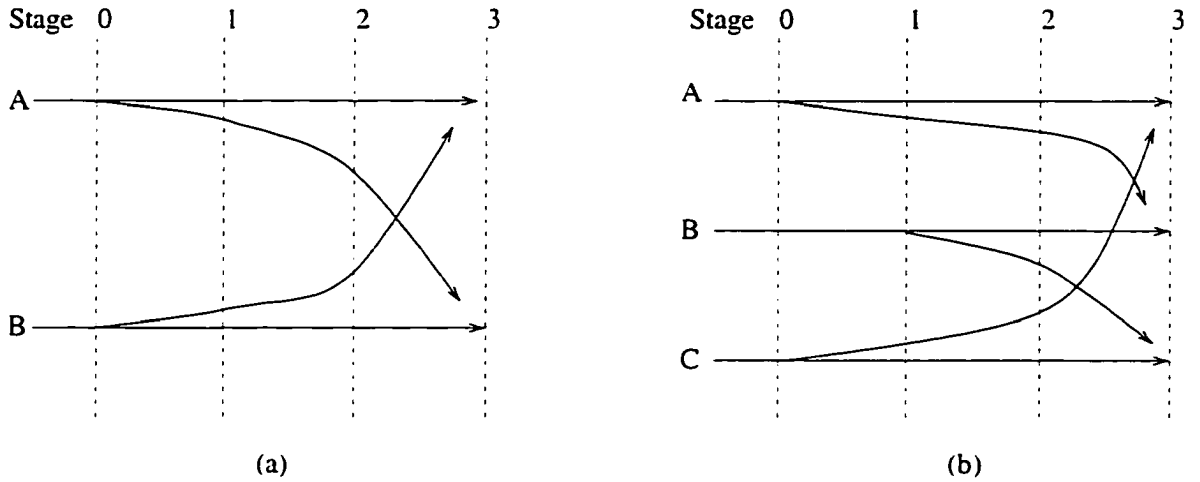


Figure 5.7 Abstraction of multicast deadlocks in a MIN.

possible. The number of virtual channels required is b^j at the j^{th} stage to provide a deadlock-free algorithm for UNI-MINs. This is essentially equivalent to an N-dilated MIN. Similarly, the consumption channel deadlock problem can be solved using multiple consumption channels, as shown in Figure 5.8 (b). To implement multiple consumption channels, the switches at the last stage must have the capability to process more than one incoming message concurrently. In direct networks, it is not uncommon to have more than one consumption channel since the router switch is closely coupled with the processor. To provide multiple consumption channels in MIN systems, special switches are needed at the last stage. If the number of consumption channels is equal to the number of input ports, the messages that arrive the last stage will eventually be consumed. The consumption channels can be either virtual consumption channels or physical consumption channels [112].

5.2.2 Switch Grouping

Deadlocks in MINs occur due to multiple tree operations of the multicast messages. However, not all combination of tree operations can lead to a deadlock cycle due to the partitionable structure of the MINs. For example, multiple tree operations at different switches (with one tree operation per switch) at the last stage are mutually exclusive. The branches created at these switches will not block each other since they all are eventually consumed by the destination nodes. Similarly, at the network level, we can partition the buffers and channels of a baseline UNI-MIN using 2×2 switches into two separate sets, as shown in Figure 5.9 (a). These two separate sets have no buffers or channels in common. Similarly, the 16 nodes with 4×4 switches can be partitioned in to four separate sets of buffers and channels as shown

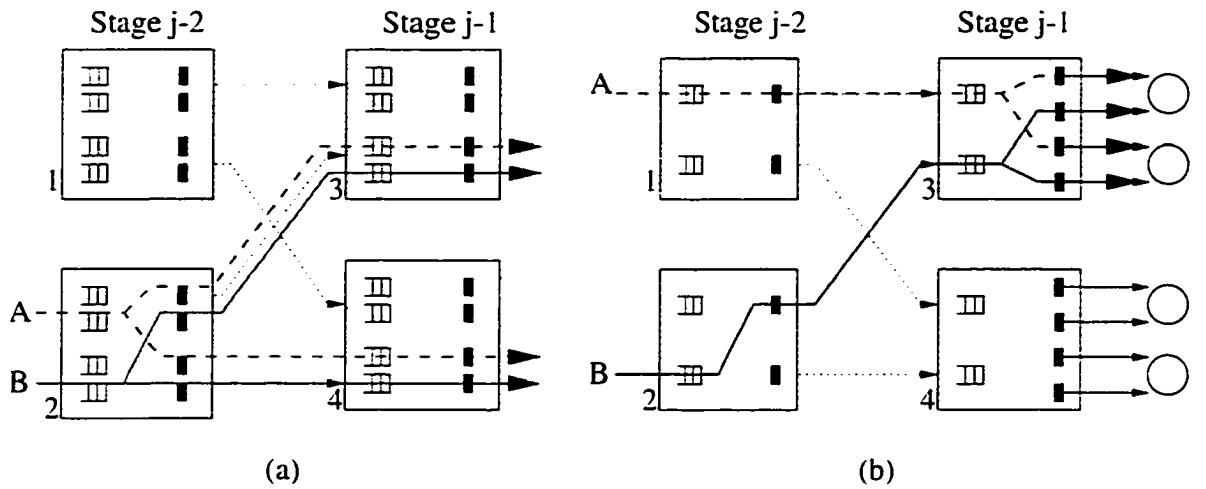


Figure 5.8 Deadlock avoidance (a) using virtual channels (b) using multiple consumption channels.

in Figure 5.9 (b). A message using buffers within a partition of the network will not request the buffers of any other partition in the network. Partitioning of the MIN on the basis of its topology is examined in detail in the literature [6, 113]. After partitioning the MIN, we need to group the switches of the partitions at each stage. The grouping is done in such a way that the tree operations within a group do not interact with the tree operations of any other group. Because of the special structure of baseline UNI-MINs, the grouping can be determined using following equations. The number of switches in the group N_{gj} at stage ($j \in [0, 1, \ldots, n-2]$) for the single consumption channel model is given by, $N_{gj} = b^{(n-1)-j}$ and for the b consumption channels model, $N_{gj} = b^{(n-2)-j}$. For the baseline networks, the switch (i, j) belongs to group (k, j), i.e., the switch belong to group k at stage j. The group label (k, j) can be calculated using the equation, $k = \lfloor \frac{i}{N_{gj}} \rfloor$.

For other switch-based networks, the algorithm shown in Figure 5.10 can be used to partition switches into groups. The switches that have possible permutations to access common buffers/switches at a later stage are grouped together. The algorithm can be applied directly to unidirectional MINs. For BI-MINs, a simple network transformation is needed. The switches (i, j) in a BI-MIN are redefined as (i, j, k) where k is the new stage label. The value k is equal to j and $2n - (j + 1)$ for forward communication links and backward communication links, respectively. The BI-MIN network is partitioned into two sets based on the direction of the communication links–forward and backward networks [100]. With the reconfiguration, the BI-MIN can be considered as two concatenated unidirectional MINs as shown in Figure 5.11. The unidirectional MIN that sends the messages away from the source node is denoted as the

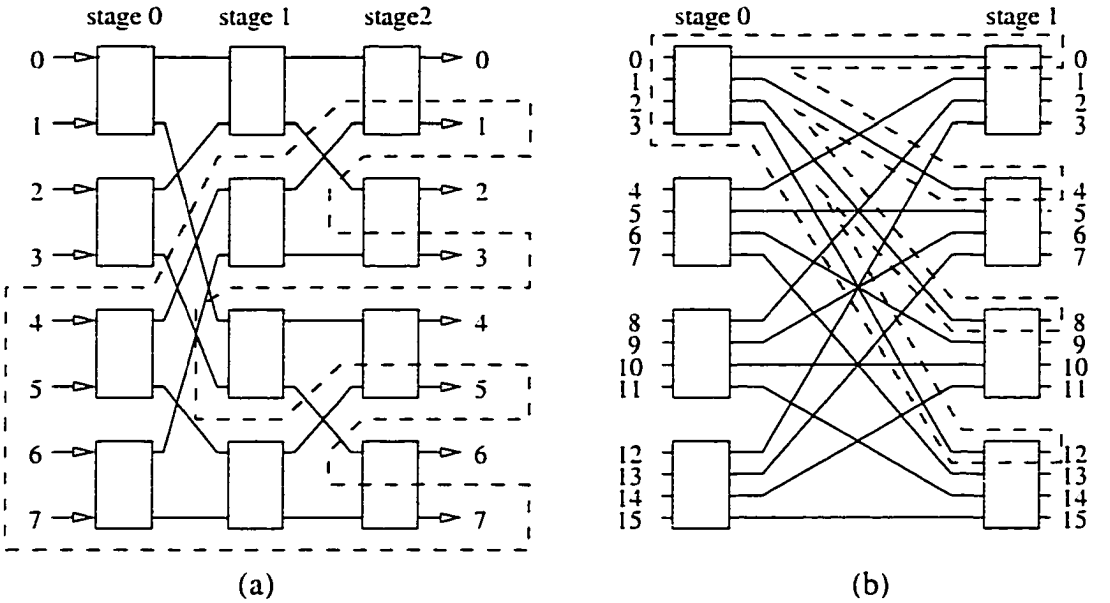


Figure 5.9 Partitioning of the baseline networks (a) 8 nodes using 2 × 2 switch (b) 16 nodes using 4 × 4 switch.

forward network and the unidirectional MIN that sends the messages back to the destination nodes is denoted as the backward network. The turnaround routing operation forms the connections between the forward and backward networks. Note that, in some cases, the messages can turn around at the early stages.

The grouping algorithm, shown in Figure 5.10, assigns a group tag to each and every switch in the network. The group tag is a set of row numbers. First, the group tag of switches that are connected to the input ports of the processing nodes are assigned to their own row numbers. These group tags are propagated back along the communication links to the previous stage. The group tags of the previous stages are marked as the union of the group tags propagated back from the output ports. This process is recursively carried out until the 0^{th} stage of the forward network is reached. The switches that have the same group tag are considered to be in the same group. An example of the switch grouping is shown in Figure 5.11. Each switch at stage 5 of the backward network constitutes a group. The switches $(0, 1, 3)$ and $(1, 1, 3)$ have the same group tag and are considered to be in the same group. All switches in the forward network and the stage 3 of backward networks belong to the same group. This grouping algorithm can be applied to all of the MIN topologies.

If the tree operations are performed by multiple multicast messages in the same switch group, there is a possibility that a deadlock configuration could be formed. This is because, the switches in the same group can request the same set of buffers at a later stage. The

Switch Grouping Algorithm

```
/* b ports switch, n stages, and r rows MINs */
/* Replace n by 2n for BI-MIN */

group_tag = {∅}

begin
    for i = 0 to r − 1
        group_tag(i, n − 1) = {i}; /* Replace n − 1 by n − 2 for b consumption
        channels models */
    for j = n − 1 to 0 /* Start j iteration from n − 2 for b consumption channels
    models */
        for i = 0 to r − 1
            for k = 0 to b − 1
                l = row number of the next stage switch from
                    the permutation of port k;
                group_tag(i, j) = group_tag(i, j) ∪ group_tag(l, j + 1);
    Group the switches with the same group tag;
end
```

Figure 5.10 Switch grouping algorithm.

tree operations from multiple mulitcast messages that initiate from different groups will never create any deadlock configuration.

A node can receive only one message or multiple messages at the same time. The number of the messages that the nodes can receive is usually dependent on the number of consumption channels. If the last stage switches and the processing nodes have the capability to transfer b messages concurrently, all the incoming messages at the last stage will be consumed by the processing nodes without contention. Multiple consumption channels not only increase the throughput but also eliminate the deadlock due to consumption channels dependencies (in Figure 5.6 (b)).

The switch grouping technique for the systems with b consumption channels is sligthly different from the algorithm in Figure 5.10. The grouping process starts from the stage $(n-2)$. The switches at the last stage do not need to be grouped since they cannot be invloved in a deadlock configuration. The dashed line in Figure 5.12 shows an example of switch grouping at each stage of the baseline network. The grouping example for the single consumption channel model is shown in Figure 5.12 (a) for 64 nodes using 4×4 switches. The dashed lines in Figure 5.12 represent the switches in the same group at each stage of the baseline UNI-MIN. For the single consumption channel model, the switches at the first stage (stage zero) belong to the same group. Notice that the number of switches in a group decreases as

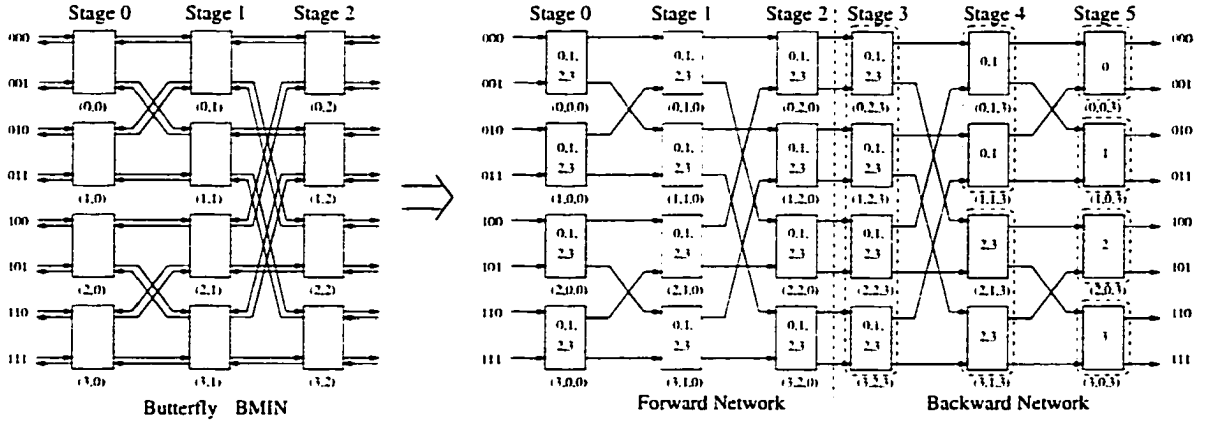


Figure 5.11 Partitioning of BI-MIN into forward and backward networks.

the number of stages increases. Figure 5.12 (b) shows the grouping for b consumption channels model. It is important to note that the number of switches in the group at each stage for b consumption channels model is significantly less than the single consumption therefore it allow more concurrent tree operations. All MIN systems proposed so far support the consumption of a single message in a cycle. However, we have analyzed and evaluated the cases for both single and multiple consumption channels.

5.2.3 Deadlock Prevention Techniques

In the tree-based multicasting approach, the tree operations created by multiple multicast messages are the primary causes of the deadlock configurations. Deadlocks can be prevented either through the use of additional network resources or by controlling the routing behavior. Additional buffer space or additional physical/virtual channels can be used to prevent deadlocks. Several tree-based multicast algorithms [24, 99, 100, 101] use additional buffer space for deadlock prevention. In [100, 101], when the multicast tree operation is performed, the branch message is forwarded only when there is enough buffer space to store the entire message at the next switch. The buffer dependencies from the tree operations are therefore eliminated at the buffer. The maximum packet size in these systems is thus limited. Packetization at the source and reassembly at the destinations are required if the actual message is larger than the limited packet size.

Control of the routing behavior is another approach for deadlock prevention. The single switch deadlock can be prevented locally at the switch using some priority schemes [98]. Priority-based schemes for tree-based multicasting can be found in [98]. An example of these schemes is the upper-port first approach where a message from the upper input port has higher

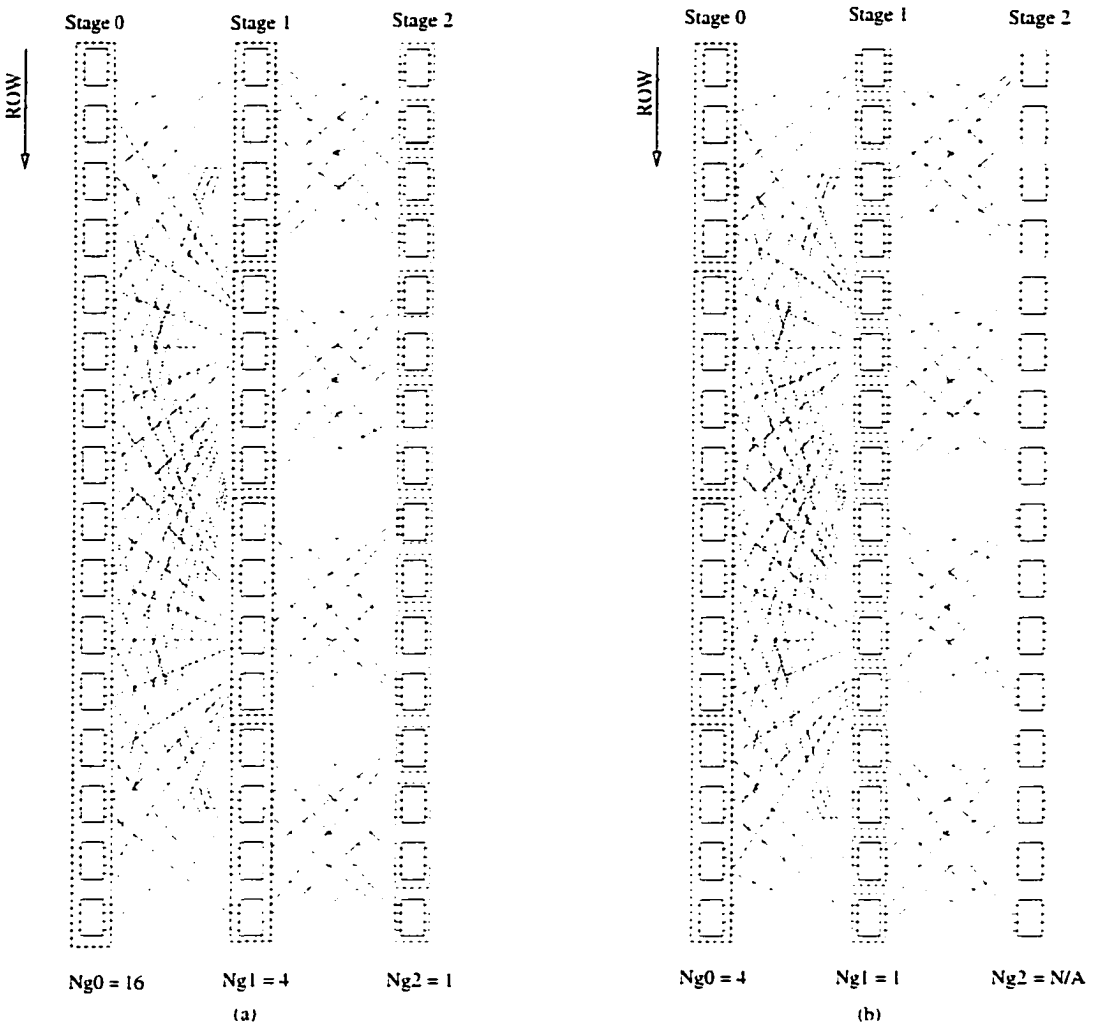


Figure 5.12 Switch grouping of the in a 4 × 4-switches 64 nodes baseline network (a) one consumption channel model (b) *b* consumption channel model.

priority to gain access to the network resources.

Deadlock freedom and starvation freedom are two major requirements of the priority-based schemes. The multi-switch deadlock requires a complicated deadlock prevention mechanism. A synchronous worm could prevent the multi-switch deadlock configurations by restricting the multiple branches to be forwarded synchronously at the same stage. If one branch is blocked, all the other branches cannot be forwarded to the next stage. The synchronous mechanism prevent the multi-switches deadlocks created between the consecutive stages. Thus, the single switch deadlock can be prevented using the priority machanism and the multi-switch deadlock can be eleminated using the synchronous worm. However, in asynchronous routing, it is difficult to

control the routing behavior to prevent deadlock configurations. It is more prone to deadlocks since the branches are allowed to be forwarded independent of the multicast destinations. We have analyzed this problem and have proposed an effective solution.

5.3 The Proposed Algorithms

In this section, we propose a deadlock prevention technique for asynchronous tree-based multicasting (ATBM) in MINs. We have also discuss the use of this technique for UNI-MINs as well as BI-MINs.

5.3.1 ATBM Framework

In the previous section, we examined the main causes of deadlocks in MINs. We observed that one way of preventing deadlocks in asynchronous MINs is by avoiding the operations that have a potential to create deadlock configurations. This simple concept motivates the key idea behind the proposed ATBM scheme.

In the proposed ATBM scheme, the multicast routing information is encoded in the message header. The tree-operations are performed at the appropriate switches to distribute the multicast messages to the destinations. Multiple branches of a multicast message are forwarded asynchronously. However, the ATBM scheme imposes certain restrictions. The switches of the MIN are grouped first using the grouping algorithm presented in Section 5.2.2. As noted earlier, concurrent tree operations within a group could lead to a deadlock configuration. The proposed ATBM scheme prevents such occurrences by serializing the tree operations within a switch group. Thus the formation of deadlock configurations are prevented. The imposition of serialization is temporary and involves only the messages that are branching at the same stage within a group. Once the multicast headers of a message reach their destinations, a subsequent tree operation (if blocked in the same group) can be initiated. The time at which a message header reaches the destination can be detected at the switch as described next. The tree operations of different multicast messages in different groups of switches are allowed to proceed without any restrictions.

The ATBM framework required only a small buffer space that can store the header of a multicast message at each input ports. There is no limit on the maximum message size that can be multicasted. Multicasting to any number of destinations can be completed in a single communication step using one communication start-up time.

Theorem 5.1: The ATBM algorithm is deadlock free.

Proof: Using the grouping technique explained in the previous section, buffer dependencies in different groups of switches become mutually exclusive. Therefore, the initiation of tree operations in different group of switches will not lead to any deadlock. A deadlock configuration can be created by the multicast messages only when at least two tree operations from different multicast messages are performed in the same group of switches. Thus, by the serialization technique, deadlock cycles can be prevented and deadlock cannot occur. Q.E.D

Additional hardware is required in the ATBM scheme for serializing the tree operations in the switches of the same group. A control line can be implemented that interconnects the switches of a group. To preserve fairness, a hardwired token passing mechanism can be employed. The token passing mechanism is a practical approach to solve the mutual exclusion problem between switches. The deadlock recovery algorithm *DISHA* [58] has also adopted a similar token passing technique in the deadlock recovery process. The token can be passed from one switch to another through the control line in a round-robin fashion within a group. When a switch is ready to do a multicast operation, it waits until it gets the token. The switch holds the token while multicasting a message and releases it after the headers of message arrives at the destinations. Thus, the switch can relinquish the token when the number of flits that have been forwarded to the next stage is more than the sum of the buffer sizes along the path of the switches to the destinations. This value is equal to $B \times (n - j)$ where B is the buffer size at a switch, j is the current stage and n is the total number of stages. The deadlock-free property is still held because if all destinations have been reached, the multicast message will eventually be consumed by the destinations. As the number of stages in the MINs is usually less, the waiting time for the token will be within limits and multiple multicast messages can proceed if there are no blocking. This hardware modification is simpler and faster than the feedback mechanism required for synchronous multicasting [98]. The synchronous multicasting scheme needs to synchronize all multihead worms, which requires the permutation of the whole multicast tree.

In [58], the asynchronous token implementation was proposed where the token passing time is dramatically reduced. The implementation of the control line and the token passing mechanism is just an example method of implementation of the serializations. Other efficient techniques can be explored and implemented to facilitate the serialization effectively.

5.3.2 ATBM for UNI-MINs

Multicasting operations in a UNI-MIN can be supported ideally by an algorithm that can send a multicast message to all the destinations using a single start-up phase. This can be achieved by tree operations at the appropriate stages. The bit string encoding scheme discussed

UNI-MIN Routing Algorithm(message_header)

```
1. If (unicast_message)
       Forward a flit to the output port specified in the routing_tag;
2. If (multicast_message)
       Perform routing function according to the routing table;
       If (routing function returns multiple output ports)
           Wait to obtain the token;
           Hold the token;
           Replicate and forward the header flit to the output ports supplied
           by the routing function;
           Release the token after the header flits arrive at the destination(s);
       else
           Forward the header flit to the output port returned
           by the routing function;
```

Figure 5.13 The ATBM routing algorithm for UNI-MIN.

in Section 5.1 is well suited to for the ATBM scheme. The routing information in the header needs only N bits for any multicast operations in an N processors system. A minimum buffer size of N bits per input port is required to store the multicast message header in the ATBM scheme. If the flit size is less than N bits, the central buffers can be used to store the header processing. If the routing table indicates that a tree operation need to be performed at the current switch, the multicast message has to wait to gain access to the token before replicating the flits. After obtaining the token, the input port holds the token and proceeds with the tree operation by forwarding independent headers to different output ports. If there is another multicast message requesting for a tree operation from the same switch group, the requesting message is blocked until the other message releases the token. While one branch of a multicast message is blocked either waiting for the token or by other messages, the other branches can proceed asynchronously. Bubbles are introduced in the branches that proceed asynchronously as discussed earlier by Chiang and Ni [98]. The formal description of the algorithm is given in Figure 5.13.

5.3.3 ATBM for BI-MINs

In BI-MINs, it is possible that there are more than one alternative paths from the source to the destination. The unicast message can be adaptively routed to the turnaround stage and can use the destination tag routing in the backward network to reach its destination. For

multicasting, there are two choices of turnaround operations, single turnaround or multiple turnaround. Even though multiple turnaround approach uses less network resources in terms of the communication links, it is not convenient for the ATBM approach for two reasons. First, multiple tree-operations of the same message could be necessary within a switch group. It is difficult to identify such a scenario. Since all switches in the forward network will constitute of a single group, the serialization overhead is higher. Second, more information regarding multiple turnaround stages need to be contained in the header which will lead to a higher overhead. These problems do not exist in the single turnaround approach where the switches of a group are at the same stage and there is only one turnaround point. For the single turnaround, the least common ancestor of the source address and all destination addresses can be computed in a similar manner. The multicast turnaround stage T is defined as the first position where the s_i and d_i of all multicast destinations are different from the left hand side. The multicast message is adaptively routed to the stage T and performs tree operations to forward multiple messages to all of the multicast destinations.

A routing table can be used to support the multicast tree operations in the backward routing phase. Each multicast destination is represented by a single bit string in the header. The output port(s) to which the message needs to be forwarded at the current switch is obtained by comparing the encoded destinations and the routing table. This approach enables multicasting to arbitrary number of destinations in one communication phase. The routing table can be initialized during the system start-up and is static during the system operation.

The formal ATBM algorithm for BI-MIN is shown in Figure 5.14. The conventional turnaround routing is performed for the unicast message. The destination tag routing is used if the message turns around at current stage or if it is already in the backward network. Otherwise, the message is forwarded to the first available port in the forward network. The multicast message performs similar turnaround routing operations. A multicast message is first checked to determine if it needs to be routed in the backward network. The backward output port(s) is supplied by the routing function (in bit string encoding scheme). If tree operations are performed (multiple output ports), the switch must wait for the token within the switch group. After the token is received, the tree operation is initiated. The token is released when all the multicast destinations have received the message header. The flit number can be used to check this condition. The multicast message is adaptively routed in the forward network before the turnaround stage. Only one turn around point is allowed while using this algorithm.

For the two-sided BI-MIN, both unicast and multicast operations can be completed within two communication phases. Each communication phase is for the processing nodes in one side. The UNI-MIN routing algorithm can be used for the nodes in opposite side and the BI-MIN

BI-MIN Routing Algorithm(message_header)

```
1. If (unicast_message)
       if (the message is in the backward network) or (current_stage
       = turnaround_stage)
           Forward a flit to the backward output port specified in the routing_tag;
       else
           Forward to the available output port in the forward network:
2. If (multicast_message)
       if (the message is in the backward network) or (current_stage
       = turnaround_stage)
           Perform routing function according to the routing table:
           If (routing function returns multiple output ports)
               Wait to obtain the token:
               Hold the token;
               Replicate and forward the header flit to the specified backward
               output ports:
               Release the token after the header flits arrive at the destinations:
           else
               Forward the header flit to the specified backward output:
       else
           Forward to the available output port in the forward network:
```

Figure 5.14 The ATBM routing algorithm for BI-MINs.

routing algorithm can be used for the nodes in the same side. The same switch grouping algorithm can be applied twice starting from each side.

5.4 Performance Evaluation

The proposed ATBM algorithm can complete any multicast operation in a single step regardless of the distribution or the number of the destination nodes. However, there is still an overhead incurred in the waiting time due to the serialization of the tree operations (in case of multiple multicast within a group) that adds on to the performance penalty for avoiding deadlocks. The overheads of ATBM were investigated using wormhole network simulator. The complehensive simulation results are presented in this section.

5.4.1 Preliminary Performance Results

We first evaluate the performance of ATBM scheme for UNI-MIN. The objective of the preliminary evaluation is to validate the performance improvement of the ATBM approach, therefore simulation environment is set preferable to software-based scheme. We design the

experiments to evaluate the performance of the 2 × 2 and 4 × 4 switch-based unidirectional baseline MINs networks. MINs of 16, 32, 64, and 256 nodes are considered in our study.

The time to transfer a flit between two switches is set to 20 *ns* which is close to the parameters in the contemporary technology. Each experiment was simulated for 150,000 messages. Statistical results of the first 40,000 messages are not included in the result to reduce the transient effects of the network. The simulation results stay within a spectrum of 5%. The message generation time at each node was assumed to be exponentially distributed.

We generate a mixture of unicast and multicast traffic. We assumed uniform traffic distribution where there is an equal probability to send a message from one node to any other node. The inter-arrival time of unicast and multicast messages is assumed to be exponentially distributed. The average number of multicast destination scales up with the number of nodes in the system and we have assumed it to be $\frac{N}{2}$ with a standard deviation of $\frac{N}{4}$, where N is the number of nodes in the system. To compare the performance in different network sizes, we have used the normalized network load to determine the inter-arrival time at each node. The network load, denoted by T_r, is defined as the ratio of the average network load generated by the processing nodes in the system to the total buffer resources available in the network. The inter-arrival times for unicast (T_{arru}) and multicast messages (T_{arrm}) can be obtained using the equations,

$$T_{arru} = \frac{B_m \times N}{(B_t \times T_r \times (1.0 - M_p))},$$

$$T_{arrm} = \frac{B_m \times M_c \times N}{(B_t \times T_r \times M_p)},$$

where B_m is the amount of buffer-time required by a message. Buffer-time is defined as the total duration for which a message occupies buffer space in the network. M_p denotes the ratio of multicast messages generated to the unicast messages generated. M_c denotes the average number of destinations per multicast operation. B_t is the total buffer space in the network which is equal to the product of buffers at all input ports and the number of switches.

The ATBM algorithm that allows the switches to release the token when all multicast destinations receive the multicast headers is simulated. Multiple tree operation requests within the same group are processed in a round-robin fashion. A delay of 40 nanosecond is added when the tree operation is performed (not including the waiting time). The multicast header is assumed to fit in one flit using bit string encoding scheme. Each input port is associated with one flit buffer. The size of unicast and multicast messages are fixed to 128 flits and 64 flits, respectively.

We compare the ATBM scheme with the CMIN algorithm [87] to show the performance improvement from supporting the multicast at the hardware level. The CMIN algorithm is

the binomial unicast-based multicast algorithm which requires $\lceil \log_2(d+1) \rceil$ to complete the multicast operation to d destinations. The multicast tree structure is constructed such that the blocking is minimized. The purpose of our study is to investigate the effect ATBM scheme on network latency. Therefore, the communication start-up time is not included in the multicast latency. If we include the communication start-up time, the performance of the software multicast will be degraded severely .

There are two other previously proposed hardware multicast schemes for MIN, synchronous multicast [98] and multiport encoding schemes [101]. The results of these two schemes are not shown because it is very difficult to get a fair comparison. The synchronous multicast algorithm requires a feedback mechanism. Intuitively, the ATBM algorithm should perform as good as the synchronous multicast algorithm with lower hardware requirements. The multiport encoding algorithm needs buffers of the size of one message at each input channels. The multicast operation using this scheme involves several communication start-up phases. Therefore it will incur higher start-up latency and the performance will not be comparable to the ATBM scheme.

Figure 5.15 (a) shows the multicast communication latency for baseline networks using 2×2 switches. The single consumption channel model is assumed. The mixture of the traffic is set to 20% multicast traffic with 80% unicast traffic. The ATBM algorithm has less multicast latency by a factor of four compared to the software multicast scheme (even after neglecting the start-up latency). Figure 5.15 (b) shows the comparison of unicast latency. The result of unicast latency is important because it reflects the impact of the multicast algorithm in terms of additional network load. Both unicast and multicast latencies of the ATBM algorithm are lower than the respective latencies of the CMIN algorithm. The latency curve of the CMIN algorithm leads to an early saturation. At heavy load, the multicast operations tend to be blocked at the early stage. Therefore, they do not congest the network. This behavior is also reflected in the simulation results.

The simulation results for baseline MINs using 4×4 switches are shown in Figure 5.16. All simulation parameters are the same as the previous results. Similar trends can be observed for both multicast and unicast latency. The ATBM scheme performs very well for 256 nodes systems. As previlously discussed, the b consumption channel model can enhance the performance of the ATBM scheme. Figure 5.17 show the multicast latency under the b consumption channel model with a little performance improvement for the ATBM scheme compared to the results in Figure 5.15 (a) and Figure 5.15 (b).

The proportion of multicast and unicast traffic is set to 50% for the simulation results in Figure 5.18. The ATBM scheme gives even better performance for high ratio of multicast traffic. This is because the ATBM scheme uses the tree multicast structure with is quite

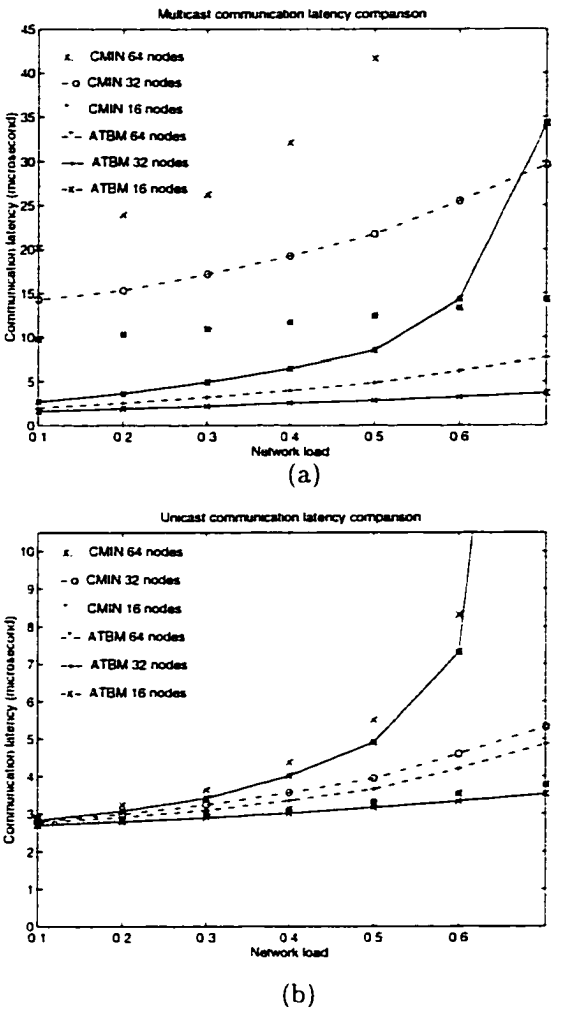


Figure 5.15 Performance comparison of ATBM and CMIN schemes ($M_p = 0.2$) (a) multicast latency (b) unicast latency.

suitable for the MIN topology.

The performance penalty of the ATBM algorithm is the blocking time of the tree operation. To investigate the impact of waiting time, the first stage waiting time versus the traffic load is shown in Figure 5.19 (a). The multicast traffic ratio is set to 50 percent. The communication latency in a contention free environment can be approximated to the message length (32 cycles). The average waiting time is about one message latency at the operating point of networks (0.5). Figure 5.19 (b) shows the average number of messages waiting for the tree operation at the first stage.

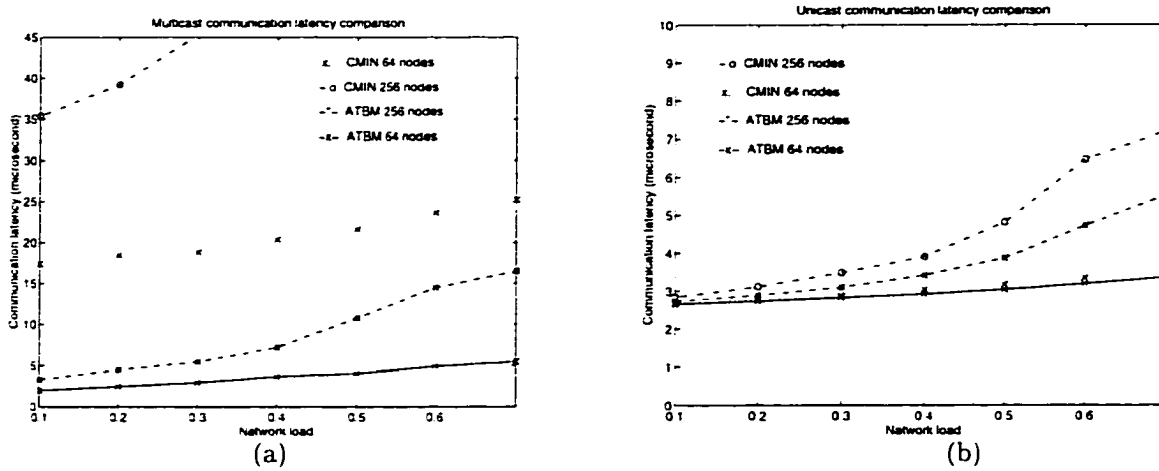


Figure 5.16 Performance comparison of ATBM and CMIN Schemes ($M_p = 0.2$) (a) multicast latency (b) unicast latency.

5.4.2 Performance Results under Realistic Enivironment

In the second part of the performance results, the realistic network parameters are used in the simulation. We have simulated baseline UNI-MINs and butterfly BI-MINs. A variety of MIN sizes using 2x2, 4x4, and 8x8 switches were considered. The network parameters were set similar to the current technology trend. We have assumed 3.2Gbits per second communication links. The buffer size at the switches is assumed to be enough to store only the header of the incoming messages. The transmission time of one flit between a pair of switches is assumed to be 20 nanoseconds. The routing decisions are assumed to take 60 nanoseconds. The message startup time is set to 0.5 microsecond. Reception latency is ignored. Unless explicitly specified, the message size is assumed to be equal to 64 flits for both unicast and multicast messages. Each flit is assumed to be 16 bits wide. Both multicast and unicast communication latencies are considered as the performance metrics. The ATBM algorithms for both UNI-MINs and BI-MINs are compared with the previously proposed software based algorithms [87, 86]. The software based algorithm proposed for BI-MINs is unicast-based UMIN algorithm and the respective algorithm for UNI-MIN is unicast-based CMIN algorithm.

The additional overhead of the ATBM scheme is the serialization overhead of the tree operations. To simulate this behavior, additional delay is added to reflect the token passing time. The token passing mechanism is assumed to be synchronized with the switch clock. The average token passing time is equal to $20 \times N_{gj}/2$ nanoseconds, where N_{gj} is the number of switches in the group. The token passing time does not include the serialization delay in which the token is held by the other multicast message. After the token is released, the token passing

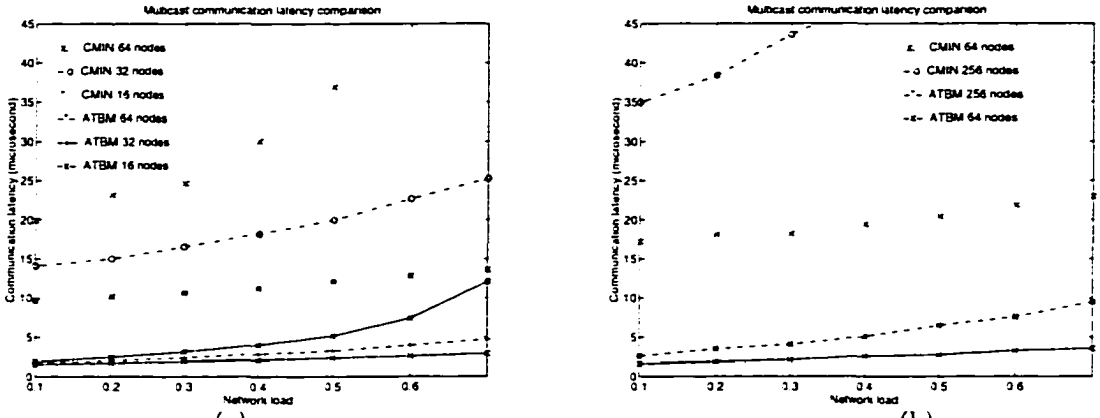


Figure 5.17 Performance comparison under b consumption channels model ($M_p = 0.2$) (a) MINs using 2×2 switches (b) MINs using 4×4 switches.

delay is penalized before the next message can initiate the tree operation.

5.4.2.1 Performance Comparison under Contention-Free Environment

To examine the effect of the proposed scheme on multicast communication, we first analyze the ATBM algorithm under contention-free conditions. Each experiment was repeated 1000 times. The multicast source and destination nodes were selected randomly. The average multicast latency measures are plotted with respect to the number of destinations.

Figure 5.20 (a) shows the simulation results for a 64-node UNI-MIN system using 2×2, 4×4, and 8×8 switches. The performance results of the ATBM algorithms are very promising as the multicast latency using ATBM algorithm is significantly lower than the unicast-based CMIN scheme for all the cases. The multicast latency of the ATBM scheme remains almost constant when the number of destinations increases. This is because the ATBM approach requires only one communication step regardless of the number of destinations. Similar trend is observed for the 256-node system, as shown in Figure 5.20 (b). Figures 5.21 (a) and (b) show the performance results for BI-MIN systems. The ATBM approach perform quite well in BI-MIN systems. The ATBM latency is much lower than that of UMIN scheme and remains unchanged as the number of destination increases. The multicast latency of UMIN scheme increases stepwise as the number of destinations increases. The additional latency accumulated from the communication phase is required when the number of destinations exceeds some threshold (in the order of $\lceil \log_2(d+1) \rceil$.)

The multicast latency components of the ATBM scheme consists of startup latency, routing

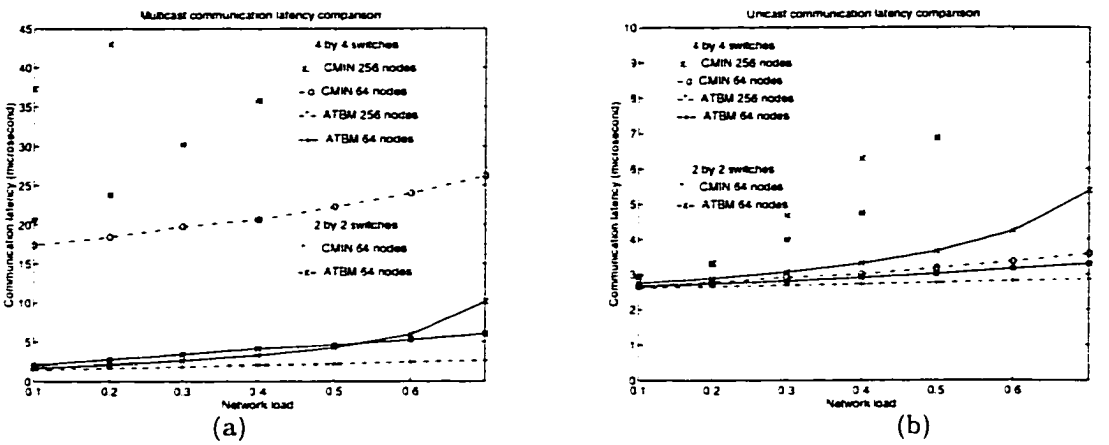


Figure 5.18 Performance comparison of ATBM and CMIN schemes ($M_p = 0.5$) (a) multicast latency (b) unicast latency.

delay, token passing time, and transmission delay. The results from the simulation are very close to the theoretical results. For example, the 64-node system BI-MIN using 8×8 switches, the theoretical multicast latency is equal to 2.12 microseconds which is the summation of 0.5 microsecond (startup latency), 3×60 nanoseconds (routing delay), 4×20 nanoseconds (token passing delay), and $(4 + 64) \times 20$ nanoseconds (transmission delay).

The multicast latency of the ATBM schemes is lower than the multicast latency of the unicast-based schemes by more than a factor of 4 when the number of destinations increases to more than one-half of the system size. The simulation results further validate the claim for the hardware supported multicasting in scalable parallel systems.

5.4.2.2 Performance of Unicast and Multicast Traffic with Contentions

The ATBM scheme serializes some of the tree operations for deadlock prevention purpose. The serialization incurs two additional overheads–serialization waiting time and token passing time. When there are several multicast operations, the initiation of tree operations might have to wait for other multicast messages to release the token. We simulate networks with contentions to study the impact of these overheads. The impact of the multicast operations on the latency of the unicast communication is also a crucial issue that needs to be considered in designing multicast algorithms. We have simulated a mixture of unicast and multicast traffic to study such an environment. Each experiment was simulated for 140,000 messages. Measurements of the first 40,000 messages were not included in the statistical results to reduce the transient effects of the network. The simulation results vary within a spectrum of 5%. The

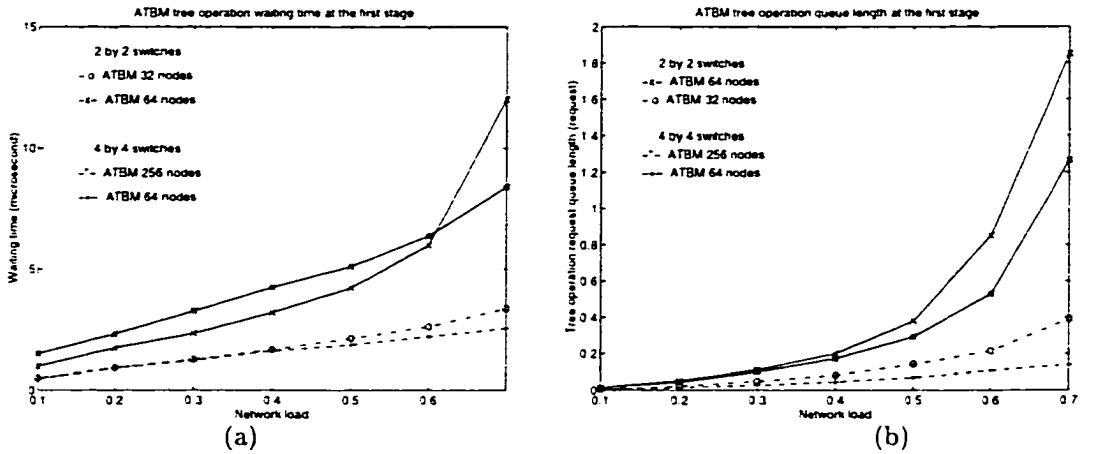


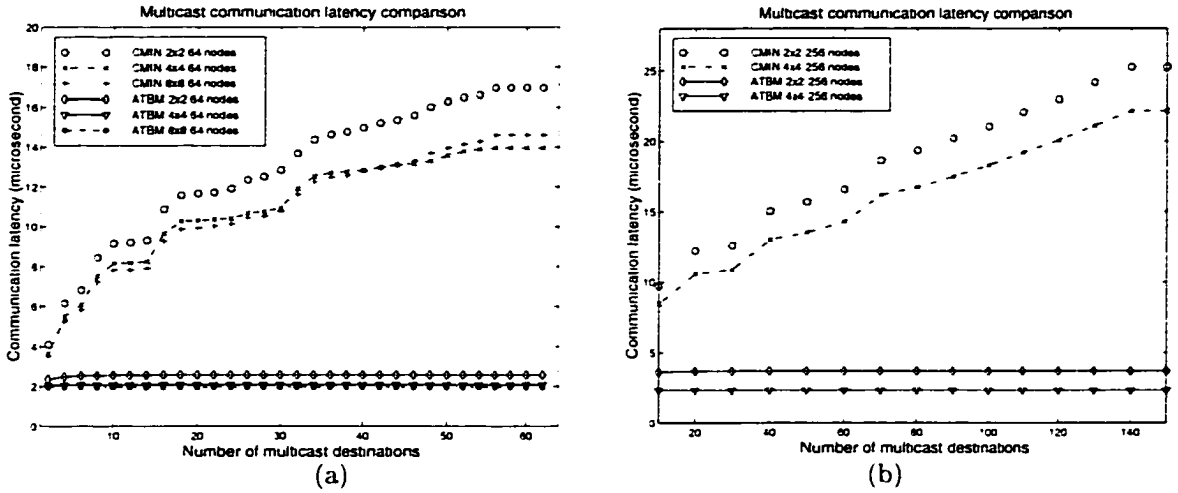
Figure 5.19 Impact of ATBM on the first stage of MIN (a) waiting time (b) queue length.

normalized network load, defined in previous section, is used to determine the arrival time of unicast and multicast messages.

5.4.2.3 Simulation Results with 50% Multicast Traffic

In the first set of results, the ratio of unicast to multicast traffic is set at 50% ($M_p = 0.5$). A single consumption channel model was assumed. Figure 5.22 (a) shows the multicast latency versus the network load for the UNI-MIN system using 2 × 2 switches. The ATBM multicast latency is lower than the software multicast latency by a factor of 3 for small size systems (16-node and 64-node). For larger size system (256-node), the ATBM scheme outperforms the unicast-based CMIN scheme by a factor of 4. As the network load increases, the software-based scheme generates more traffic in the network. The message contentions lead to early saturation. In light to medium traffic conditions, the multicast latency of the ATBM scheme performs very well compared to the CMIN scheme. The effect of contention due to serialization becomes prominent with the heavy traffic. The performance results for BI-MIN are given in Figures 5.23 (a) and (b). Similar performance improvement for BI-MIN is achieved using ATBM approach. The ATBM performance in BI-MIN is comparable to the UMIN performance in the heavy traffic condition.

With a mixture of unicast and multicast traffic, the performance of unicast communication is shown in Figure 5.22 (b). The multicast operations using ATBM scheme consume less network resources, therefore they impose less restrictions on the unicast communication. The unicast performance gain using ATBM scheme ranges from 25% to 40% depending on the



(a) (b)

Figure 5.20 Performance comparison under contention-free condition (a) 64-node UNI-MINs (b) 256-node UNI-MINs.

system size with respect to the CMIN algorithm. Figure 5.23 (b) shows the unicast latency versus network load for BI-MIN systems. The performance improvement in BI-MIN systems is comparable to the UNI-MIN systems before the saturation point. In 64-node and 256-node systems, the unicast latency of the ATBM approach saturate slightly earlier than that of the UMIN scheme. With realistic network parameters, the overall performance of ATBM is better for both multicast and unicast communications. Note that these MIN systems are using 2×2 switches that offer the minimum degree of connections.

As the switch size increases, the system offers more degree of connections. Contemporary MIN systems are constructed using 4×4 or 8×8 switches. With the same number of nodes, MIN systems using larger switches have lower congestion, compared to 2×2-switch systems. Figure 5.24 (a) shows the multicast latency versus the network load for the UNI-MIN systems using 4×4 switches. The same performance improvement is obtained in terms of the multicast latency. The contentions due to the serialization of tree operations is less as the degree of connectivity increases. For the 16-node UNI-MINs, the ATBM approach offers performance improvement of a factor of 3. The multicast latency of the ATBM algorithm is 4 times less than the software-based approach for 64-node and 256-node systems. For the UNI-MINs using 4×4 switches, the ATBM approach extends the operating range at the heavy traffic condition. In light to medium traffic region, the ATBM scheme has less impact on the unicast performance. As shown in Figure 5.24 (b), the unicast latency results using ATBM scheme is lower than that of the CMIN scheme. Because of the high penalty that we have imposed on the token passing time, the unicast performance of a 256-node system using the ATBM

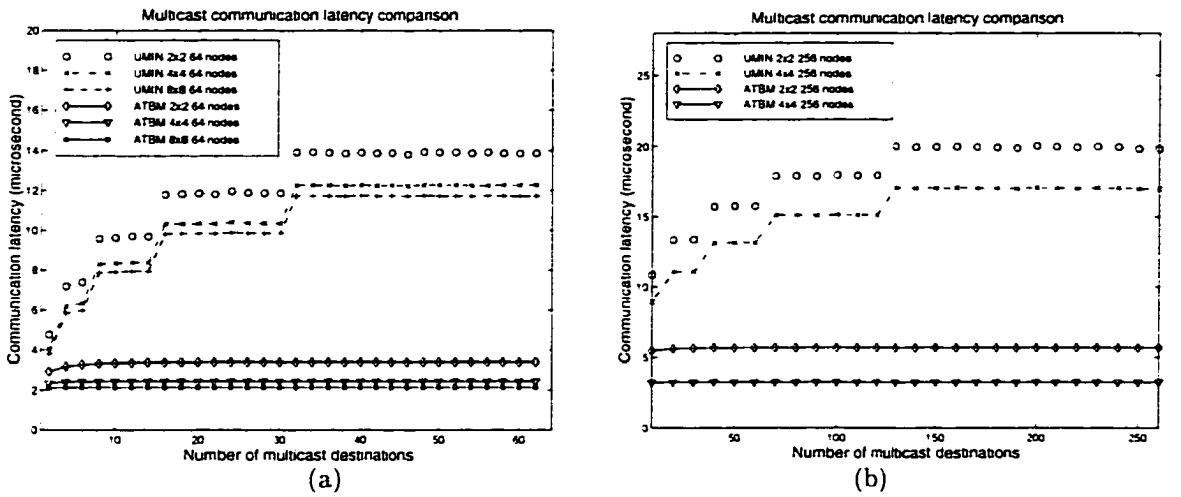


Figure 5.21 Performance comparison under contention-free condition (a) 64-node BI-MINs (b) 256-node BI-MINs.

scheme degrades under heavy traffic. However, the ATBM scheme provide better performance on a wide operating range compared to the CMIN scheme. Performance comparison of ATBM and UMIN algorithms for BI-MIN systems are given in Figures 5.25 (a) and (b). In BI-MIN systems, the ATBM scheme shows significant performance improvement over unicast-based scheme.

The simulation results of the MIN systems using 8 × 8 switches are shown in Figures 5.26 and 5.27. These results represent systems with high degree of connections. The same level of performance improvement has been observed for both unicast and multicast communication. For the 512-node MIN systems, both unicast and multicast latency results using the ATBM scheme are significantly lower than the software-based scheme. This is mainly because of the fact that the traffic created while using the ATBM schemes is much less than the traffic created by the unicast-based schemes.

5.4.2.4 Simulation Results with 20% Multicast Traffic

In most classes of parallel applications, multicast communication constitutes only a small portion of the total network traffic. In the next set of simulation results, we assumed that the multicast traffic accounts for 20% of the total network traffic ($M_p = 0.2$). Figures 5.28 and 5.29 shows the performance comparison for the MIN systems using 2 × 2 switches. The comparison of Figure 5.28 (5.29) and Figure 5.22 (5.23) reveals the effect of the multicast traffic ratio. The latency curves for 50% multicast traffic ratio saturate earlier than the curve with

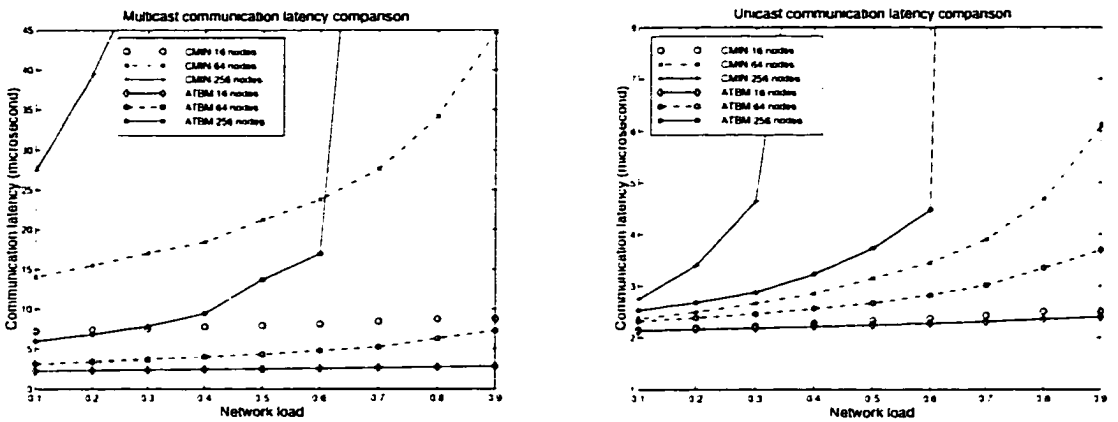


Figure 5.22 Performance comparison for 2 × 2-switch with $M_p = 0.5$ (a) multicast communication latency for UNI-MINs (b) unicast communication latency for UNI-MINs.

20% multicast traffic ratio. The ATBM approach performs better compared to unicast-based approach for 20%-80% ratio of multicast-unicast traffic as the operating region extends to cover higher network load.

The simulation results of MIN systems using 4×4 and 8×8 switches are shown in Figure 5.30 to Figure 5.33, respectively. With less traffic from multicast messages and the high degree of connectivity in large switches, the ATBM scheme can maintain the performance improvement ratio over a wide range of network load. The impact on the unicast communication is also reduced as shown in Figures 5.31 (b) and 5.33 (b). The unicast latency using unicast-based UMIN scheme is higher since the network is more congested due to the multicast traffic. Even in low multicast-unicast traffic ratio, this interference of multicast to unicast is still prominent in the software-based approach.

5.4.2.5 Effect of Message Size

One of the major advantages of the ATBM approach is that there is no limitation on the message size. The message sizes of 128 and 256 flits were simulated to study the effect of message size using the ATBM scheme. A single consumption channel is assumed and the multicast ratio is set to 50%. Figure 5.34 (a) shows the latency results of 128 flits messages in a 4 × 4 switch-based UNI-MINs. The ATBM scheme performs quite well for the message size of 128 flits. The additional latency incurred is only from the additional transmission delay of the messages. The same observation is obtained from the systems using 8 × 8 switches as shown

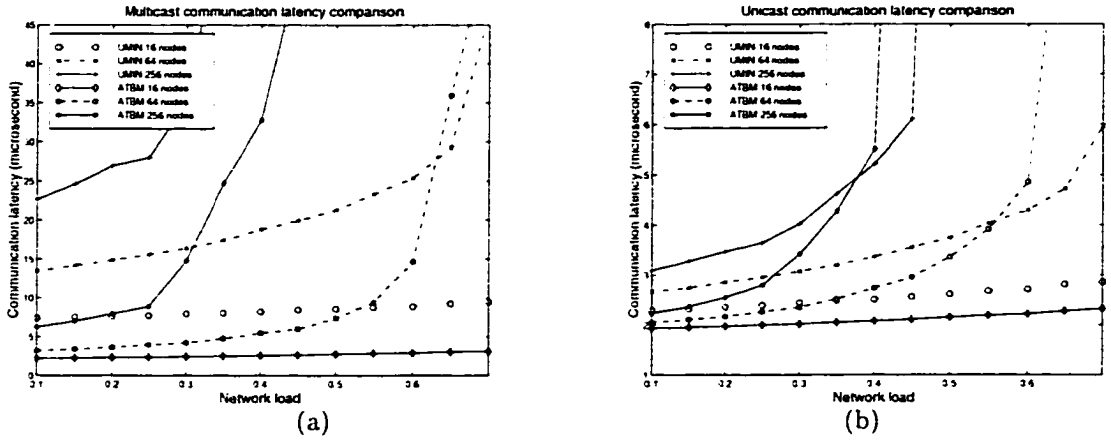


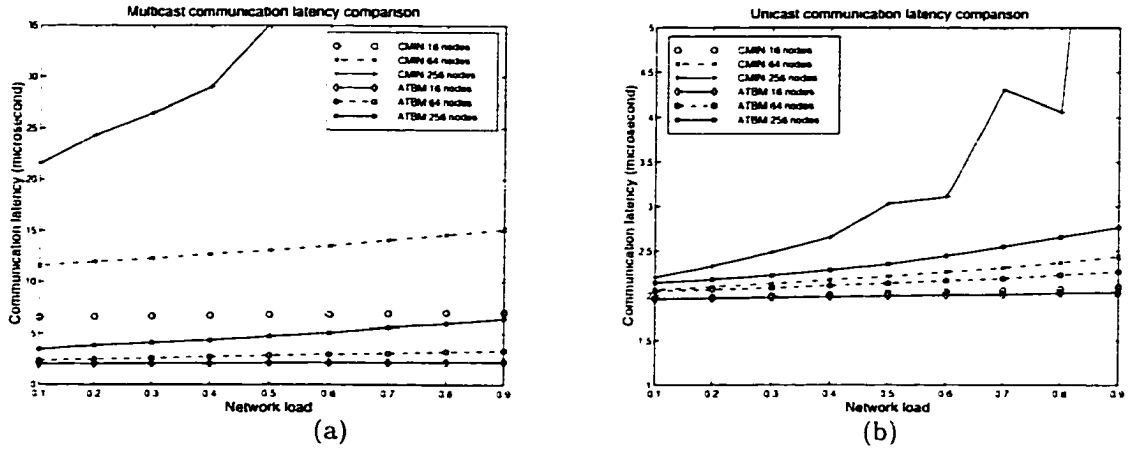
Figure 5.23 Performance comparison for 2 × 2-switch with $M_p = 0.5$ (a) multicast communication latency for BI-MINs (b) unicast communication latency for BI-MINs.

in Figure 5.34 (b). The results of BI-MIN systems using 4 × 4 switches and 8 × 8 switches are shown in Figures 5.35 (a) and (b), respectively.

The multicast latency results for 256 flits messages are shown in Figure 5.36. The results of UNI-MIN systems using 4 × 4 switches and 8 × 8 switches are shown in Figures 5.36 (a) and (b), respectively. The latency of the software-based multicast scheme increases due to the increase in transmission delay. The transmission delay accumulates through several phases of communications. For large size messages, the ATBM approach outperforms the software-based scheme for all system sizes. Since the ATBM approach permits the initiation of tree operations after all the headers of the current tree operation reach the destinations, non-conflicting multicast message can be routed incurring a small latency. Same level of performance improvement is observed for BI-MIN systems as shown in Figures 5.37 (a) and (b).

5.4.2.6 Evaluation of BI-MINs with b Consumption Channels

As mentioned in Section 5.2, the nodes using b consumption channels can significantly reduce the number of switches in the group which, in turn, will reduce the serialization and token passing overheads. We have simulated the MIN systems using nodes with b consumption channels with 50% multicast traffic ($M_p = 0.5$.). The multiple channels are implemented using additional consumption channels. Each consumption channel has its the communication link connected to the switch. The simulation results for MIN using 4 × 4 switches are shown in Figures 5.38 and 5.39. The multicast and unicast latency results for UMIN are shown in Figures



(a) (b)

Figure 5.24 Performance comparison for 4 × 4-switch with $M_p = 0.5$ (a) multicast communication latency for UNI-MINs (b) unicast communication latency for UNI-MINs.

5.38 (a) and (b), respectively. Figures 5.39 (a) and (b) show the performance comparison in BI-MIN systems. Each processing node has additional hardware to concurrently receive four messages. This model eliminates the deadlock problem in the last stage switches. Both unicast-based and ATBM schemes benefit from the b consumption channels as the network throughput increases. The ATBM performance improves significantly compared to the single-port model (shown in Figures 5.24 and 5.25). With the less number of switches in the group, the serialization waiting time is less and more number of multiple tree operations are allowed to be initiated. These two factors enhance the performance of the ATBM scheme. Similar trends have been observed for the systems using 8 × 8 switches, as shown in Figure 5.40.

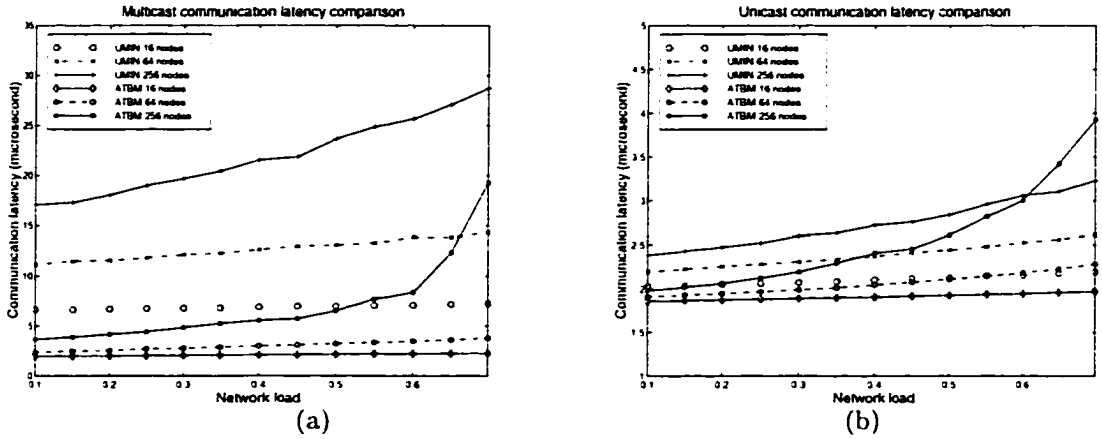


Figure 5.25 Performance comparison for 4 × 4-switch with $M_p = 0.5$ (a) multicast communication latency for BI-MINs (b) unicast communication latency for BI-MINs.

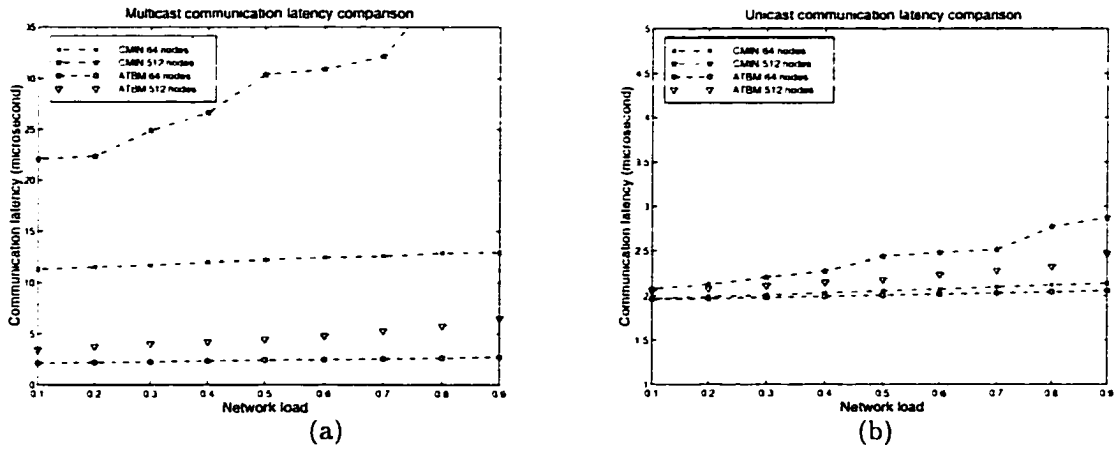
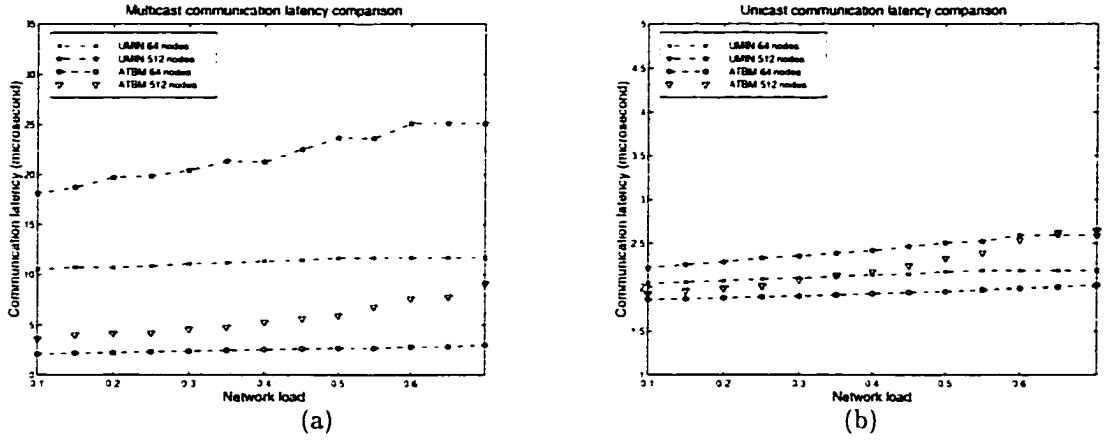
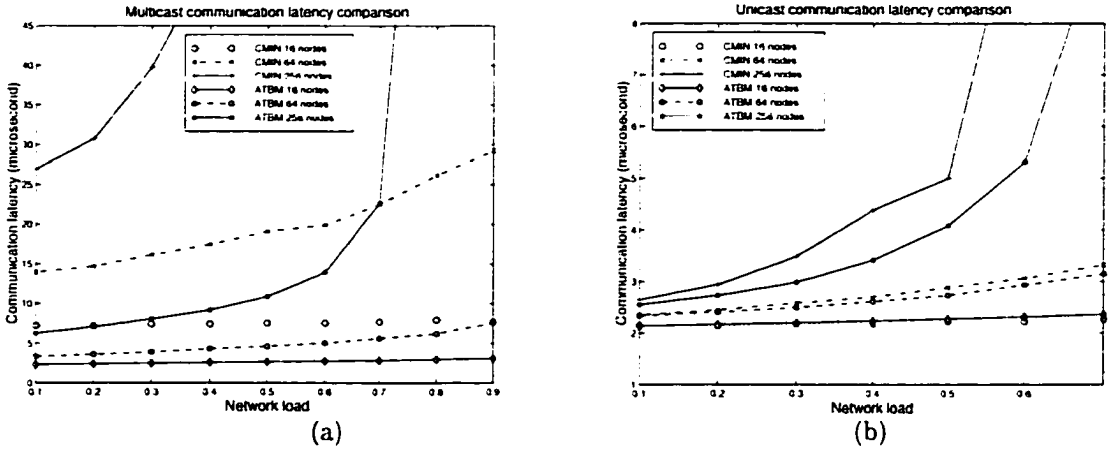


Figure 5.26 Performance comparison for 8 × 8-switch with $M_p = 0.5$ (a) multicast communication latency for UNI-MINs (b) unicast communication latency for UNI-MINs.



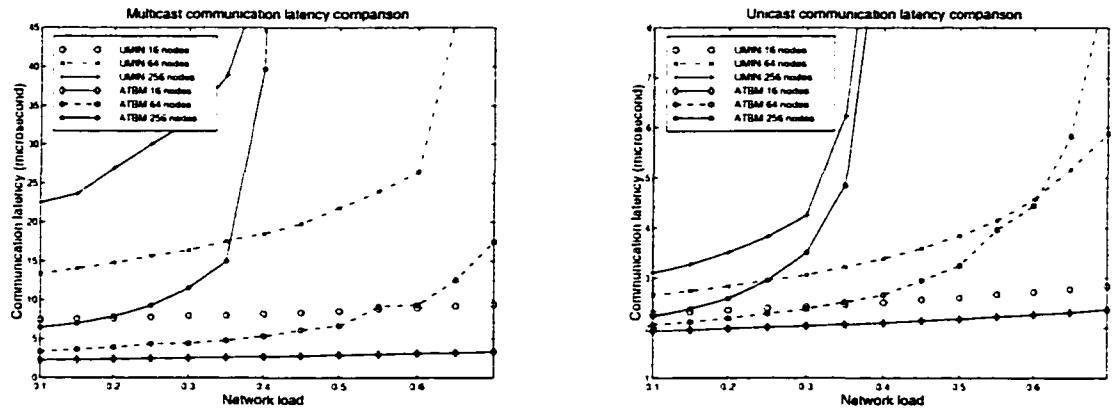
(a) (b)

Figure 5.27 Performance comparison for 8 × 8-switch with $M_p = 0.5$ (a) multicast communication latency for BI-MINs (b) unicast communication latency for BI-MINs.



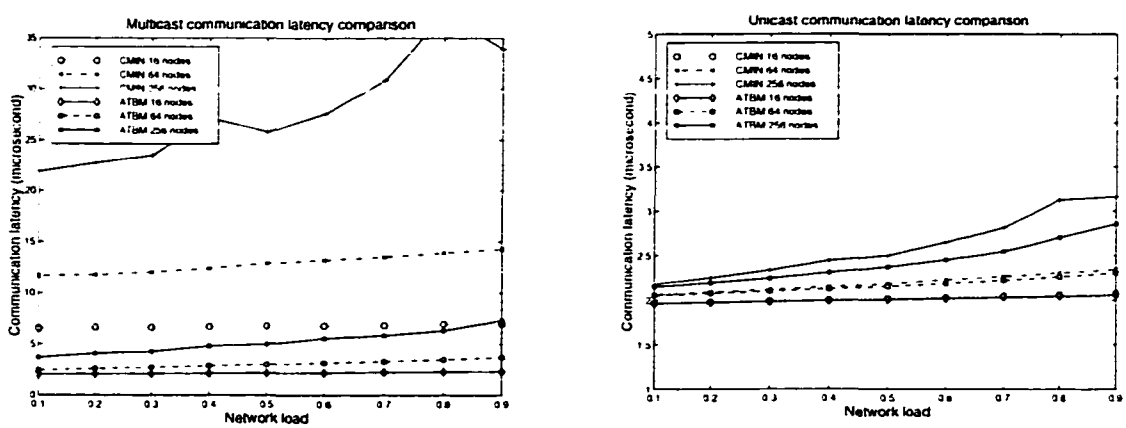
(a) (b)

Figure 5.28 Performance comparison for 2 × 2-switch with $M_p = 0.2$ (a) multicast communication latency for UNI-MINs (b) unicast communication latency for UNI-MINs.



(a) (b)

Figure 5.29 Performance comparison for 2 × 2-switch with $M_p = 0.2$ (a) multicast communication latency for BI-MINs (b) unicast communication latency for BI-MINs.



(a) (b)

Figure 5.30 Performance comparison for 4 × 4-switch with $M_p = 0.2$ (a) multicast communication latency for UNI-MINs (b) unicast communication latency for UNI-MINs.

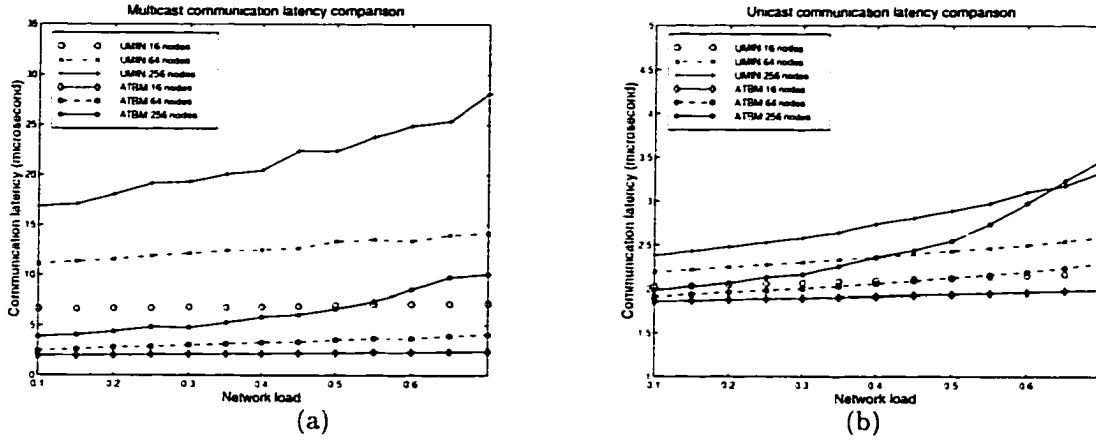


Figure 5.31 Performance comparison for 4 × 4-switch with $M_p = 0.2$ (a) multicast communication latency for BI-MINs (b) unicast communication latency for BI-MINs.

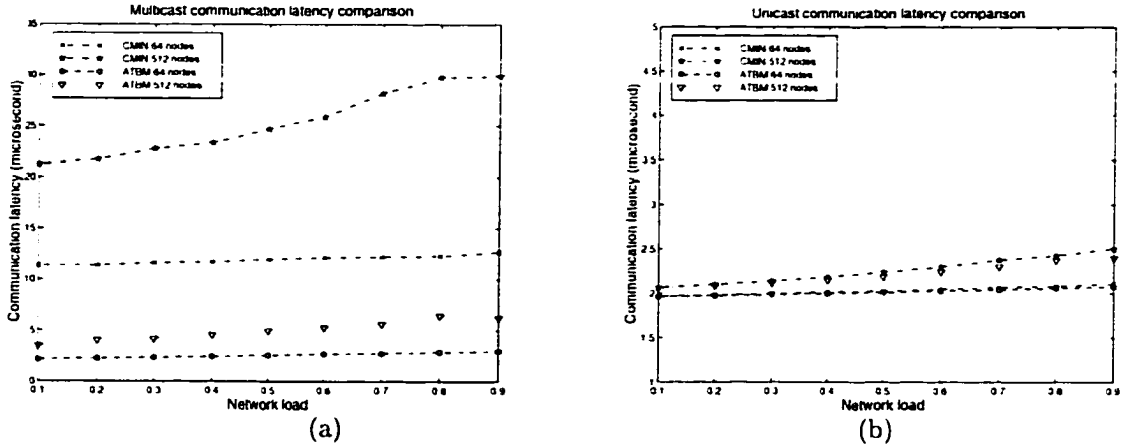


Figure 5.32 Performance comparison for 8 × 8-switch with $M_p = 0.2$ (a) multicast communication latency for UNI-MINs (b) unicast communication latency for UNI-MINs.

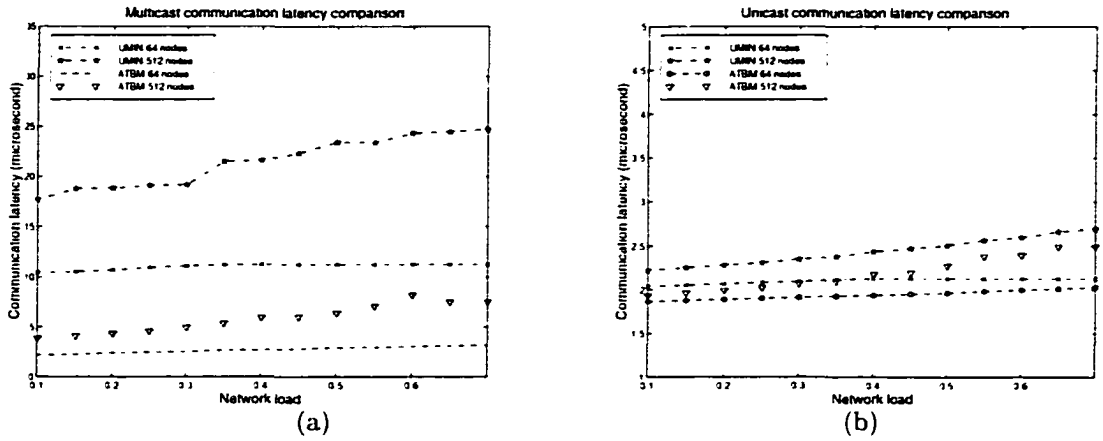


Figure 5.33 Performance comparison for 8 × 8-switch with $M_p = 0.2$ (a) multicast communication latency for BI-MINs (b) unicast communication latency for BI-MINs.

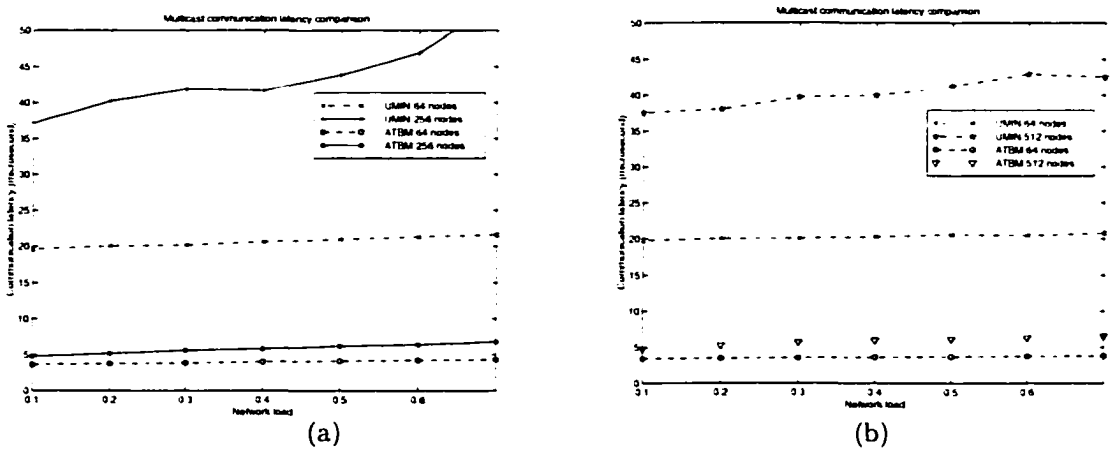
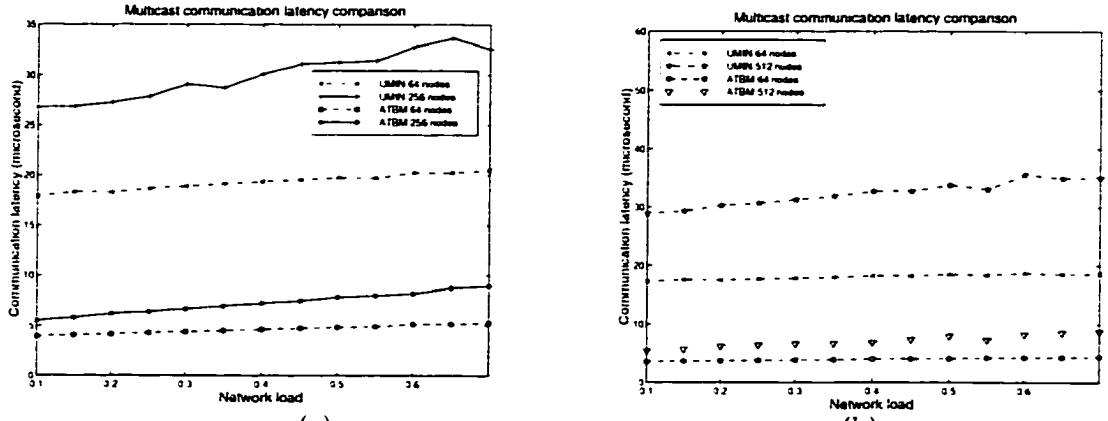
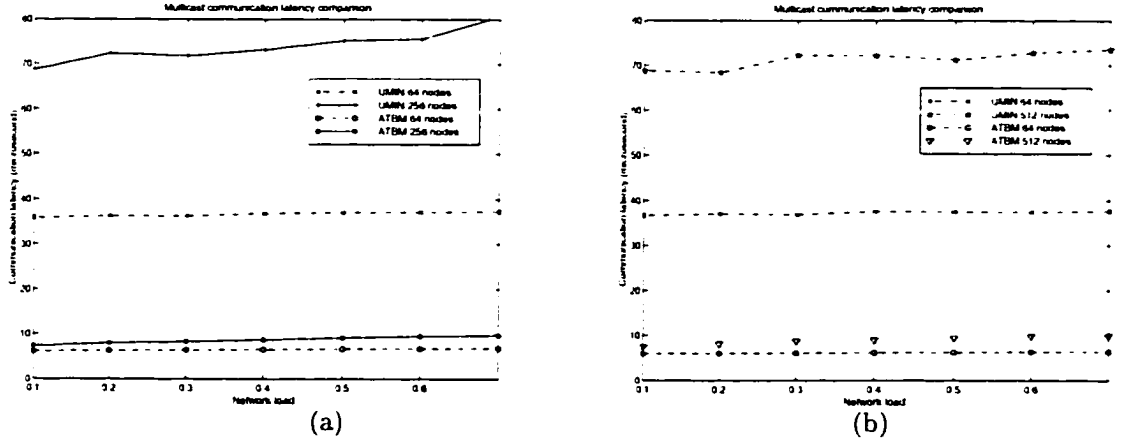


Figure 5.34 Multicast communication latency for 128 flits message with $M_r = 0.5$ (a) 4 × 4-switch UNI-MIN systems (b) 8 × 8-switch UNI-MIN systems.



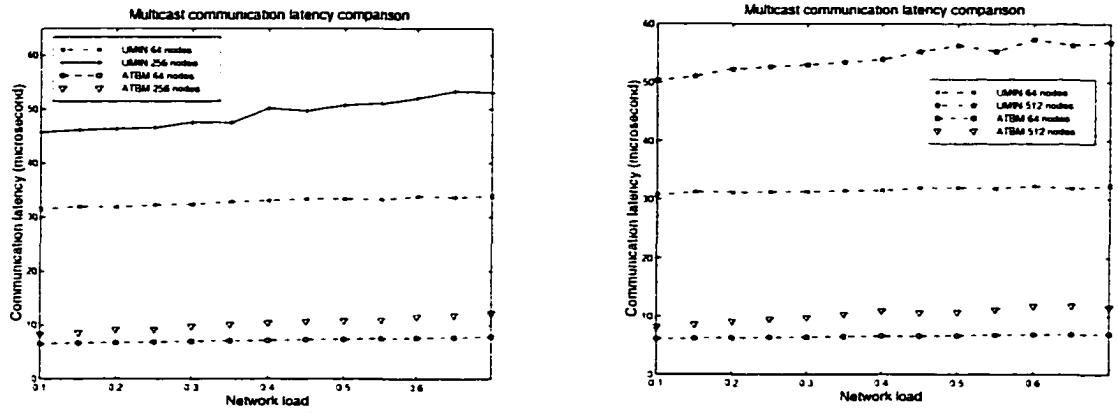
(a) (b)

Figure 5.35 Multicast communication latency for 128 flits message with $M_r = 0.5$ (a) 4×4-switch BI-MIN systems (b) 8×8-switch BI-MIN systems.



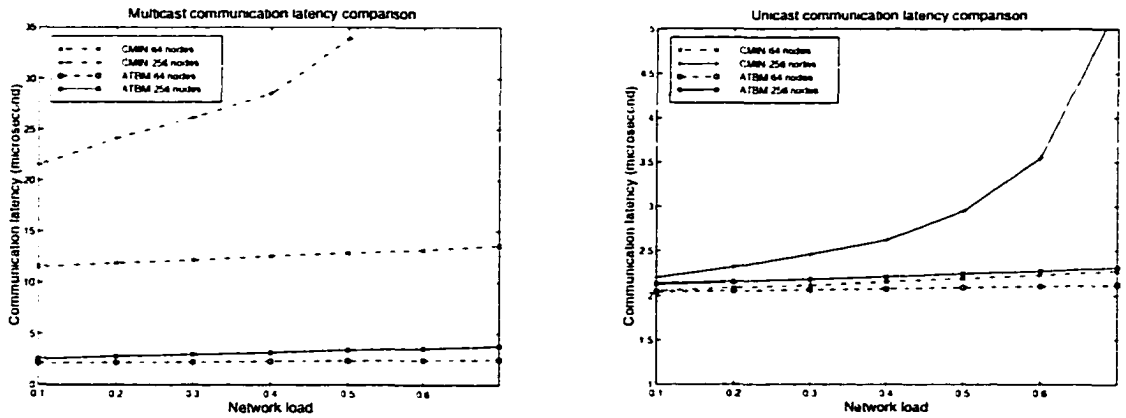
(a) (b)

Figure 5.36 Multicast communication latency for 256 flits message with $M_r = 0.5$ (a) 4×4-switch UNI-MIN systems (b) 8×8-switch UNI-MIN systems.



(a) (b)

Figure 5.37 Multicast communication latency for 256 flits message with $M_r = 0.5$ (a) 4×4-switch BI-MIN systems (b) 8×8-switch BI-MIN systems.



(a) (b)

Figure 5.38 Performance comparison for b consumption channels model, 4×4-switch with $M_r = 0.5$ (a) multicast communication latency for UNI-MINs (b) unicast communication latency for UNI-MINs.

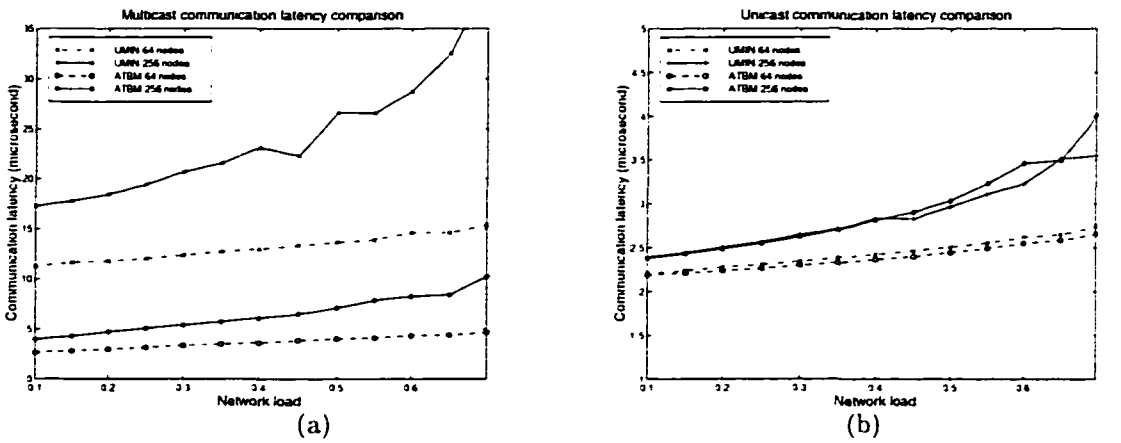


Figure 5.39 Performance comparison for b consumption channels model. 4×4-switch with $M_r = 0.5$ (a) multicast communication latency for BI-MINs (b) unicast communication latency for BI-MINs.

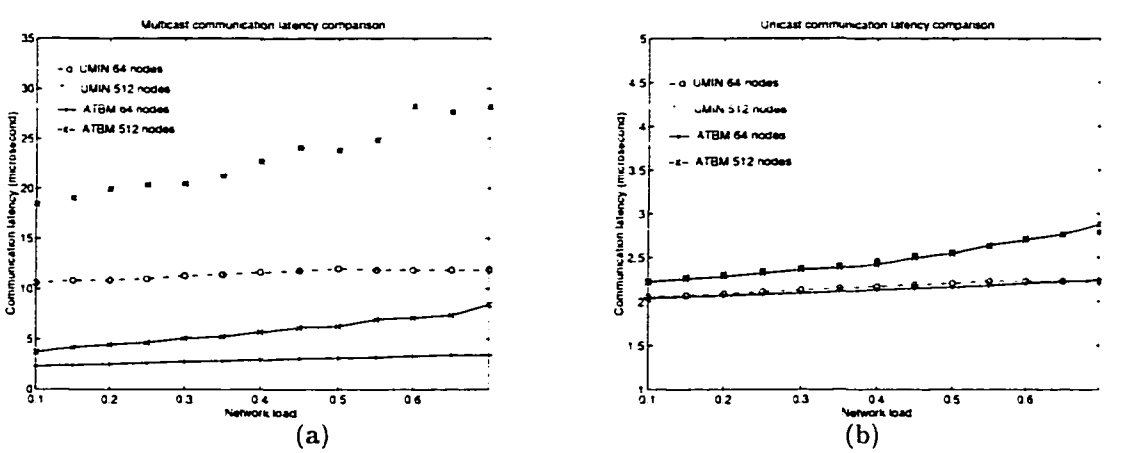


Figure 5.40 Performance comparison for b consumption channels model. 8×8-switch with $M_r = 0.5$ (a) multicast communication latency for BI-MINs (b) unicast communication latency for BI-MINs.

6 CONCLUSIONS AND FUTURE WORKS

This dissertation describes techniques fo · improving performance of communication subsystems used in multicomputers. The study is focused on wormhole-switched interconnection networks. We develop communication techniques that efficiently support both unicast and multicast communication in multicomputers. Several hardware techniques used to support multicast operations are proposed. This chapter concludes the research work and indicates research issues for future investigations.

6.1 Concluding Remarks

We have analyzed adaptive routing of unicast messages in two dimensional meshes in light of two new concepts -region of adaptivity and balanced traffic distribution. Using these concepts, we illustrate how various algorithms cause an uneven traffic distribution in the network and their impact on the system performance. Previously-proposed, fully-adaptive algorithms have aimed at improving the adaptivity of the routing schemes. We have demonstrated through simulations that symmetric and balanced traffic distribution have a significant impact on the system performance along with higher adaptiveness. Motivated by this observation, we have proposed a new adaptive routing algorithm, called Positive-First-Negative-First(PFNF) for two-dimensional meshes. The algorithm uses a combination of Positive-First and Negative-First routing to balance the traffic distribution in the network. The simulation results show that the proposed scheme performs better than the previous schemes in terms of the average network latency and throughput. The concept of region of adaptivity and how it relates to the network traffic distribution can be used in choosing these balanced algorithms.

For the two dimensional mesh networks, the basic low level communication primitives are studied and extended to a set of advanced communication primitives. The set of advanced communication primitives can be used to support efficient collective communication. In Chapter 4, a new algorithm called two-phase multicast (TPM) algorithm for multicasting in two-dimensional mesh networks is presented. The TPM algorithm uses the same routing algorithm as that employed for unicast communication. The algorithm is very simple and re-

quires at most two communication start-up steps to multicast to any member of destinations. During the first step, the message is sent to a node using a path such that the nodes covered during the first step can send the message to the remaining destinations in the second step, if required. The path taken in the first step guarantees that all the nodes will be reached within two start-up steps. The TPM scheme can be used in conjunction with the deterministic as well as adaptive routing schemes. We have applied the TPM framework to two adaptive routing algorithms. 3P and PFNF. The performance of the proposed algorithm is evaluated through simulations. We have considered realistic parameters and have included the associated overheads in our experiments. The results demonstrate that the TPM algorithm performs better than the previously proposed multicast routing algorithms. The simulation also indicates that the multicast operations can benefit from the adaptivity.

The tree-based multicasting technique is explored in Chapter 5. Efficient deadlock avoidance techniques for tree-based multicasting are proposed. The deadlock configurations that result from the tree operation are presented. The switch grouping technique is used to analyze the potential deadlock cycles. We have developed a framework for tree-based multicasting in MINs. Based on the switch groups, an asynchronous tree-based multicasting (ATBM) scheme is proposed for MINs. Deadlocks are prevented by serializing the initiations of tree operations within the same group at the same stage. We have developed complete algorithms for multicasting in unidirectional as well as bidirectional MINs. The performance of the proposed algorithm is evaluated through simulations. We have considered realistic parameters and have included the associated overheads in our experiments. The results demonstrate that the ATBM algorithm performs significantly better than the previously proposed software-based multicast routing algorithms.

6.2 Future Research

The following is the list of research issues to be investigated in future.

- **Collective communication in 2-D mesh networks:** It has been shown in our works that the hardware support multicast communication in 2-D mesh networks significantly reduce the communication latency. Other collective communication operations, such as gather, scatter, and barrier synchronization, can be benefited from this approach as well.
- **Multicasting in irregular networks:** Multicast issues in the irregular networks needs to be investigated.

BIBLIOGRAPHY

[1] G. Amdahl, "Validity of the single-processor approach to achieving large scale computing capabilities," in *Proceedings of the AFIPS Conference*, pp. 483–485, 1967.

[2] K. Hwang, *Advanced Computer Architecture, Parallelism, Scalability, Programmability.* New York: McGraw-Hill Inc., 1993.

[3] D. Lenoski, J. Laudon, K. Gharachorloo, W. Weber, A. Gupta, J. Hennessy, M. Horowitz, and M. Lam, "The stanford DASH multiprocessor," *IEEE Computer*, pp. 63–79, March 1992.

[4] "Cray T3D system architecture overview," Mountain View, CA: Cray Research, Inc., 1993.

[5] S. L. Scott, "Synchronization and communication in the T3E multiprocessor," in *Proceedings of the Seventh International Conference on Architectural Support for Programming Languages and Operating Systems*, pp. 26–36, October 1996.

[6] C. Wu and T. Feng, eds., *Tutorial: Interconnection networks for parallel and distributed processing*, Washington, D.C.: IEEE Computer Society Press, 1984.

[7] C. Kruskal and M. Snir, "The performance of multistage interconnection networks for multiprocessors," *IEEE Transaction on Computers*, vol. C-32, pp. 1091–1098, December 1983.

[8] A. S. Tanenbaum, *Computer Networks.* Englewood Cliffs, NJ: Prentice-Hall Inc., third ed., 1996.

[9] P. Kermani and L. Kleinrock, "Virtual cut-through: A new computer communication switching technique," *Computer Networks*, vol. 3, no. 4, pp. 267–286, 1979.

[10] L. M. Ni and P. K. McKinley, "A survey of wormhole routing techniques in direct networks," *IEEE Computer*, vol. 26, pp. 62–76, February 1993.

[11] W. J. Dally and C. L. Seitz, "The torus routing chip," *Journal of Distributed Computing*, vol. 1, no. 3, pp. 187–196, 1986.

[12] A. Agrawal, B. Lim, D. Kranz, and J. Kubiatowicz, "APRIL: A processor architecture for multiprocessing," in *Proceedings of the International Symposium on Computer Architecture*, pp. 104–114, June 1990.

[13] W. J. Dally, "The J-machine: A fine-grained concurrent computer," in *Proceedings of the Information Processing*, (Amsterdam, Netherlands), pp. 1147–1153, 1989.

[14] M. Fillo, S. W. Keckler, W. J. Dally, N. P. Carter, A. Chang, Y. Gurevich, and W. S. Lee., "The M-machine multicomputer," tech. rep., Artificial Intelligence Laboratory Memo 1532, Massachusetts Institute of Technology, 1995.

[15] "nCUBE 6400 processor manual." Beaverton, OR: NCUBE Company, 1990.

[16] "Paragon XP/S product overview." Beaverton, OR: Intel corporation, 1991.

[17] R. E. Kessler and J. L. Schwarzmeier, "Cray T3D: A new dimension for cray research," *Compcon*, pp. 176–182, Spring 1993.

[18] N. Koike, "NEC Cenju-3," in *Proceedings of the 8th International Parallel Processing Symposium*, pp. 396–401, April 1994.

[19] R. Alverson, D. Callahan, D. Cummings, B. Koblenz, A. Porterfield, B. Smith, "The tera computer system," in *Proceedings of the International Conference on Supercomputing*, pp. 1–6, June 1990.

[20] C. B. Stunkel, D. G. Shea, P. H. Hochschild, and M. Tsao, "The SP1 high-performance switch," in *Proceedings of the Scalable High Performance Computing Conference*, pp. 150–157, 1994.

[21] C. B. Stunkel, D. G. Shea, B. Abali, M. G. Atkins, C. A. Bender, D. G. Grice, P. Hochschild, D. J. Joseph, B. J. Nathanson, R. A. Swetz, R. F. Stucke, M. Tsao, and P. R. Varker, "The SP2 high-performance switch," *IBM System Journal*, vol. 34, pp. 185–204, February 1995.

[22] C. E. Leiserson, Z. S. Abuhamdeh, D. C. Douglas, C. R. Feynman, M. N. Ganmukhi, J. V. Hill, W. D. Hillis, B. C. Kuazmaul, M. A. St. Pierre, D. S. Wells, M. C. Wong-Chan, S. Yang, and R. Zak, "The network architecture of the connection machine CM-5," in *the ACM Symposium on Parallel Algorithm and Architecture*, pp. 272–285, 1992.

[23] "Computing surface: CS-2 communication networks." Waltham, MA: Meiko Limited, 1993.

[24] M. D. Schroeder, A. D. Birrell, M. Burrows, H. Murray, R. M. Needham, T. L. Rodeheffer, E. H. Satterthwaite, and C. P. Thacker, "Autonet: A high-speed self-configuring local area network using point-to-point links." tech. rep., SRC Research Report 59, 1990.

[25] N. J. Boden, D. Cohen, R. E. Felderman, A. E. Kulawlk, C. L. S. ad J. N. Seizovic, and W. Su, "Myrinet: A Gigabit-per-second local area network," *IEEE Micro*, vol. 15, pp. 29–36, February 1994.

[26] P. K. McKinley, Y. Tsai, and D. F. Robinson, "Collective communication in wormhole-routed massively parallel computers." *IEEE Computer*, vol. 28, pp. 39–50, December 1995.

[27] P. Mitra, D. Payne, L. Shuler, R. van de Geijn, and J. Watts, "Fast collective communication libraries, please." in *Proceedings of the Intel Supercomputing Users' Group Meeting*, 1995.

[28] "MPI: Message passing interface forum." Message passing interface forum, http://www.mpi-forum.org, 1994.

[29] L. M. Ni, "Should scalable parallel computer support efficient hardware multicast?." in *Proceedings of the ICPP Workshop on Challenges for Parallel Processing*, pp. 2–7, 1995.

[30] D. K. Panda, "Issues in designing efficient and practical algorithms for collective communication on wormhole-routed systems." in *Proceedings of the ICPP Workshop on Challenges for Parallel Processing*, pp. 8–15, 1995.

[31] D. K. Panda, "Fast barrier synchronization in wormhole k-ary n-cube networks with multidestination worms," in *Proceedings of the International Symposium on High Performance Computer Architectures*, pp. 200–209, April 1995.

[32] J. H. Upadhyay, V. Varavithya, and P. Mohapatra, "Efficient and balanced adaptive routing in two-dimensional meshes." in *Proceedings of the International Symposium on High Performance Computer Architecture*, pp. 112–121, 1995.

[33] J. H. Upadhyay, V. Varavithya, and P. Mohapatra, "A traffic-balanced adaptive wormhole-routing for two-dimensional meshes," *IEEE Transactions on Computers*, vol. 46, pp. 190–197, February 1997.

[34] P. Mohapatra and V. Varavithya, "A hardware multicast routing algorithm for two-dimensional meshes," in *Proceedings of the Eighth Symposium on Parallel and Distributed Processing*, pp. 198–205, October 1996.

[35] V. Varavithya and P. Mohapatra, "Tree-based multicasting on wormhole routed multistage interconnection networks," in *Proceedings of the International Conference on Parallel Processing*, (Bloomingdale, IL), August 1997.

[36] V. Varavithya and P. Mohapatra, "A tree-based multicasting algorithm for wormhole-routed bidirectional MINs," Tech. Rep. TR-ACAR-97-03, Department of Electrical and Computer Engineering, Iowa State University, June 1997.

[37] J. Duato, S. Yalmachili, and L. M. Ni, *Interconnection networks: An Engineering Approach*. Los Alamitos, California: IEEE Computer Society, 1997.

[38] P. Mohapatra, "Wormhole routing techniques for directly connected multicomputer systems," To appear in the *ACM Computing Survey*.

[39] V. Karamcheti and A. A. Chien, "Do faster router imply faster communication?," in *Proceedings of the parallel computer routing and communication workshop*, May 1994.

[40] V. Karamcheti and A. A. Chien, "Software overhead in messaging layers: Where does the time go?," in *Proceedings of ASPLOS-VI*, (San Jose, California), October 1994.

[41] D. K. Panda, S. Singal, and P. Prabhakaran, "Multidestination message passing mechanism conforming to base wormhole routing scheme," in *Proceedings of Parallel Computer Routing and Communication Workshop*, pp. 131–145, May 1994.

[42] N. Nupairoj and L. M. Ni, "Benchmarking of multicast communication services," Tech. Rep. MSU-CPS0ACS-103, Dept. of Computer Science, Michigan State University, April 1995.

[43] S. Pakin, V. Karamcheti, and A. A. Chien, "Fast messages (FM): Efficient, portable communication for workstation clusters and massively-parallel processors," *IEEE Concurrency*, vol. 5, no. 2, April-June 1997.

[44] D. Culler, K. Keeton, L. T. Liu, A. Mainwaring, R. Martin, S. Rodrigues, K. Wright, and C. Yoshikawa, "The generic active message interface specification," tech. rep., Computer Science Division, University of California at Berkeley, http://now.CS.Berkeley.EDU/Papers2/index.html, 1994.

[45] S. S. Mukherjee and M. D. Hill, "The impact of data transfer and buffering alternatives on network interface design," in *Proceedings of the 4th International Symposium on High-Performance Computer Architecture (HPCA-4)*, pp. 207–218, February 1998.

[46] I. Schoinas and M. D. Hill, "Address translation mechanisms in network interface design," in *Proceedings of the 4th International Symposium on High-Performance Computer Architecture (HPCA-4)*, pp. 219–230, February 1998.

[47] K. Mackenzie, J. Kubiatowicz, M. Frank, W. Lee, V. Lee, A. Agarwal, and M. F. Kaashoek, "Exploiting two-case delivery for fast protected messaging," in *Proceedings of the 4th International Symposium on High-Performance Computer Architecture (HPCA-4)*, pp. 231–243, February 1998.

[48] W. J. Dally, "Performance analysis of k-ary n-cube interconnection networks," *IEEE Transactions on Computers*, vol. 39, pp. 775–785, June 1990.

[49] L. M. Ni, Y. Gui, and S. Moore, "Performance evaluation of switch-based wormhole networks," in *Proceedings of the International Conference on Parallel Processing*, August 1995.

[50] W. J. Dally and C. L. Seitz, "Deadlock-free message routing in multiprocessor interconnection networks," *IEEE Transactions on Computers*, vol. 36, pp. 547–553, May 1987.

[51] J. H. Kim and A. A. Chien, "The impact of packetization in wormhole-routed networks," in *Proceedings of the PARLE*, June 1993.

[52] K. G. Shin and S. W. Daniel, "Analysis and implementation of hybrid switching," in *Proceedings of the International Symposium on Computer Architecture*, pp. 211–219, 1995.

[53] J. Rexford and K. G. Shin, "Support for multiple support classes of traffic in multicomputer routers," in *Proceedings of the parallel computer routing and communication workshop*, pp. 116–128, May 1994.

[54] W. J. Dally, "Virtual channel flow control," *IEEE Transactions on Parallel and Distributed Systems*, vol. 3, pp. 194–205, March 1992.

[55] D. D. Kandlur and K. G. Shin, "Traffic routing for multicomputer networks with virtual cut-through capability," *IEEE Transactions on Computers*, vol. 41, pp. 1257–1270, October 1992.

[56] W. J. Dally, R. Dennison, D. Harris, K. Kan, and T. Xanthopoulos, "The reliable router: A reliable and high-performance communication substrate for parallel computers," in *Proceedings of the Parallel Computer Routing and Communication Workshop*, pp. 241–255, May 1994.

[57] D. H. Linder and J. C. Harden, "An adaptive and fault tolerant wormhole routing strategy for k-ary n cubes," *IEEE Transactions on Computers*, vol. 40, pp. 2–12, January 1991.

[58] K. V. Anjan and T. M. Pinkston, "*DISHA:* A deadlock recovery scheme for fully adaptive routing," in *Proceedings of the International Parallel Processing Symposium*, April 1995.

[59] W. J. Dally and H. Akoi, "Deadlock-free adaptive routing in multicomputer networks using virtual channels," *IEEE Transactions on Parallel and Distributed Systems*, vol. 4, pp. 466–475, April 1993.

[60] C. Su and K. G. Shin, "Adaptive deadlock-free routing in multicomputers using only one extra channel," in *Proceedings of the International Conference on Parallel Processing*, vol. 3, pp. 175–182, August 1993.

[61] Y. M. Boura and C. R. Das, "Efficient fully adaptive wormhole routing in n-dimensional meshes," in *Proceedings of the 14th International Conference on Distributed Computing Systems*, 1994.

[62] J. Duato, "A new theory of deadlock-free adaptive routing in wormhole networks," *IEEE Transactions on Parallel and Distributed Systems*, vol. 4, pp. 1320–1331, December 1993.

[63] K. Aoyama and A. A. Chien, "The cost of adaptivity and virtual lanes," *Journal of VLSI Design*, vol. 2, no. 4, 1994.

[64] A. A. Chien, "A cost and speed model for k-ary n-cube wormhole routers," in *Proceedings of the Hot Interconnects*, (Palo Alto, CA), August 1993.

[65] S. Konstantinidou and L. Snyder, "The chaos router," *IEEE Transactions on Computers*, vol. 43, pp. 1386–1397, December 1994.

[66] K. V. Anjan and A. M. Pinkston, "An efficient, fully adaptive deadlock recovery scheme: DISHA," in *Proceedings of the International Symposium on Computer Architecture*, pp. 201–210, 1995.

[67] J. H. Kim, Z. Liu, and A. A. Chien, "Compressionless routing: A framework for adaptive and fault-tolerant routing," in *Proceedings of the International Symposium on Computer Architecture*, pp. 289–300, 1994.

[68] P. Lopez, J. M. Martinez, and J. Duato, "A very efficient distributed deadlock detection mechanism for wormhole networks," in *Proceedings of the 4th International Symposium on High-Performance Computer Architecture (HPCA-4)*, pp. 57–67, February 1998.

[69] K. D. Gunther, "Prevention of deadlock in packets-switched data transport systems," *IEEE Transactions on Communication"*, vol. COM-29, pp. 512–524, April 1981.

[70] J. Duato, "A theory of deadlock-free adaptive multicast routing in wormhole networks," *IEEE Transactions on Parallel and Distributed Systems*, pp. 976–987, September 1995.

[71] J. Duato, "A theory of fault-tolerant routing in wormhole networks," in *Proceedings of the International Conference on Parallel and Distributed Systems*, pp. 600–607, December 1994.

[72] J. Duato, "A necessary and sufficient condition for deadlock-free adaptive routing in wormhole networks," in *Proceedings of the International Conference on Parallel Processing*, vol. 1, pp. 142–149, December 1994.

[73] J. Duato, "A necessary and sufficient condition for deadlock-free adaptive routing in wormhole networks," *IEEE Transactions on Parallel and Distributed Systems*, vol. 6, pp. 1055–1067, October 1995.

[74] J. Duato, "A necessary and sufficient condition for deadlock-free routing in cut-through and store-and-forward networks," *IEEE Transactions on Parallel and Distributed Systems*, vol. 7, pp. 841–854, August 1996.

[75] L. Schwiebert and D. N. Jayasimha, "A necessary and sufficient condition for deadlock-free meshes," *Journal of Parallel and Distributed Computing*, vol. 32, pp. 103–117, January 1996.

[76] C. J. Glass and L. M. Ni, "Adaptive routing in mesh-connected networks," in *Proceedings of the International Conference on Distributed Computing Systems*, pp. 12–19, 1992.

[77] C. J. Glass and L. M. Ni, "The turn model for adaptive routing," *Journal of the ACM*, vol. 41, pp. 874–902, September 1994.

[78] X. Lin, P. K. McKinley, and L. M. Ni, "The message flow model for routing in wormhole-routed networks," in *Proceedings of the International Conference on Parallel Processing*, vol. 1, pp. 294–297, 1993.

[79] Y. M. Boura and C. R. Das, "A class of partially adaptive routing algorithms for n-dimensional meshes," in *Proceedings of the International Conference on Parallel Processing*, vol. 3, pp. 175–182, August 1993.

[80] A. A. Chien and J. H. Kim, "Planar adaptive routing: Low-cost adaptive networks for multiprocessors," *Journal of the ACM*, pp. 91–123, January 1995.

[81] A. A. Chien and J. H. Kim, "Planar adaptive routing: Low-cost adaptive networks for multiprocessors," in *Proceedings of the International. symposium on Computer Architecture*, pp. 268–277, May 1992.

[82] L. Schwiebert and D. N. Jayasimha, "Optimally fully adaptive minimal wormhole routing for meshes," *Journal of Parallel and Distributed Computing*, vol. 27, pp. 56–70, 1995.

[83] K. Bolding, M. Fulgham, and L. Snyder, "The case for chaotic adaptive routing," *IEEE Transactions on Computers*, vol. 46, December 1997.

[84] P. K. McKinley, H. Xu, A. Esfahanian, and L. M. Ni, "Unicast-based multicast communication in wormhole routed networks," in *Proceedings of the International Conference on Parallel Processing*, vol. 2, pp. 10–19, 1992.

[85] P. K. McKinley, H. Xu, A. Esfahanian, and L. M. Ni, "Unicast-based multicast communication in wormhole routed networks," *IEEE Transactions on Parallel and Distributed Systems*, vol. 5, pp. 1252–1265, December 1994.

[86] H. Xu, Y. Gui, and L. M. Ni, "Optimal software multicast in wormhole routed multistage networks," in *Proceedings of Supercomputing*, pp. 703–712, 1994.

[87] C. Chiang and L. M. Ni, "Efficient software multicast in wormhole-routed unidirectional multistage networks," in *Proceedings of the Symposium of Parallel and Distributed Processing*, pp. 106–113, 1995.

[88] R. Kesavan, K. Bondalapati, and D. K. panda, "Multicast on irregular switch-based networks with wormhole routing," in *Proceedings of the International Symposium on High Performance Computer Architecture*, pp. 48–57, 1997.

[89] X. Lin, P. K. McKinley, and L. M. Ni. "Performance evaluation of multicast wormhole routing in 2d-mesh multicomputers," in *Proceedings of the 1991 International Conference on Parallel Processing*, vol. I, pp. 435–442, August 1991.

[90] Y. Tsai and P. K. McKinley, "An extended dominating node approach to broadcast and global combine in multiport wormhole-routed mesh networks," *IEEE Transactions on Parallel and Distributed Systems*, vol. 8, pp. 41–57, January 1997.

[91] P. Fraigniaud, "Complexity analysis of broadcasting in hypercubes with restricted communication capabilities," *Journal of Parallel and Distributed Computing*, vol. 16, pp. 15–26, 1992.

[92] J. Park, H. Choi, N. Nupairoj, and L. Ni, "Construction of optimal multicast trees based on the parameterized communication model," in *Proceedings of the International Conference on Parallel Processing*, 1996.

[93] Y. Tseng, D. K. Panda, and T. Lai, "A trip-based multicasting models in wormhole-routed networks with virtual channels," *IEEE Transactions on Parallel and Distributed Systems*, vol. 7, pp. 138–150, February 1996.

[94] C. Chiang and L. M. Ni, "Multi-address encoding for multicast," in *Proceedings of the Parallel Computer Routing and Communication Workshop*, pp. 146–160, May 1994.

[95] X. Lin, P. K. McKinley, and L. M. Ni, "Deadlock-free multicast wormhole routing in 2-D mesh multicomputers," *IEEE Transactions on Parallel and Distributed Systems*, vol. 5, pp. 793 804, August 1994.

[96] R. V. Boppana, S. Chalasani, and C. S. Raghavendra, "On multicast wormhole routing in multicomputer networks," in *Proceedings of the Sixth Symposium on Parallel and Distributed Processing*, October 1994.

[97] X. Lin, P. K. McKinley, and A. Esfahanian, "Adaptive multicast wormhole routing in 2-D mesh multicomputers," *Journal of Parallel and Distributed Computing*, vol. 28, pp. 19–31, 1995.

[98] C. Chiang and L. M. Ni, "Deadlock-free multi-head wormhole routing," in *Proceedings of the First High Performance Computing-Asia*, 1995.

[99] M. P. Malumbres, J. Duato, and J. Torrellas, "An efficient implementation of tree-based multicast routing in distributed shared-memory multiprocessors," in *Proceedings of the Eighth Symposium on Parallel and Distributed Processing*, pp. 186–189, October 1996.

[100] D. K. Panda and R. Sivaram, "Fast broadcast and multicast in wormhole multistage networks with multidestination worms," Tech. Rep. OSU-CISRC-04/95-TR21, Department of Computer and Information Science, Ohio State University, 1995.

[101] R. Sivaram, D. K. Panda, and C. B. Stunkel, "Fast broadcast and multicast on wormhole multistage networks using multiport encoding," in *Proceedings of the Eighth IEEE Symposium on Parallel and Distributed Processing*, pp. 36–45, October 1996.

[102] R. V. Boppana and S. Chalasani, "A comparison of adaptive wormhole routing algorithms," in *Proceedings of the International Symposium on Computer Architecture*, pp. 351–360, May 1993.

[103] V. Varavithya, "Wormhole routing algorithms for mesh interconnection networks," Master's thesis, Iowa State University, Ames, IA, 1994.

[104] C. J. Glass and L. M. Ni, "Maximally fully adaptive routing in 2-D meshes," in *Proceedings of the International Conference on Parallel Processing*, August 1992.

[105] V. Varavithya, J. Upadhyay, , and P. Mohapatra, "An efficient fault-tolerant routing scheme for two-dimensional meshes," in *Proceedings of the International Conference of High Performance Computing*, (New Delhi, India), pp. 773–778, December 1995.

[106] J. Upadhyay, V. Varavithya, and P. Mohapatra, "Routing algorithms for torus networks," in *Proceedings of the International Conference of High Performance Computing*, (New Delhi, India), pp. 743–748, December 1995.

[107] J. J. Dongarra, S. W. Otto, M. Snir, and D. Walker, "A message passing standard for MPP and workstations," *Communications of the ACM*, vol. 39, pp. 84–90, July 1996.

[108] R. V. Boppana, S. Chalasani, and C. S. Raghavendra, "Resource deadlock and performance of wormhole multicast routing algorithms," Tech. Rep. CS-95-6, Division of Computer Science, University of Texas at San Antonio, July 1995.

[109] H. D. Schwetman, "Introduction to process-oriented simulation and CSIM," in *Proceedings of the Winter Simulation Conference*, pp. 154–157, December 1990.

[110] X. Lin and L. M. Ni, "Multicast communication in multicomputer networks," *IEEE Transactions on Parallel and Distributed Systems*, vol. 4, pp. 1105–1117, October 1993.

[111] C. B. Stunkel, R. Sivaram, and D. K. Panda, "Implementing multidestination worms in switch based parallel systems: Architectural alternatives and their impact," in *Proceedings of the International Symposium on Computer Architecture*, pp. 50–61, June 1997.

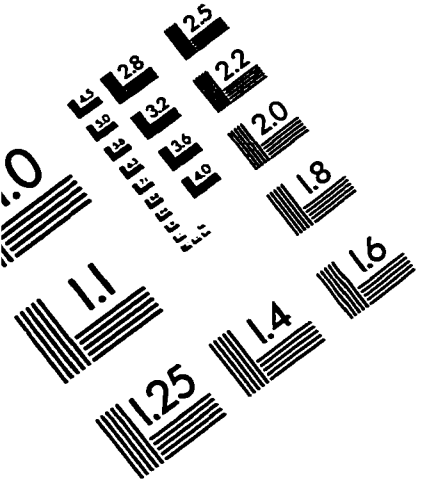
[112] S. Balakrishnan and D. K. Panda, "Impact of multiple consumption channels on wormhole routed k-ary n-cube networks," in *Proceedings of the International Parallel Processing Symposium*, pp. 163–167, 1993.

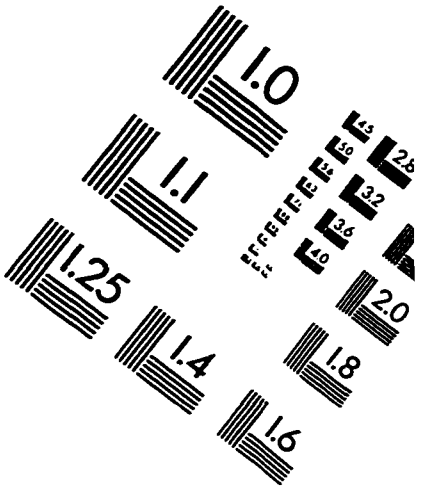
[113] H. J. Siegel, "The theory underlying the partitioning of permutation networks," *IEEE Transactions on Computers*, vol. C-29, pp. 791–800, September 1980.

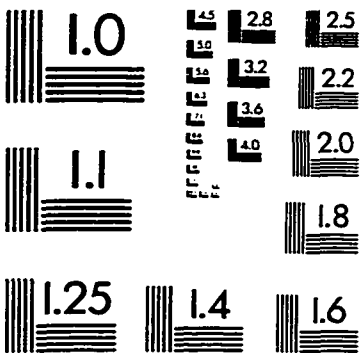
BIOGRAPHICAL SKETCH

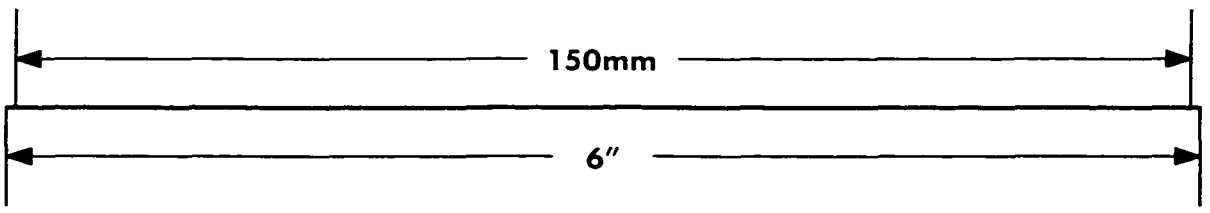
Vara Varavithya was born on February 19, 1968, Bangkok Thailand. He received his B.S. in Electrical Engineering with honors from the King Mongkut's Institute of Technology North Bangkok in 1989 and his M.S. and Ph.D. in Computer Engineering from Iowa State University in 1994 and 1998, respectively. From 1989 to 1991, he had been employed as an electrical engineer at the National Petrochemical Company, Rayong Thailand. Since 1991, he has joined the Department of Electrical Engineering King Mongkut's Institute of Technology, North Bangkok. He was selected as a Royal Thai Government Scholarship Recipient in 1993 and pursued his graduate study at Iowa State University. He has served in several research assistant and teaching assistant positions in the Department of Electrical and Computer Engineering. Upon his Ph.D. completion, he was awarded the Graduate Research Excellence Award from the Graduate College of Iowa State University. Vara is a member of the IEEE and the IEEE Computer Society.

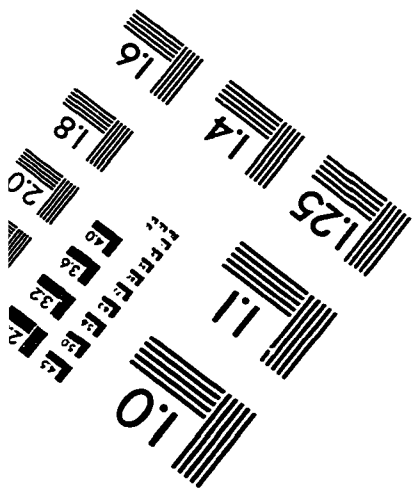
IMAGE EVALUATION TEST TARGET (QA-3)

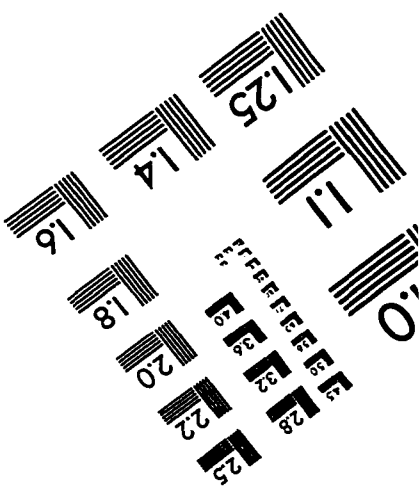












APPLIED IMAGE, Inc
1653 East Main Street
Rochester, NY 14609 USA
Phone: 716/482-0300
Fax: 716/288-5989