1

PM-1 3½"x4" PHOTOGRAPHIC MICROCOPY TARGET
NBS 1010a ANSI/ISO #2 EQUIVALENT

PRECISIONSM RESOLUTION TARGETS

PIONEERS IN METHYLENE BLUE TESTING SINCE 1974

10800 COUNTY ROAD 3, BURRISVILLE, MD 21037 USA
TEL 812 438 7887 FAX 812 438 7887 TLR 5168889488

National Library
of Canada

Bibliothèque nationale
du Canada

Canadian Theses Service Service des thèses canadiennes

Ottawa, Canada
K1A 0N4

NOTICE

The quality of this microform is heavily dependent upon the quality of the original thesis submitted for microfilming. Every effort has been made to ensure the highest quality of reproduction possible.

If pages are missing, contact the university which granted the degree.

Some pages may have indistinct print especially if the original pages were typed with a poor typewriter ribbon or if the university sent us an inferior photocopy.

Reproduction in full or in part of this microform is governed by the Canadian Copyright Act, R.S.C. 1970, c. C-30, and subsequent amendments.

AVIS

La qualité de cette microforme dépend grandement de la qualité de la thèse soumise au microfilmage. Nous avons tout fait pour assurer une qualité supérieure de reproduction.

S'il manque des pages, veuillez communiquer avec l'université qui a conféré le grade.

La qualité d'impression de certaines pages peut laisser à désirer, surtout si les pages originales ont été dactylographiées à l'aide d'un ruban usé ou si l'université nous a fait parvenir une photocopie de qualité inférieure.

La reproduction, même partielle, de cette microforme est soumise à la Loi canadienne sur le droit d'auteur, SRC 1970, c. C-30, et ses amendements subséquents.

**PERFORMANCE OF SOME DEADLOCK PREVENTION ROUTING
ALGORITHMS FOR MULTICOMPUTER SYSTEMS**

by

Rajiv Kachru, M.Sc.(Comp. Eng.)

**A thesis submitted to the Faculty of Graduate Studies and Research
in partial fulfillment of
the requirements for the degree of**

Master of Computer Science

**School of Computer Science
Carleton University
Ottawa, Ontario**

**April 10 1992
© Copyright
1992, Kachru**

National Library
of Canada

Bibliothèque nationale
du Canada

Canadian Theses Service Service des thèses canadiennes

Ottawa, Canada
K1A 0N4

The author has granted an irrevocable non-exclusive licence allowing the National Library of Canada to reproduce, loan, distribute or sell copies of his/her thesis by any means and in any form or format, making this thesis available to interested persons.

The author retains ownership of the copyright in his/her thesis. Neither the thesis nor substantial extracts from it may be printed or otherwise reproduced without his/her permission.

L'auteur a accordé une licence irrévocable et non exclusive permettant à la Bibliothèque nationale du Canada de reproduire, prêter, distribuer ou vendre des copies de sa thèse de quelque manière et sous quelque forme que ce soit pour mettre des exemplaires de cette thèse à la disposition des personnes intéressées.

L'auteur conserve la propriété du droit d'auteur qui protège sa thèse. Ni la thèse ni des extraits substantiels de celle-ci ne doivent être imprimés ou autrement reproduits sans son autorisation.

ISBN 0-315-75976-3

Canada

**The undersigned recommend to the Faculty of Graduate Studies and Research
acceptance of the thesis**

**" Performance of Some Deadlock Prevention Routing Algorithms for
Multicomputer Systems"**

submitted by

Rajiv Kachru, M.Sc.(Comp. Eng.)

**in partial fulfillment of the requirement for
the degree of Master of Computer Science**

Thesis Supervisor

Director, School of Computer Science

Carleton University

ABSTRACT

The recent use of large multicomputer systems has led to extensive studies on the performance of such systems. The performance of large multicomputers is dependent, in part, on effective communication between the processors of the system. Effective communication is dependent on the routing of messages in the network and the absence of deadlock. Various routing schemes have been used/proposed that allow efficient communication.

This thesis considers three routing algorithms that are suitable for multicomputer systems: Merlin's algorithm, E-routing and Wormhole routing. Simulation is used to evaluate their relative performance. The performance of the routing schemes is greatly influenced by the network topology and the communication characteristics. Two network topologies-the hypercube and the torus-are considered. The performance of routing algorithms is evaluated under three types of communication patterns: uniform communication, spatial locality and temporal locality.

Results indicate that the performance of the routing algorithms may be heavily influenced by the form of communication and the locality in communication can cause a loss in the throughput of the network at a relatively low link utilization. Wormhole routing and the e-routing, which use fixed routing, are especially susceptible to the locality in communication. It is also shown that at high link utilization algorithms that use fixed routing schemes have large average message delays.

ACKNOWLEDGEMENTS

I would like to thank Dr. Sivarama Dandamudi for his expert guidance throughout the course of this research. He was always available to discuss the various aspects of the research and kept me on the straight and narrow path which greatly contributed towards the completion of this thesis.

I would also like to thank Dr. John Oommen, Dr. Frank Fiala and Dr. Petriu for serving on my committee, and for providing valuable comments and criticism. Many thanks are extended to the faculty, staff and fellow graduate students at the School of Computer Science. They were always there to provide assistance and reassurance when I needed it. I thank Gautam Aggarwal for his encouraging words which cheered me up on many a cloudy day.

I am especially thankful for the financial support received from Carleton University and the School of Computer Science.

Finally, and most importantly, I would like to thank my parents-Moti Lal and Krishna-whose sacrifices in making this thesis possible will be known only to them. This thesis is dedicated to them.

TABLE OF CONTENTS

ACCEPTANCE SHEET	ii
ABSTRACT	iii
ACKNOWLEDGEMENTS	iv
TABLE OF CONTENTS	v
LIST OF FIGURES	viii
LIST OF TABLES	xiii
CHAPTER 1:	
INTRODUCTION	1
1.1 Thesis outline	5
CHAPTER 2:	
DEADLOCK AND INTERCONNECTION NETWORKS	6
2.1 Interconnection networks	6
2.2 Deadlock	12
2.2.1. Direct Deadlock	14
2.2.2. Indirect Deadlock	15
2.2.3. Reassembly Deadlock	18
2.3. Summary	20
CHAPTER 3:	
DEADLOCK PREVENTION ROUTING ALGORITHMS	21
3.1. Deadlock Prevention methods	23
3.1.1 Permanent Buffer Assignment	23
3.1.2 Deadlock Prevention By Congestion Control	24
3.1.3 Partitioning Buffers into two separate classes	25

3.2. Deadlock Prevention By buffer Assignment	25
3.3. DAG-Based Algorithm for the Prevention of Store-And-Forward Deadlock	30
3.4. E-routing for Deadlock Prevention	34
3.5. Wormholing to Prevent Deadlock	37
3.6. Summary	42
 CHAPTER 4:	
EXPERIMENTAL ENVIRONMENT	44
4.1. The Experimental Environment for the Hypercube	45
4.1.1 Permanent Buffer Assignment	45
4.1.2 E-routing in a Hypercube	49
4.1.3 Wormholing in a Hypercube	50
4.2. The Experimental Model for a Torus	53
4.2.1 Permanent Buffer Assignment in a Torus	53
4.2.2. E-routing in a Torus	54
4.2.3. Wormholing in a Torus	55
4.3 Summary	55
 CHAPTER 5:	
PERFORMANCE EVALUATION UNDER UNIFORM COMMUNICATION	58
5.1. Performance Evaluation in a Hypercube	61
5.2. Performance Evaluation in a Torus	68
5.3. Summary	75
 CHAPTER 6:	
PERFORMANCE EVALUATION UNDER LOCALITY	76
6.1. Spatial Locality	76
6.1.1 Performance Evaluation in a Hypercube	77
6.1.2. Performance Evaluation in a Torus	83
6.2. Temporal Locality	89
6.2.1. Temporal Locality in a Hypercube	90
6.2.2. Temporal Locality in a Torus	96
6.3. Summary	103

CHAPTER 7:	
CONCLUSIONS AND FURTHER RESEARCH	104
7.1. Summary and Conclusions	104
7.2. Further Research	108
REFERENCES	109

LIST OF FIGURES

Figure	Page
2.1. Six Node Ring Network.....	7
2.2 64 Node Mesh.....	9
2.3. 64 Node Torus.....	10
2.4. 8 Node Hypercube.....	11
2.5. Direct Deadlock.....	14
2.6. Resolution of Direct Deadlock.....	15
2.7. Indirect Deadlock.....	16
2.8. Network of 4 Nodes with Channels and Nodes.....	16
2.9. Network of 4 Nodes with Unidirectional Channels.....	17
2.10. Numbered Virtual Channels in a 4 Node Network.....	17
2.11. Channel Dependency Graph.....	17
2.12. Deadlock due to Free Grant of Storage Buffers.....	18
2.13. Resolution of Free Buffer Grant Deadlock.....	19
2.14. Deadlock due to Preallocation.....	20
2.15. Resolution of Preallocation Deadlock.....	20
3.1. 4 Node Network.....	27
3.2. Buffer Graph for a 4 Node Network.....	28
3.3. Buffer Graph with Additional Paths.....	29
3.4. Unidirected Graph G for a Node Network.....	30
3.5. G1 Graph with Directed Edges.....	31
3.6. G2 Graph with Directed Edges.....	32
3.7. 8 Node Hypercube with Upper and Lower Channels.....	34

3.8. Cube Connected Cycle of Dimension 3.....	35
3.9. Physical Channel of a Typical Network.....	40
3.10. Two Laned Virtual Channel.....	41
3.11a. 16-Flit FIFO Buffers.....	41
3.11b. 16-Flit Four-Lane Buffers.....	41
4.1. Permanent Buffer Assignment in an 8 Node Hypercube.....	48
4.2. E-routing in an 8 Node Hypercube.....	50
4.3. Buffer Pool Partitioning using FIFO.....	51
4.4. Buffer Pool Partitioning using 2 Lanes.....	52
4.5. Buffer Pool Partitioning using 4 Lanes.....	52
5.1. Throughput in Hypercube using Wormholing under Uniform Communication and Variable Number of Virtual Channels.....	59
5.2. Throughput in a Hypercube under Uniform Communication and Variable Buffer Capacity.....	61
5.3. Average Message Delay in a Hypercube under Uniform Communication and Variable Buffer Capacity.....	62
5.4. Throughput in a Hypercube under Uniform Communication and Variable System Size.....	63
5.5. Average Message Delay in a Hypercube under Uniform Communication and Variable System Size.....	64
5.6. Throughput in a Hypercube under Uniform Communication and Variable System Size.....	65
5.7. Average Message Delay in a Hypercube under Uniform Communication and Variable Message Generation Rates.....	66
5.8. Throughput in a Hypercube under Uniform Communication and Variable Link Service Time.....	66
5.9. Average Message Delay in a Hypercube under Uniform Communication and	

Variable Link Service Times.....	67
5.10. Throughput in a Torus under Uniform Communication and Variable Buffer Capacity.....	69
5.11. Average Message Delay in a Torus under Uniform Communication and Variable Buffer Capacity.....	69
5.12. Throughput in a Torus under Uniform Communication and Variable System Size.....	71
5.13. Average Message Delay in a Torus under Uniform Communication and Variable System Size.....	71
5.14. Throughput in a Torus under Uniform Communication and Variable Message Generation Rate.....	72
5.15. Average Message Delay in a Torus under Uniform Communication and Variable Buffer Capacity.....	72
5.16. Throughput in a Torus under Uniform Communication and Variable Link Service Time.....	74
5.17. Average Message Delay in a Torus under Uniform Communication and Variable Link Service Time.....	74
6.1. Throughput in a Hypercube under Spatial Locality in Communication and Variable Buffer Capacity.....	78
6.2. Average Message Delay in a Hypercube under Spatial Locality in Communication and Variable Buffer Capacity.....	78
6.3. Throughput in a Hypercube under Spatial Locality in Communication and Variable System Size.....	79
6.4. Average Message Delay in a Hypercube under Spatial Locality in Communication and Variable System Size.....	79
6.5. Throughput in a Hypercube under Spatial Locality in Communication and Variable Message Generation Rate.....	80

6.6. Average Message Delay in a Hypercube under Spatial Locality in Communication and Variable Message Generation Rate.....	81
6.7. Throughput in a Hypercube under Spatial Locality in Communication and Variable Link Service Time.....	82
6.8. Average Message Delay in a Hypercube under Spatial Locality in Communication and Variable Link Service Time.....	82
6.9. Throughput in a Torus under Spatial Locality in Communication and Variable Buffer Capacity.....	84
6.10. Average Message Delay in a Torus under Spatial Locality in Communication and Variable Buffer Capacity.....	84
6.11. Throughput in a Torus under Spatial Locality in Communication and Variable System Size.....	85
6.12. Average Message Delay in a Torus under Spatial Locality in Communication and Variable System Size.....	85
6.13. Throughput in a Torus under Spatial Locality in Communication and Variable Message Generation Rate.....	86
6.14. Average Message Delay in a Torus under Spatial Locality in Communication and Variable Message Generation Rate.....	87
6.15. Throughput in a Torus under Spatial Locality in Communication and Variable Link Service Rate.....	87
6.16. Average Message Delay in a Torus under Spatial Locality in Communication and Variable Link Service Time.....	88
6.17. Throughput in a Hypercube under Temporal Locality in Communication and Variable Buffer Capacity.....	91
6.18. Average Message Delay in a Hypercube under Temporal Locality in Communication and Variable Buffer Capacity.....	91
6.19. Throughput in a Hypercube under Temporal Locality in Communication and	

Variable System Size.....	92
6.20. Average Message Delay in a Hypercube under Temporal Locality in Communication and Variable System Size.....	93
6.21. Throughput in a Hypercube under Temporal Locality in Communication and Variable Message Generation Rate.....	94
6.22. Average Message Delay in a Hypercube under Temporal Locality in Communication and Variable Message Generation Rate.....	94
6.23. Throughput in a Hypercube under Temporal Locality in Communication and Variable Link Service Time.....	95
6.24. Average Message Delay in a Hypercube under Temporal Locality in Communication and Variable Link Service Time.....	96
6.25. Throughput in a Torus under Temporal Locality in Communication and Variable Buffer Capacity.....	98
6.26. Average Message Delay in a Torus under Temporal Locality in Communication and Variable Buffer Capacity.....	98
6.27. Throughput in a Torus under Temporal Locality in Communication and Variable System Size.....	99
6.28. Average Message Delay in a Torus under Temporal Locality in Communication and Variable System Size.....	99
6.29. Throughput in a Torus under Temporal Locality in Communication and Variable Message Generation Rate.....	100
6.30. Average Message Delay in a Torus under Temporal Locality in Communication and Variable Message Generation Rate.....	101
6.31. Throughput in a Torus under Temporal Locality in Communication and Variable Link Service Rate.....	102
6.32. Average Message Delay in a Torus under Temporal Locality in Communication and Variable Link Service Time.....	102

LIST OF TABLES

Table	page
1. Differences between the routing algorithms.....	46

CHAPTER 1

INTRODUCTION

Computer technology has made tremendous strides in the past two decades with rapid increases in computing power at falling costs. Application areas and complexity however have grown at a much faster pace. This has led to the designing of large multicomputer systems capable of resolving complex problems with superior cost/performance options than conventional computers.

The extensive use of such multicomputers has led to exhaustive studies on the performance of large multicomputer systems. The performance of such systems is dependent, to a great extent, on effective communication between the processors (or nodes) of the system and the cost of the system on the interconnection network of the system.

The management of efficient communication between the nodes of a multicomputer system is an important research area that deals with the problem of deadlock prevention (DP) and the routing of messages within a network. The network provides the physical link with which to communicate but the physical link is not sufficient for communication. The communication has to be both dependable and efficient. Efficiency can be defined as the ratio of the network performance to the network capacity. In an ideal network the network efficiency should be as close to unity as possible but this, however, is not always the case as the throughput or performance of the network is dependent on several factors. A network can achieve ideal efficiency if the network is completely dependable and there are

no delays in the transfer of messages. A delay in the transfer of messages may occur in the buffering of messages at intermediate nodes (blocking), or due to the abortion of messages due to deadlock. These two factors contribute to a great extent to the large communication time in networks. To reduce this time several options have been used. To reduce the buffering of messages at intermediate nodes, circuit switching (as opposed to packet switching) is used in many networks. In circuit switching a connection between the source and destination nodes is created before any communication can take place. On the other hand, in packet switching a message is broken up into packets and the packets are routed to their destination. This, however, does not overcome the problem of blocking. To avoid deadlock, DP algorithms may be used. In both cases the throughput of the system increases, in comparison to their absence, but it still does not reduce the overheads of communication to a minimum. An ideal network should accept a message from the source and deliver the message to its destination without buffering and without deadlock. This, however, is not possible with the current state-of the art of data communication. Various algorithms [Dally87, Kermani79, Merlin80, Gelernter81] have been proposed to get a high throughput of the system while keeping the message delay or the time required for a message to reach from its source node to its destination to a minimum. Various networks have also been proposed that reduce the overheads connected with inter-nodal communication while utilizing the physical structure of the network to minimize the number of links traversed by a message from its source to its destination.

The basic requirements that should be fulfilled by a flexible and efficient DP algorithm are [Blazewics87]:

- 1) *Topological independence*: An algorithm should prevent deadlock for general networks i.e., for an arbitrary network topology.
- 2) *Reconfiguration independence*: An algorithm preventing deadlock should be independent of any changes in the structure of network node connections. In particular,

adding or removing a node must not imply any change in the performance of the algorithm. This is especially important because it makes the algorithm robust with respect to both node and link failures and simplifies network development.

3) *Size independence*: The size of the buffer pool required at a node should be independent of the size of the network. This is especially important in networks where the number of nodes in a network can increase, and that without bound.

4) *Efficient buffer utilization*: It should be possible to assign any free buffer to any message requiring it.

5) *No message loss*: No message should be lost as a result of the execution of a DP algorithm. A lack of this property implies that network protocols requiring retransmission of messages are necessary. In such a scenario additional overheads and the possibility of other deadlock types cannot be disregarded.

6) *Unrestricted packet routing*: The packet should be free to construct packet routes dynamically and the DP algorithm should not impose any *a priori* routing restrictions.

7) *Freedom from livelock*: No message transmission should be indefinitely suspended due to unfair routing imposed by a DP algorithm.

No known DP routing algorithm satisfies all these requirements. The essential requirements, however, are that a DP algorithm should provide freedom from livelock and that it is topologically independent. The DP algorithms that shall be discussed, in this thesis, will satisfy these essential requirements and some, if not all, of the others.

In recent years, a different approach has been taken to communication in multicomputer systems. The approach advocates the use of wormholing as opposed to a store-and-forward strategy. Such a strategy has been shown to provide a greater throughput and a lower average message delay. Wormholing is a hybrid form of circuit and

packet switching which gives the performance of circuit switching at low loads and that of packet switching at high loads.

The routing employed by the wormholing is a form that may be employed by packet switched networks, yet its throughput is greater than conventional packet switched networks. Applications on multicomputer systems, however, exhibit some form of locality. This locality may in some cases adversely affect the performance of the system. To fairly judge the performance of routing algorithms, the throughput of the algorithms should be considered under different conditions. The performance of the algorithms under three forms of traffic variations, temporal locality, spatial locality and uniform distribution should be considered. By temporal locality we mean that several nodes are repeatedly chosen for the destination of successive packets. Spatial locality implies the topological proximity of the nodes, i.e., nodes at a proximity will communicate more often than nodes further apart. Finally uniform distribution ignores physical proximity or any other form of locality and messages are equally distributed to all nodes.

The topology of the network may also affect the performance of networks. The distance between the nodes, depending upon the topology, may vary by a great margin and the performance of such networks is directly associated with the network topology. For example, 64 nodes arranged in a 8-ary 2-cube (torus) has a maximum path length of 8 while 64 nodes forming a ring have a maximum path length of 32, a length 4 times as long.

Many other factors may affect system performance like delta and hybrid weighted routing instead of shortest path routing. These factors, however, will not be considered in the thesis. The thesis will study in depth the performance of three DP algorithms on two different topologies, the torus and the hypercube. These networks have been chosen as benchmarks as several commercial multicomputers are implemented using these topologies. The three algorithms represent the changes in multicomputer design. The store-and-forward DP algorithms used in the previous models have been replaced by wormholing.

Wormholing, however, uses a DP algorithm similar to that employed by a store-and-forward DP algorithm. A comparison of wormholing should encompass the relative improvements in performance *vis-a-vis* not only store-and-forward algorithms but also with a store-and-forward routing algorithm employing the same DP techniques and routing. A comparison of wormholing with such an algorithm is also included in the thesis.

1.1 Thesis Outline

The thesis consists of seven chapters. Chapter two describes some interconnection networks and some common forms of deadlock. The interconnection networks described in detail are those belonging to the class of k-ary n-cubes. They include meshes and rings, tori and hypercubes. The k-ary n-cubes are discussed in depth as the torus and the hypercube form the base model on which the simulations are performed. There are several forms of deadlock, but many of them are not applicable to multicomputer systems. The forms of deadlock and the respective deadlocks are the ones applicable to multicomputer systems.

Chapter three looks at some DP routing algorithms and their implementation. Simple networks are used to explain their DP strategies and their implementation.

Chapter four deals with the experimental environment and the simulation model. As mentioned earlier, two base models are used in the simulation-the torus and the hypercube. The chapter explains the implementation of the algorithms on both the torus and the hypercube and also the other factors taken into account while performing the simulation.

The results of the simulation with uniform distribution (i.e. no locality in communication) are discussed in chapter five. The effect of locality (both spatial and Stemporal) on the system performance are discussed in chapter six. The thesis ends with chapter seven containing the summary and directions for further research.

CHAPTER 2

DEADLOCK AND INTERCONNECTION NETWORKS

A critical component of any multicomputer system is the underlying interconnection network. The use of interconnection networks leads to the possibility of deadlocks occurring between communicating nodes. This chapter discusses the types of interconnection networks and the common forms of deadlocks applicable to multicomputer systems. Section 2.1 reviews interconnection networks while Section 2.2 describes some common forms of deadlocks and the ways to prevent their occurrence.

2.1. Interconnection Networks

As mentioned earlier a critical component of any multicomputer system is its communication network. Many algorithms are communication-limited rather than processing limited. Many different network topologies have been proposed for use in multicomputers, and these include trees, shuffle exchange networks, binary n -cubes and others. The binary n -cube is a special case of the family of k -ary n -cubes, cubes with n dimensions and k nodes in each dimension. Most multicomputers have been built using networks that are n -cubes or are isomorphic to k -ary n -cubes. These include rings, meshes, tori and the binary n -cubes. This section is restricted to the description of k -ary n -cubes, which include rings and meshes. These topologies should serve to give an idea of the interconnection networks while the torus and hypercube are used as a base for the simulation model.

In a k -ary n -cube, n refers to the number of dimensions and k refers to the radix or number of nodes in a dimension. The radix k , the number of dimensions n and the number of nodes N are related by the equation:

$$N=k^n \quad (k \leq n \leq N, n=\log_k N) \quad (1)$$

A node in the k -ary n -cube can be identified by an n -digit base k address, $a_0, a_1, \dots, a_i, \dots, a_{n-1}$. The i th digit of the address a_i represents the nodes position in the i th dimension. Each node can forward messages to its neighbor in each dimension. The simplest example of a k -ary n -cube is a ring network.

Figure 2.1 A six-node ring network

Ring interconnection networks are formed with nodes connected in a circle with each node being connected to two of its neighbors as shown in Figure 2.1. In such a network, the communication channels are assumed to be bidirectional. The dimension of the network is unity and hence by Eq. (1) the number of nodes N in the network is equal to the radix. A message is routed from its source to its destination through successive nodes to its destination. The path is dependent on the routing algorithm and is generally the shortest path. The shortest path is defined as the route along which a message, from its source node to its destination, traverses the minimum number of communication links.

In this example, there are two distinct paths that a message may take from a source node to a destination node. For example, If a message is bound from node 1 to node 5, it may take the route $1 \rightarrow 2 \rightarrow 3 \rightarrow 4 \rightarrow 5$ or the more direct route $1 \rightarrow 6 \rightarrow 5$. Of the two routes the route $1 \rightarrow 6 \rightarrow 5$ is the shorter and a message would most likely take that path. The maximum path length is the longest route that a message may have to take from its source to its destination. In this case as the links are bidirectional the longest path is equal to $N/2$, i.e. 3, but if a large network of say 64 is taken then the maximum path length is 32. As the message delay in a multicomputer system is largely due to the number of hops that a message has to traverse, ring networks are not considered as suitable topologies for multicomputer systems.

Another form of k-ary n-cubes is the mesh. The torus and the mesh fall into the same category of k-ary n-cubes with both having the same number of nodes in each dimension. For a 64 node k-ary n-cube we use an 8-ary 2-cube as shown in Figure 2.2. The mesh according to Eq. (1) has 2 dimensions and 8 nodes in each dimension. An 8-ary 2-cube mesh is shown in Figure 2.2. The mesh and the torus differ only in their border nodes. In a torus all the nodes are uniform with each node having the four output channels. The nodes in a mesh can be divided into two categories: internal nodes and border nodes. The internal nodes, like those in the torus, have 4 communication channels. The border nodes, however, have either three or two channels.

The border nodes with two channels are those that are situated at the corners of the mesh. In Figure 2.2 the corner nodes are (0,0), (0,7), (7,0) and (7,7). The other border nodes are those with 3 channels and are not located at the corners of the mesh. Examples of such nodes are (0,1), (0,2), (0,3), etc. The links of the nodes are assumed to be bidirectional. A message is routed from its source to its destination through successive intermediate nodes. The path of a message is dependent on the routing algorithm and generally, the shortest path is chosen. The shortest path is not unique. A message from the

Figure 2.2 A 64-node mesh

node (0,0) to (1,1) may take any of the following paths: $(0,0) \rightarrow (0,1) \rightarrow (1,1)$ or the path $(0,0) \rightarrow (1,0) \rightarrow (1,1)$. Both these paths are one of the shortest paths. The path taken by the message, however, is dependent on the routing algorithm. In a mesh a message destined from the source node (0,0) to the destination node (7,7) has several shortest paths: two examples are: one from $(0,0) \rightarrow (0,1) \rightarrow (0,2) \rightarrow (0,3) \dots (0,7) \rightarrow (1,7) \dots (7,7)$ and another $(0,0) \rightarrow (1,0) \dots (7,0) \dots (7,7)$. The path length or the number of hops for such a message in a 64 node mesh is 14 while a similar message in the torus requires just 2 hops (see Figure 2.3). The primary difference in the number of hops is due mainly to the absence of border links. The maximum path length in a mesh is 14 and the maximum path length in a torus of equivalent size and dimension is 8. This marked difference in path lengths as well as the non-uniformity of nodes makes meshes unsuitable for multicomputer systems.

The equivalent 64 node torus is shown in Figure 2.3 in which again assume that the links are bidirectional. Each node is connected to four of its neighbors, and thus node (0,0) can communicate with node (0,1), (1,0), (7,0), and (0,7). A message is routed from its source to its destination through successive nodes to its destination. The path taken by a message is dependent on the routing algorithm and is generally the shortest path. For example a message destined from the source node (0,0) to the destination node (7,7) has

Figure 2.3 A 64-node torus

two shortest paths: one by going from the source (0,0) through (7,0) to (7,7) and another from (0,0) through (0,7) to (7,7). In both cases the number of hops or the number of links crossed is minimum and is equal to two. There are many other routes to get, for example, from (0,0) to (0,1)→(0,2)→(0,3)→(0,4)→(0,5)→(0,6)→(0,7)→(7,7) but they are

generally not considered as they are not the shortest paths. For our example, a message from the node (0,0) to the node (4,4) has a path length of 8 and is the maximum for this network.

Another form of the k-ary n-cube is the binary hypercube. A binary hypercube is an interconnection of $N=2^n$ nodes which may be thought of as placed at the corners of an n-dimensional cube with each side of the cube having two nodes. An example of such a structure for $N=8$ is shown in Figure 2.4. In this case, $n=3$ and the structure formed is a three dimensional cube. The address of each node is represented in base two where the address of each node from its neighbor differs in exactly one bit position. In the given

Figure 2.4 An example hypercube network with 8 nodes

example, each node is connected to three of its neighboring nodes. Thus, for example, node (0,0,0) can communicate with nodes (0,0,1), (0,1,0) and (1,0,0). The route is formed by taking the exclusive OR of the source and destination nodes. Thus, for a message from node (0,0,0) to node (1,1,1) the route is (1,1,1), i.e., the source and destination addresses differ in every dimension by one bit. The routes that can be taken are numerous and all of them are the shortest paths. To use the same example, to send a message from the node 0 to node 7, the route could be (0,0,0) to (0,0,1) $\rightarrow$ (0,1,1) $\rightarrow$ (1,1,1), or (0,0,0) $\rightarrow$ (0,1,0) $\rightarrow$ (1,1,0) $\rightarrow$ (1,1,1). In the case of this hypercube, the maximum distance traversed by a message is three. The selection of the shortest path from

a set of shortest paths is dependent on the routing algorithm, which may place restrictions on the course of a message to prevent such incidents as deadlock. For a network of 64 nodes the corresponding hypercube would have 6 dimensions and hence a maximum path length of 6. This is in comparison to a 64-node torus which has a maximum path length of 8. If the average number of hops is taken as half the maximum path length (as is often the case in uniformly distributed communication) then a packet on a torus has, on average, to traverse four nodes to reach its destination while a packet on a hypercube has to traverse only three nodes: a difference of 33 $\frac{1}{3}$ %. It is for this reason that hypercubes are often preferred over tori for building multicomputer systems.

2.2. Deadlock

One of the major problems that has to be solved in the design of interconnection networks is deadlocks. A deadlock occurs if the transmission of messages is halted indefinitely due to the unavailability of buffers. This is the result of a circular wait for buffers. The messages in the network form a cycle and can only proceed towards their destination if a message releases one or more of its buffers. To ensure dependable communication between nodes in a network, a message must be guaranteed to reach its destination in finite time. To solve the problem of deadlocks in interconnection networks one of two options may be taken. Messages may be routed along any path in the network, preferably along one of the shortest paths, and if a deadlock occurs then a deadlock detection and resolution algorithm is invoked. Alternatively deadlock free routing algorithm may be used in the routing of messages which guarantees deadlock free communication. Two forms of deadlock free routing algorithms may be used to provide deadlock free routing: deadlock avoidance routing algorithms and deadlock prevention routing algorithms. Deadlock avoidance routing algorithms allow the forwarding of messages along any route as long as the transmission of this message does not lead to deadlock. Deadlock free routing algorithms forward a message along a predefined path and are hence

stricter in their routing policies. Deadlock avoidance routing algorithms are more flexible in their message routing, but have excessive overheads such as knowledge of the global state to predict the possibility of deadlock. As this is the case most multicomputer systems implement deadlock free routing on deadlock prevent algorithms rather than on deadlock avoidance schemes. The use of deadlock resolution algorithms require the abortion and retransmission of messages while the deadlock prevention routing requires messages to follow a predefined path. This predefined path limits the optional paths a message may use and may lead to an increase in the transmission time of a message. For example in Figure 2.4 a message from the node (0,0,1) to the node (1,1,0) requires three hops to reach its destination as the source and destination nodes differ in all three dimensions. The message may follow any one of the following paths:

$$(0,0,1) \rightarrow (0,0,0) \rightarrow (1,0,0) \rightarrow (1,1,0)$$

$$(0,0,1) \rightarrow (0,0,0) \rightarrow (0,1,0) \rightarrow (1,1,0)$$

$$(0,0,1) \rightarrow (1,0,1) \rightarrow (1,0,0) \rightarrow (1,1,0)$$

$$(0,0,1) \rightarrow (1,0,1) \rightarrow (1,1,1) \rightarrow (1,1,0)$$

$$(0,0,1) \rightarrow (0,1,1) \rightarrow (0,1,0) \rightarrow (1,1,0)$$

$$(0,0,1) \rightarrow (0,1,1) \rightarrow (1,1,1) \rightarrow (1,1,0)$$

All these paths are the shortest paths as they require only three hops to reach the destination node. But if a message is allowed to take any of these paths, the possibility that a deadlock can occur is present. To prevent deadlocks the e-cube (edge-cube) routing algorithm [Dally87] may be used. The e-cube routing algorithm routes messages according to its path in order of decreasing dimensions. In our example, the only path available to the message is $(0,0,1) \rightarrow (1,0,1) \rightarrow (1,1,1) \rightarrow (1,1,0)$. If the buffers of the node (1,0,1) are occupied then the message will be blocked at node (0,0,1) even though the buffers on the

node (0,0,0) or (0,1,1) are unoccupied. This leads to an increase in the message delay as only a fraction of the possible communication paths are utilized. A deadlock detection and resolution algorithm would allow any of these paths to be used but as most multicomputer systems have simple communication techniques, deadlock prevention algorithms are preferred over deadlock detection and resolution algorithms. The following sections will discuss some common forms of deadlock and the methods used for resolving such deadlocks.

2.2.1. Direct Deadlock

The simplest form of deadlock that can occur in communication networks is direct deadlock. This form of deadlock occurs when two adjacent nodes of a packet-switched network are both filled with packets for transmission to the other node. These two adjacent nodes form a cycle and hence a deadlock occurs. This is shown in Figure 2.5.

Figure 2.5 Direct deadlock

This form of deadlock can be resolved by partitioning the buffer pool of all nodes in two: one for the input packets and one for the output packets, or by allocating a separate buffer for each input channel. To avoid copying from an input to an output buffer a logical mechanism can be used, thus decreasing the message transmission delay. A form of resolving this type of deadlock, with separate input and output buffers, is shown in Figure 2.6.

Figure 2.6 Resolution of direct deadlock

2.2.2. Indirect Deadlock

A more complicated form of deadlock is formed when the nodes that form a deadlock are not directly communicating with each other. This form of deadlock occurs when more than two nodes are involved in a cycle. Indirect deadlock may occur even when the buffers have been divided into two separate pools (to avoid the direct deadlock) for input and output packets. This form of deadlock occurs when a cycle is formed in a network with each node awaiting the release of a buffer in an adjacent node. An example of this form of deadlock is shown in Figure 2.7.

As shown in Figure 2.7 the input and output buffers have been partitioned but to forward a message, a packet has to be transferred from an input to an output buffer and if the output buffer is full then a chain of blocked messages is formed. If this chain forms a loop then a deadlock occurs. To resolve this form of deadlock more complicated solutions are required. One of these forms may be the use of upward and downward buffers [Dally87]. An example of this is shown in Figures 2.8 and 2.9.

Figure 2.7 An example of an indirect deadlock

Figure 2.8 Network of 4 nodes with channels (C) and nodes (N)

The network consists of nodes (N) and physical channels (C). For simplicity the channels are assumed to be unidirectional. The physical channel may be split into two virtual channels, a higher channel and a lower channel, as shown in Figure 2.10. Messages from a node with a lower number than their destination node are routed via the higher channel and those from a node numbered higher than their destination node are routed via the lower channel. For the given example the routing of messages will be forwarded by the dependency graph shown in Figure 2.11. The ordering of virtual channels is $C13 > C12$

Figure 2.9 Network of 4 nodes with their unidirectional channels

Figure 2.10 Numbered virtual channels in a 4-node network

Figure 2.11 Channel dependency graph

C11> C10> C03> C02> C01. The channel C00 is not used. In this form of routing no cycles can be formed and hence the routing is deadlock free. This is shown in Figure 2.11.

For example if a message is to be sent from node N3 to node N0 then it follows the route C03→ C02→ C01. The route from N0 to N3 is just C10. To reduce the number of hops a bidirectional link is generally used. In that case there would be two additional channels in the opposite direction from those shown and would, for example, reduce to one the number of hops from N3 to N0.

2.2.3. Reassembly Deadlock

Reassembly deadlocks may occur when large messages are broken down into packets at the source node, transmitted individually and then are reassembled at the destination node. There are two main types of such deadlocks [Gunter81]. In the first type, a deadlock may occur if the operating system freely grants storage buffers. In this case the entire buffer pool may be filled with incomplete reassembly messages. This is shown in Figure 2.12.

Figure 2.12 Deadlock due to free grant of storage buffers

Each node in Figure 2.12 has 6 buffers. All the messages are multipacket messages destined for node 2. Node 2 also contains 6 buffers and it contains packets of the multipacket messages. As node 2 has no free buffers it cannot receive any more packets from node 1 and as the packets in node 2 are all multipacket messages hence none of the messages can be reassembled at the node. As this is the case no buffers will become free in finite time and a deadlock situation occurs.

A simple solution to such a deadlock is shown in Figure 2.13. In this strategy a preallocation policy is used. If a packet is a multipacket message then the number of packets required for the message are preallocated and may not be allocated to any other messages. This shown in Figure 2.13.

Figure 2.13 Resolution of free buffer grant deadlock

Figure 2.14 Deadlock due to preallocation

In this case the operating system preallocates 3 buffers for both messages a and b. If the messages c or d request a buffer then their requests will be denied as all the buffers are full (3 buffers each are preallocated to a and b). This allows messages to be reassembled at the nodes and hence prevent deadlock.

The other form of reassembly deadlock is when the reassembly of messages cannot take place due to preallocation. This is shown in Figure 2.14. In node 3 three buffers each have been allocated to messages a and b. Messages from node 2 to be forwarded are blocked and a deadlock occurs as messages from node 2 cannot advance to node 3 and the reassembly of messages a and b cannot take place as the last packets of a and b are blocked

in node 1. A simple solution to this problem may be the reservation of an input buffer for every channel. This allows the node to accept packets even when all the buffers are full. If a packet, to be forwarded to another node, is received, then it is forwarded towards its destination, but if the packet received is the first packet of a multipacket message then it is discarded and a request for reassembly buffers is registered in the queue. When sufficient space for reassembly becomes available then a request for retransmission is returned. This is shown in Figure 2.15.

Figure 2.15 Resolution of preallocation deadlock

i and j are packets to be forwarded towards their destinations. The input buffer allows the packets to be forwarded towards their destination even though there are no free buffers at the node. This frees the buffers for the remaining packets of a and b to be forwarded towards node 3 where they can be reassembled.

2.3. Summary

In this chapter the basic types of interconnection networks used in multicomputer systems were reviewed. The torus, mesh and binary hypercubes, with their topological properties, were presented and the reasons for their use in multicomputer systems was discussed. Section 2.2 described the various forms of deadlock common found in multicomputer systems. The reason for their occurrence as well as the ways of preventing such deadlocks was also reviewed.

CHAPTER 3

DEADLOCK PREVENTION ROUTING ALGORITHMS

In most cases the occurrence of deadlock in a network has an undesirable impact on the network users. When deadlocks occur, they are treated in an ad-hoc fashion and since changes in existing implementation is costly, it is preferable to have a deadlock prevention scheme rather than a detection and recovery procedure. The primary aim of a DP procedure is that it should have a minimal effect on network performance. Deadlock prevention algorithms use different data communication techniques. One of these techniques is circuit-switching. Circuit-switching [Kermani79] was originally designed for human conversation and a complete path of communication links must be set up before the real communication begins. Furthermore, the path is tied up for the duration of the whole session between the two parties. Data transmission is, however, bursty and when applied to data communication networks, circuit-switching suffers from some drawbacks: one is the slow path set up time which delays the transfer of messages from the sender to the receiver till the whole path has been set up. If the circuit is released after each message, then the excessive signaling delay especially for short messages is disadvantageous. Another is the low channel utilization due to the fact that the paths are tied up for the duration of the communication but are not being used during idle periods. As circuit-switching suffers from such drawbacks it is generally not used in data communication networks.

To achieve a better channel utilization, it may be feasible to free channels on a path when no communication is taking place. In such a case a message switching technique may

be used. In message switching, messages are routed towards their destination without establishing a path beforehand. Storage buffers at each node allow the forwarding of the message from node to node towards its destination. The forwarding of the message is performed by a routing algorithm (assumed deadlock free) and the destination of the message is attached to the header of the message. For a source-destination pair of nodes communication links are not allocated to complete paths, but they are shared, and this leads to greater channel utilization.

A further extension of the message switching technique is the division of messages into equal length packets, called packet switching. Each packet contains its own addressing information (destination). Each packet is routed separately from the source to its destination. When all the packets of a message reach the destination, reassembly takes place at the destination node. This does induce extra overheads but the use of packets allows the use of a number of links on a path simultaneously. This allows a higher channel utilization and a lower net delay in message transmission.

Another method for data transmissions is a hybrid form of circuit and packet switching. This method divides a message into equal size packets (as in packet switching) but instead of buffering it at each node (as in the store-and-forward networks), the packets can be transmitted out by the node if the necessary output channel is free. When an intermediate node receives a packet it looks at the routing information contained in the header of the packet. If the required output channel is free then the packet is transmitted further towards its destination. In this strategy, a packet need only be buffered in the case of an output channel being busy, and the delay due to unnecessary buffering is avoided. This hybrid version of both packet switching and circuit switching overcomes the disadvantages of both these switching methods as during heavy loads the system behaves like a packet switched network while during low loads it behaves like a circuit switched network.

This chapter is organized as follows. Section 3.1 describes some general methods employed to prevent deadlock while Sections 3.2-3.5 present some DP algorithms. The DP algorithms describe the methods used to prevent deadlocks in data communication networks and are simple variations of the DP methods described in Section 3.1. The algorithms presented have been chosen because of their simplicity and also to explain the use of more than one method of DP in a given algorithm. The latter may be essential for the avoidance of deadlock in some cases, and may be used to increase the throughput of the network in others. Section 3.6 is a summary of the chapter.

3.1. Deadlock Prevention Methods

The essential difference between message and packet switched systems is that in a packet switched networks a message may consist of multiple packets. Measurements in networks, (particularly in the ARPANET in which the value is 1.1 [Kleinrock74]), show that the average length of a packet is closer to unity. Therefore it will be assumed that messages consist of a single packet. In such a case the only forms of deadlock that will be encountered are the direct and indirect deadlocks. As the prevention of indirect deadlock does not allow direct deadlock to take place, the three main strategies for deadlock resolution are valid for both forms of deadlock.

The following sections explain the three main DP methods used in multicomputer systems. The principle of these methods is first described and the implementation of these methods on some given algorithms is explained in depth in the later parts of this chapter.

3.1.1. Permanent Buffer Assignment

The use of the method of permanent buffer assignment to prevent deadlock is based on the idea that a buffer can communicate with only a given set of buffers. These buffers do not form a cycle, and hence any messages being transmitted via these buffers are guaranteed to be deadlock free.

This method reserves a certain number of buffers to every virtual line passing through a given node. In such a case the buffers are permanently assigned to virtual lines and communicate with only the assigned buffers in the neighboring nodes. Permanent buffer assignment is acceptable only if the cost of buffer storage is not a criteria for the network or if the usage of the buffers assigned to a virtual line is very high. This, however, is very rare and dynamic buffer allocation schemes may be used to guarantee a high buffer usage rate. An example of this strategy is shown in Section 5.2. In this case the arcs of the buffers represent the virtual lines and deadlock is prevented by the permanent assignment of buffers to these arcs. The usage of logical buffers also insures a high buffer usage rate.

3.1.2. Deadlock Prevention by Congestion Control

The use of permanent buffer assignments leads to the reservation of buffers to a certain class of messages. This, in effect, means that though a buffer may be empty it may not be available to another message, that belongs to a different class. A better way of utilizing buffers would be to allow any message to occupy any buffer. This is allowed in the method of DP by congestion control. There are two main ways of preventing deadlock by congestion control:

- (a) *Congestion control by alternative routing:* This uses the idea of rerouting packets through alternative routes to provide a more uniform distribution of traffic throughout the network. The packets are directed away from the overloaded regions on the basis of the current traffic. As long as there are still regions which are less loaded, this method may prevent deadlock. However, additional methods of congestion control (i.e. discarding packets) may be required as local congestion may spread throughout the network.
- (b) *Congestion control by discarding packets:* In this method packets are discarded (immediately or after some time-out), if no buffers are available. This leads to a loss

in the throughput of the network and also requires the retransmission of packets. This form of congestion control is, however, required as the flooding of the network with packets may result in deadlock if packets are not discarded.

Both methods of congestion control require that a certain amount of excess buffer storage is maintained at all times. This may not be desirable, but is essential to prevent deadlock, much like the assignment of buffers to virtual lines.

3.1.3. Partitioning Buffers into Two Separate Classes

The first two methods of DP either restrict the availability of buffers to messages, or discard the messages on their way to their destination. This leads to an excessive delay in the transmission of messages. The method of two separate buffer classes restricts the routing of packets into two separate classes. A packet is allowed to flow upwards or downwards, (see Section. 2.2.2), in the hierarchy of nodes, but never upwards again after a downward move. The flow of packets is thus unidirectional. This prevents buffer deadlocks from occurring as no circular chain of packets in which the scenario where each buffer awaits for the next packet in the chain to release its current buffer is encountered.

3.2. Deadlock Prevention by Buffer Assignment

One of the methods to prevent deadlock, which uses the concept of guaranteed paths, is presented by Merlin and Schweitzer [Merlin80a]. The method proposed is similar to that of permanent buffer assignments (see Section. 3.1.1). In this case the arcs represent the virtual channels and they connect the buffers, which are allowed to communicate with only a fixed set of buffers in neighboring nodes. Several assumptions are made in this method:

- 1) All nodes, communication channels and destination nodes are operational.
- 2) No messages are lost.

- 3) Messages are sent in complete units, i.e. multipacket messages are not allowed.
- 4) No protocol prevents the destination node from turning over messages to the user as soon as they arrive.
- 5) Any topology with a finite number of nodes may be used and any routing algorithm is permitted, which allows a bounded number of hops to a node.

The deadlock prevention method affects only the transfer of messages between the adjacent nodes in the network. For a given network, it is assumed that there exist a certain number of nodes, communication channels between them, a set of buffers for the communication channel and a set of message routes permitted by the algorithm. A route consists of a path from the source node to the destination node via a set of nodes and communication channels. Each message is assumed to fit in a single buffer.

To obtain a deadlock free routing scheme a buffer graph (BG) is constructed for the given network. A node is represented as a subset of buffers and directed arcs represent the flow of messages from one buffer to another. Arcs are only permitted between buffers in the same node or between buffers in different nodes connected by a communication channels. The BG requires two properties:

- 1) There is at least one path for each route in the network.
- 2) The BG contains no directed loops.

For a message to reach its destination, it is placed in the source node at a buffer which has a guaranteed path to the destination node. The message travels from its source buffer to the next buffer in its guaranteed path and hence all messages in the network wait for the next buffer in their paths. As the BG is loop free, waiting loops cannot be formed and deadlock can never occur. An example of this is shown in Figures 3.1 and 3.2.

Figure 3.1 A 4-node network

Figure 3.1 represents a network consisting of four nodes. Each node is connected to two of its neighbors by a communication channel. Each node is assumed to have a set of four buffers. The BG for this network is shown in the Figure 3.2. All the buffers have been numbered and any route that does not visit a node more than once is permitted. All the possible routes of the network are contained in the BG, and as no directed loops are formed the BG is deadlock free. It can be seen from Figure 3.2 that three buffers are sufficient to allow all the routes of the network to be portrayed, and the rest of the buffers have not been connected up. In such a case the buffers can be connected up in any way as long as this is permitted by the physical link in the network and no loops are formed in the BG. If a message has to go from B to C via A and D then it will be placed in buffer 2 and will go through the buffers in the order $2 \rightarrow 3 \rightarrow 6 \rightarrow 9$ and the reverse equivalent of this would be the route $5 \rightarrow 6 \rightarrow 7 \rightarrow 8$. If the shortest route is to be considered then the message may take the direct route from B to C, $4 \rightarrow 5$ and its reverse $1 \rightarrow 2$ or $9 \rightarrow 12$. This shows that a network may have several guaranteed paths and so if any of the buffers is full an alternate path may be chosen.

The use of a BG restricts the set of buffers that a message can use at a node and at certain times there may be empty buffers at a node with a guaranteed path to the destination node which cannot be used by messages at adjacent nodes. This restriction is used for DP, but it may be possible to have additional arcs while maintaining the absence of deadlock. Arcs may be added to the BG provided the loop-free property of the BG is maintained and

29

6→ 7→ 8. If, when the message is in buffer 6, the buffer 7 is full and buffer 3 is empty, then the message may be sent to buffer 3 as it also has a guaranteed path to the destination (3→ 4). The use of path switching decreases the waiting time for a buffer and hence the delay time of a message to arrive at its destination.

3.3. DAG-Based Algorithm for the Prevention of Store and Forward Deadlock

The DAG (directed acyclic graph) based algorithm [Gelernter81] provides a different way to prevent store-and-forward deadlock. The DP by permanent buffer assignment requires that (see Section 3.2) :

- (a) Every permissible packet route must correspond to some path in a directed graph, which in turn depends on the completeness of the buffer graph
- (b) Buffer pools are partitioned on the basis of their position in the acyclic graph.
- (c) The size of the buffer pool may be dependent on the size of the network.

Figure 3.4 Undirected graph G for a 4-node network

These undesirable properties lead to an increase in message delay and in a decrease in the throughput of the network. The DAG-based algorithm provides a better algorithm in the sense that these restrictions do not apply.

The DAG-based algorithm requires the construction of two directed graphs to represent any network. Figure 3.4 represents the undirected graph G of a network of 4

nodes. The nodes in the graph correspond to the nodes of the network and the edges in the graph correspond to the bidirectional communication links. One of the directed graphs has the following properties:

- a) The nodes and directed edges of $G1$ represent the nodes and undirected edges of G .
- b) $G1$ is acyclic and has one node with no incoming edges (the root node).

To make a directed graph $G1$ from an undirected graph G , the following rules are followed:

- 1) Construct a spanning tree T of G with the root node r . All nodes are reachable from r via edges in T .

Figure 3.5 $G1$ graph with directed edges

- 2) Assign directions to all edges in G that are not part of T such that these edges along with those in T keep T acyclic. To accomplish this, edges between nodes at different distances from the root may be pointed away from the root, edges at the same distance from the root may be assigned either direction provided that all the edges at any given distance from the root node remain acyclic (Figure 3.5). A in Figure 3.5 represents the root node.

The other directed graph is called the $G2$ graph and in it the direction of all the edges are the inverse of those in $G1$ (Figure 3.6). Every hop that a packet makes in N

corresponds to an edge in either G1 or G2. A G1 packet is a packet whose next hop will be over a G1 edge, while a G2 packet will be a packet whose next hop will be over a G2 edge. If a packet waits on more than one output queue, then as long as it waits for at least one G2 edge it will be called a G2 packet.

Figure 3.6 G2 Graph with directed edges

The root node r has no incoming G1 edges (A in Figure 3.5) and all packets are G1 packets as no G2 edges leave r . The node without incoming edges in a G2 graph (there may be many), are called G2 roots (D in Figure 3.6). All packets in G2 roots are G2 packets because no G1 edges leave G2 roots.

For the deadlock-free property the following requirement must be satisfied:

Every node must at all times have either a G2 packet or an empty buffer. This leads to the following property:

- 1) If a non-root node finds that all its buffers are full and it contains only G1 packets, then a source packet must be redirected and rerouted over a G2 edge, (i.e., some packet is made into a G2 packet).
- 2) The root r , which contains no G2 packets, must always have an empty buffer. Any packet waiting for admission to r must be admitted to r within a finite time even if this means erasing some packet in r . This criteria of erasing packets in r can be

made rare by increasing the size of the buffer pool in r with respect to those in the rest of the network.

Forced redirection in 1 (above) occurs only when all the packets are G1 packets and act as a congestion control, thus redirecting packets from a set of congested queues to some empty queue, which is desirable. The rerouting may however be avoided if implicit pool partitioning is applied to the last buffer in the pool. In such a case, the last buffer may be refused to a G1 packet arriving over a G1 edge. These packets are called refusable packets.

Congestion control can also be implemented by having the buffer pool size of the root node the same as those of the other nodes. When traffic is such as to make deadlock possible then the excess packets are drawn towards the root node where they are eliminated.

The DAG-based algorithm has several advantages over the algorithm of permanent buffer assignment. These are:

- 1) There are no *a priori* routing restrictions.
- 2) The buffer pool size is independent of the network size, though DP requires at least 2 buffers on non-root nodes and at least one buffer on (G1 or G2) root nodes.
- 3) Buffer pools are not partitioned. Any packet may be shipped to any buffer.

This algorithm does have some drawbacks:

- 1) Packets buffered in full buffer pools on non-root nodes may be rerouted leading to an increase in the length of the path from the source and destination nodes and hence leading to a greater message delay.
- 2) Packets buffered in full buffer pools on root nodes may be deleted leading to the need for retransmission of packets and a loss of throughput of the network.

3.4. E-routing for Deadlock Prevention

The use of permanent buffer assignment and congestion control methods leads to excessive overheads with the under utilization of buffers in the former and the deletion of packets in the latter. A better method would allow buffer allocation to any packet and would not require the retransmission of aborted messages. The e-cube routing algorithm guarantees deadlock-free message routing in binary n-cubes in order of decreasing dimensions and does not require the deletion of messages nor does it require the assignment of certain buffers to a class of packets. In this form of routing the address of a node, in an n-cube, is denoted by an n-digit binary number. Each node in the network has n output channels (one for each dimension), $c_{0k}, \dots, c_{ik}, \dots, c_{(n-1)k}$. A message arriving at a node n_k destined for the node n_l is routed along channel c_{ik} where i is the position of the most significant bit in which k and l differ. Since messages are routed in order of decreasing dimensions, (and decreasing channel subscript), there are no cycles formed in the channel dependency graph and the routing is deadlock-free. In Figure 3.7, the path of a message from the node (0,0,0) to the node (1,1,1) is (1,1,1). This means that the source and the destination node addresses differ by one in all the three dimensions. The routing of the message in this case would be from (0,0,0) via the channel c_{0002} to (1,0,0), via channel c_{1001} (1,1,0) and via channel c_{1100} to node (1,1,1).

Figure 3.7 An 8-node hypercube with upper and lower channels

Figure 3.8 Cube connected cycle of dimension 3

The e-routing is not able to prevent deadlock in all k -ary n -cubes: cubes with dimension n and k nodes in each dimension. This algorithm can be modified to handle all k -ary n -cubes by using the channel dependency graph as explained in Section 2.2. An example of how routing would take place in such a network is explained using the cube connected cycle (CCC) of dimension three (see Figure 3.8) [Dally87].

Each node in the CCC has its own position in the cycle (the last binary bits of its address) and its position in the cube (the first number in the address). In Figure 3.8 node 2 in cycle 100 is N2100. There are two channels out of each node, an in-cycle channel node and an out-cycle channel. The in-cycle channel is split into three sets of virtual channels. One set of virtual channels is used to rotate a message in its cycle to get to the most

significant node in the cycle. These channels are labeled C_{2d0c} where d is the dimension of the node and c is the cycle address. The next set of virtual channels is used to decrement the dimension during the routing of the message between cycles. These channels are labeled C_{1d0c} . The third set of virtual channels is used to rotate the message around the cycle to its destination once it is in its destination cycle. These channels are labeled C_{0d0c} . The out-of-cycle channels labeled C_{1d1c} are used to toggle the bit of the current address corresponding to the dimension d . These channels, unlike the former, are physical channels.

The rotation takes place in three phases. During the first phase the message is routed around the cycle using the first set of virtual channels until it reaches a node with a dimension greater than or equal to the position of the most significant bit in which the destination address differs from the current cycle address. In phase two the message is routed from cycle to cycle and within each cycle the message is routed to the node in which the most significant dimension i in which the current cycle address and the destination cycle address differ. The message is then routed out of the cycle through this node. This is carried out until the message reaches its destination cycle. In the destination cycle the message is rotated around the cycle using the third set of virtual channels to reach its destination node. The routing is always done in the order of decreasing channel index and so is guaranteed to be deadlock free.

In the figure, if the message, for example, is to be transferred from node N0011 to node N2100 then the route that the message would take is the following:

- 1) From node N0011 to node N2100 the message uses the first set of virtual channels (C_{2d0c}) to get to node N2011.
- 2) From N2011 the out-of-cycle channel (C_{1d1c}) is used to transfer the message to node N2111.

- 3) At this cycle the second set of virtual channels are used to rotate the message to node N1111.
- 4) The out-of-channel cycle is used to transfer the message to the node N1101.
- 5) As in the previous cycle the second set of virtual channels are again used to rotate the message to node N0101.
- 6) From there the out-of-cycle channel transfers the message to node N0100 in the destination cycle.
- 7) In the destination cycle the third set of virtual channels is used to transfer the message to its destination node, in this case node N2100.

3.5. Wormholing to Prevent Deadlock

Circuit switching when applied to data communication networks leads to an excessive communication delay and is therefore not a valid option for data communication networks (see Section 3.1) . A hybrid version of circuit switching, however, may decrease the communication delay and lead to a viable alternative to store-and-forward networks. The example given in this section uses the basic idea of circuit switched networks.

For the prevention of deadlocks in interconnection networks various types of algorithms might be used that do not use store-and-forward routing [Dally87]. In store-and-forward networks, a packet is received at an intermediate node, buffered, and sent towards its destination when the required output channel is free. The time spent in buffering a packet at a node increases the message delay and is a disadvantage of the store-and-forward networks. A better way of communication might be to allow packets to be sent directly from the input channel to the required output channel of an intermediate node if the output channel is free. This is the idea behind wormholing [Dally87]. In wormholing a message is blocked at an intermediate node only if the required output channel is busy.

Only a few bits (or flits) are buffered at each node. The node examines the header flits of a message and selects the next channel on the route and begins forwarding the message towards its destination. In this way the message becomes spread out across between the source and its destination. When the header flit of a message is blocked, then all the flits of a message stop advancing and block the progress of any other message requiring the channels they occupy.

In wormhole routing virtual channels are used. The use of virtual channels restricts the routing of messages and not the assignment of buffers to messages, because the flits of a message cannot be interleaved with the flits of another message. The destination of a message is given only in the header of the flit and once the header flit of the message is accepted and assigned a buffer the remaining flits of a message must be accepted before the flits of another message is accepted. In store-and-forward (packet-switched) networks the routing information is present in all the packets and so packets of a message can be interleaved with packets of other messages, with reassembly taking place at the destination node. The prevention of interleaving messages is a major drawback of the wormhole routing as it prevents other messages from using a channel when a blocked message is present at a node, but this is essential as the routing of single packets using wormhole routing would make the algorithm unfeasible for networks.

The construction of a deadlock-free algorithm for arbitrary networks using wormhole routing requires the construction of a channel dependency graph. This is based on the concept of virtual channels, which are a group of channels that share the same physical link but in which each virtual channel has its own virtual queue.

Some assumptions are made for the construction of a channel dependency graph:

- 1) A message arriving at its destination node will eventually be consumed.
- 2) A node may generate messages for any node.

- 3) The route taken by a message is determined by its destination and not by any other traffic in the network (nonadaptive).
- 4) A node can generate messages of arbitrary length. Packets will generally be longer than a single flit.
- 5) If a queue accepts the first flit of a message then it must accept the remainder of the message before it can accept any flits from any other message.
- 6) An empty queue may arbitrate between any messages that request the queue space but it may not choose among waiting messages.
- 7) Nodes can produce messages at any rate subject to the constraint of queue space.

Wormhole routing has several advantages over store-and-forward algorithms in interconnection networks. During light loads the wormholing is analogous to message switching, while at heavy loads it compares to that of store-and-forward networks. This hybrid form of message switching allows the network to adapt to the load and leads to better throughput. Wormholing has several disadvantages. The most severe of these is that, unlike in store-and-forward networks, it does not allow other messages to pass when it blocks.

In wormholing deadlock is prevented by using virtual channels. The use of just two virtual channels (one upwards and one downwards) limits the number of messages that can be sent simultaneously, and the blocking of a message leads to the blocking of the channel, which leads to a drop in the throughput of the network. A way to avoid this might be the use of several channels. Dally [Dally90b] presents a way in which this may be achieved. The DP algorithm is the same as that in wormholing but the use of a number of virtual channels allows messages to use a channel even when a blocked message is present at the node. The use of virtual channels is advantageous for networks, especially those using

wormholing, because, unlike store-and-forward routing, the former does not allow messages to be interleaved.

The basic concept behind multiple virtual-channels is that of a multilane highway. An example of a typical network with a physical channel is shown in Figure 3.9. In the figure it can be seen that a packet B has been blocked at node 3 even though the route it has to take is different from that of packet A. This is analogous to a one lane street where no car can pass another car. A car must follow the car in front of it and if that car stops for some reason then the car following it must also stop, irrespective of whether the road ahead of it is clear. In this case it can be seen that packet B is blocked at node 3. Packet A requires the channel 4E (east of node 4) but is blocked due to the unavailability of free buffers on its route. The packet B has to travel on a different route and should therefore not be blocked. Packet B requires the buffers on the route 3N to 4N and 4N to 5N and even though the buffers 4N to 5N are free the packet cannot progress due to the blocking of packet A. To resolve this situation a two lane virtual-channel may be used. This is shown in Figure 3.10.

Figure 3.9 physical channel of a typical network

The use of two virtual-channels allows blocked packets to be passed, somewhat like a two-laned highway where each car stays in its own lane and the blocking of one lane has no effect on the other lane. In Figure 3.10 packet A is blocked but this does not affect packet B as it uses the second lane of the channel.

Figure 3.10 two laned virtual channel

The idea of using laned virtual-channels is based on the decoupling of buffer memory from a physical channel. In a conventional network (Figure 3.11a) a physical channel is coupled to its buffers in a FIFO queue, but in a virtual-channel network (Figure 3.11b) the buffers are organized into several independent queues. The use of a simple

(a) 16-flit FIFO buffers

(b) 16-flit four-lane buffers

Figure 3.11 Division of buffer memory

FIFO queue may be admissible for conventional forms of routing (store-and-forward) but in the case of wormhole routing, because the buffers cannot be used to interleave

messages, the conventional approach leads to excessive transmission delays. By using virtual-channels the buffers and channel get decoupled. Each virtual channel is coupled to the buffer associated to its lane. In this way the buffers in any lane can be allocated independently of the buffers in any other lane. This flexibility increases the channel utilization and thus the throughput. A blocked message, even one that extends through several nodes, blocks only one lane and can be bypassed by using any one of the remaining lanes. The use of multi-lane buffers in networks has an advantage over conventional networks, but it does however lead to an increase in acknowledgment traffic, which may, in some case, be a drawback. A few bits must be added to each acknowledgment signal to identify which lane is being acknowledged and because the lane buffers are smaller than a channel FIFO (see Figure 3.9), the amortization cost of block acknowledgments over several flits is restricted.

3.6 Summary

The basic requirements for a flexible and efficient routing algorithm were explained in the introduction. The DP algorithms do not meet all requirement of an efficient DP algorithm but some of these requirements are met by all the algorithms. The differences of the DP algorithms are summarized in Table 1. The table shows that none of the algorithms fulfills all the requirements required by a flexible and efficient DP routing algorithm. The requirement that shall be imposed on the routing algorithm will be that the routing algorithm should be topologically independent and that there should be no message loss. Thus a message sent from a node will eventually reach its destination and a message will never be discarded. The restriction of no message loss is imposed as the multicomputers use simple communication techniques and the retransmission of messages can cause other forms of deadlock. In such a case the DAG-based algorithm is found to be unsuitable and hence the simulation of the DP algorithms is restricted to the remaining three algorithms.

Table 1. Differences between the routing algorithms

Requirement	Permanent buffer assignment	E-cube routing	Wormholing	DAG algorithm
Topological Independence	Yes	Yes	Yes	Yes
Reconfiguration independence	No	No	No	No
Size independence	No	Yes	Yes	Yes
Efficient Buffer utilization	No	Yes	Yes	Yes
No message loss	Yes	Yes	Yes	No
Unrestricted packet routing	Yes	No	No	Yes
Freedom from livelock	Yes	Yes	Yes	Yes

CHAPTER 4

EXPERIMENTAL ENVIRONMENT

This chapter presents the experimental environment under which the performance tests on the algorithms discussed in the previous chapter are carried out. The simulations were conducted on the three of the four routing algorithms discussed in the preceding chapter. The DAG-based algorithm is not included in the comparison as the algorithm permits the discarding of packets to prevent deadlocks. This involves retransmission of discarded packets. The retransmission of packets requires additional overheads and the possibility of other forms of deadlock cannot be disregarded. Our model, therefore, assumes the fact that all packets transmitted from the source node will reach their destination nodes, and hence the DAG-based algorithm will not be considered.

The three algorithms under consideration can be classified into two categories, those using the store-and-forward routing i.e., the permanent buffer assignment and e-routing, and those using the wormholing strategy. The use of e-routing in the simulation provides a benchmark for analyzing the performance of wormholing because wormholing uses the same DP technique as the e-routing. The only difference between the two is that the e-routing uses store-and-forward routing.

The evaluation of the system performance is carried out using simulation. The experimental models for this simulation were the torus and the hypercube networks. The simulation programs were written in C and were conducted in a UNIX environment. The

experimental model used by each of the three algorithms on the hypercube is described in Section 4.1 while that for the torus is described in Section 4.2. Several performance measures used for judging the performance of the algorithms are discussed at the end of the chapter.

4.1. The Experimental Model for the Hypercube

The binary hypercube, as described in Chapter two, consists of N nodes which may be thought of as placed at the corner of an n -dimensional cube with each side of the cube having two nodes. If N is the number of nodes then the number of nodes and the number of dimensions are related by the equation:

$$N=2^n \quad (2)$$

The nodes of a binary hypercube can be identified by an n -bit binary address $a_0, \dots, a_1, \dots, a_{n-1}$. The i th digit of the address represents the nodes position in the i th dimension. Any node can forward messages to its neighbor in each dimension. The routing of the messages by the three algorithms is discussed in detail in the following section. The example is an eight node hypercube. From Eq. (2) it can be seen that this requires a three dimensional hypercube. The connected nodes are those whose addresses differ by one in any one of the three dimensions. For example the node $(0,0,0)$ is connected to the node $(0,0,1)$, $(0,1,0)$ and $(1,0,0)$. The maximum path length, which is the longest route in the network, for this example it is three. The routing of messages in the hypercube differ for all the three algorithms. They are briefly explained in the following sub-sections.

4.1.1. Permanent Buffer Assignment

In Chapter Three the DP algorithm was shown for a simple ring network using a unidirectional link. The simulation model consists of a hypercube and the routing in such a network is decidedly different from that of the ring. The model used for the simulation is

explained below. The hypercube selected is, for simplicity, a three dimensional hypercube of 8 nodes. The maximum path length in such a network is 3 and the minimum number of buffers required to prevent deadlocks is four. The numbering of the buffers is C0, C1, C2, C3 and a message at its source node is placed in buffer C0. Each of the paths allowed by the topology of the hypercube is shown in Figure 4.1. Inside a node all the buffers are linked by the relation $C0 \rightarrow C1 \rightarrow C2 \rightarrow C3$. This is from the general understanding that the lower numbered buffer is linked to a greater numbered buffer. Additionally this allows path switching inside a node. Each node is connected to three other nodes in the network. For clarity the connection of each buffer at a node with the corresponding buffer at its neighboring node is not shown. For example, the buffer C0 at the node (0,0,0) is connected the buffers C1 at the nodes (0,0,1), (0,1,0) and ((1,0,0). The connection with the buffer C1 at the nodes (0,0,1) and (0,1,0) are shown by the line, but the link between the nodes (0,0,1) and (0,0,0) is shown by the link between the buffers C3 at (0,0,0) and the buffer C4 at the node (1,0,0). The opposite link i.e., from the node (1,0,0) to the node (0,0,0) is shown by the link form the buffer C2 at the node (1,0,0) to the buffer C4 at the node(0,0,0). In this way all the buffers are connected between nodes. A single connection represents a general form of links i.e., if two buffers are connected by the relation $C0 \rightarrow C1$ then it represents the class of links:

source		destination
C0	$\rightarrow$	C1
C1	$\rightarrow$	C2
C2	$\rightarrow$	C3
C3	$\rightarrow$	C4

Likewise a relation $C2 \rightarrow C3$ also represents the same relation. The buffers $C4$ are not connected to any other buffers. This is apparent by the fact that any packet that reaches the buffer $C4$ has reached its destination node and hence has no more hops left. This criteria also ensure that no deadlocks can take place as no cycles can be formed in the figure. The routing of the packet in such a hypercube takes place as follows: The message from a source node is placed in its buffer $C0$. The route is calculated by taking the exclusive OR of the source and destination address. In a three dimension hypercube the maximum number of hops that a message can have is three, and hence the minimum number of buffers required to prevent deadlock is four (maximum number of hops +1). The routing is simple and the packet can be routed along any dimension in which the source and destination addresses differ, but the buffer index increases by one on each hop. In this case the packet is placed in buffer $C1$ after its first hop, $C2$ after its second hop and so on until it reaches its destination. After each hop the route of the packet is recalculated by taking the exclusive OR of the current and destination nodes. The route is independent of the dimension. If, for example, the packet is being routed from node $(0,0,0)$ to node $(1,1,1)$ and buffer $C1$ on the node $(0,0,1)$ is full then the packet may be routed to the buffer $C1$ on the node $(0,1,0)$ or $(1,0,0)$, whichever is free. This allows the network to be flexible in terms of routing but does not free the packet from being permanently assigned to a particular buffer. In the example if the buffers $C1$ of all the nodes $(0,0,1)$, $(0,1,0)$ and $(1,0,0)$ are filled then the message will be blocked even if the other buffers in the nodes are free. To overcome such an obstacle the use of excess buffers proves to be useful. In the example four buffers are essential to prevent deadlock. If five buffers are used then the packet may use any of the buffers $C1$ or $C2$ may be used at any of the nodes. The criteria for when more than the minimum buffers are present at a node is that :

$$\text{hops left} + \text{buffer number} \leq \text{number of hops}$$

Figure 4.1 Permanent buffer assignment in an 8-node hypercube

A relaxation of the rule for buffer assignment is allowed once the buffer is only one hops away from its destination node, and then it may occupy any buffer in the node. This is because a packet will free the buffer at its destination node in finite time and hence

prohibit deadlock. The use of excess buffers allows a packet to wait for not just one buffer, thus increasing the probability of a packet being forwarded and not blocked. The structure of a three dimensional hypercube using permanent buffer assignment is shown in Figure 4.1.

4.1.2. E-Cube Routing in a Hypercube

The usage of permanent buffer assignment places undue restrictions on the routing of messages to certain buffers and also makes the buffer size dependent on the system configuration. If the maximum path length is three, then the minimum number of buffers required is four, but if the system configuration is changed or if the system size is changed then the minimum number of buffers required to prevent deadlock would increase. This may prove cumbersome and a DP algorithm independent of system size is desirable. The e-cube routing algorithm in hypercubes routes packets in order of decreasing dimensions and hence decreasing channel subscripts.

The figure for a three dimensional hypercube is shown in Figure 4.2. The links here are assumed to be bidirectional. The routing in this case, however, differs from that allowed by permanent buffer assignment. At the source node a route of the packet is calculated by taking the exclusive OR of the source and destination node addresses. The routing takes place in decreasing order of dimensions. For example, a packet from the source node (0,0,0) to the node (1,1,1) has a route of (1,1,1). In routing by permanent buffer assignment the packet could be routed in any of the three dimensions. The e-cube routing, however, has a stricter routing criterion. The packet must be routed along dimension three before it can be routed along dimension two, i.e., the next node along its path must be (1,0,0) and the one after that must be (1,1,0). This strict routing policy decreases the number of shortest paths that a packet may take and is a major drawback of the e-cube routing. At an intermediate node, the packet may occupy any free buffer in the node and the route is recalculated by taking the exclusive OR of the destination and current

node address. In this way the packet would go from the source to the destination using the channels $C_{0002} \rightarrow C_{1001} \rightarrow C_{1100}$. The reverse of this route i.e., the transmission of a packet from the node (1,1,1) to the node (0,0,0) would follow the route $(1,1,1) \rightarrow (0,1,1) \rightarrow (0,0,1) \rightarrow (0,0,0)$ and the channels involved in the transmission would be $C_{1112} \rightarrow C_{0111} \rightarrow C_{0010}$.

Figure 4.2 E-cube routing in an 8 node hypercube

4.1.3. Wormholing in a Hypercube

The use of the e-cube routing on the hypercube places certain restrictions on the transmission of messages, specifically the fact that messages can be routed only in order of decreasing dimensions leads to the fixed routing of messages for a source destination pair.

This does not allow packets to use other shortest paths that may be available. This may cause an undue increase in the message delay. The use of wormholing in place of store-and-forwarding may also allow a decrease in the message delay. The structure is shown in Figure 4.3. The buffer pool differs from that for the e-cube routing scheme. In wormholing, a flit is forwarded to the next node towards its destination if the output channel is free. Messages cannot be interleaved as the destination address is placed only on the first flit and if the flit is blocked at a node then all the flits behind it are also blocked. The partitioning of the buffer pool into several virtual lanes allows messages to be placed on different lanes to prevent undue blocking. The buffer pool may be partitioned from a FIFO queue to a parallel queue. The simulation allows the buffer pool to be partitioned into any of a variety of ways. The different types of variations are shown in Figures 4.3, 4.4, 4.5. For simplicity only unidirectional links are shown.

Figure 4.3 Buffer pool partitioning using FIFO

In Figure 4.3 a normal buffer pool is shown. The buffer pool acts as a FIFO queue. If a message (say message 1) is blocked at node (0,0,0) bound for node (0,1,0) then message 2 bound for node (1,0,0) will also be blocked. The use of virtual channels will avoid this situation. This is shown in Figure 4.4.

The best scenario is when the number of buffers is equal to the number of virtual lanes. In Figure 4.4 if a third message were to be forwarded from the node (0,0,1), it

would be blocked at the node (0,0,1). If, however, each buffer forms a virtual lane then the messages would be as shown in Figure 4.5. The messages are blocked on their respective

Figure 4.4 Buffer pool partitioning using 2 lanes

Figure 4.5 Buffer pool partitioning using 4 lanes

channels. For example message 1 is a two packet message, and hence it occupies two buffers both of which are on different nodes. The channels they occupy are the same. In this form the message 3 can also be accommodated. In the simulation all messages are

assumed to be single packet messages and are assumed to fit in one buffer, but even for such cases the addition of virtual lanes has a positive impact on the performance of the network.

4.2. Experimental Model For A Torus

The torus, as explained in Chapter 2, consists of N nodes in a two dimensional plane. The nodes in the torus can be identified by their address in a two dimensional plane. An example of a 64-node torus is shown in Figure 2.3. A packet in a torus is routed from its source node to its destination through successive intermediate nodes. Each node is connected to four of its neighbors, in comparison to a 64 node hypercube that has 6 links for each node. The number of shortest paths is dependent on the number of direct links that a node possesses. As the number of direct links is less than that in a hypercube, the number of shortest paths is also less. The maximum hops for a message is dependent on the maximum path length for a message. In the case of this torus, the maximum path length is 8 or about 2 hops greater than that of the hypercube of equal size. The routing of messages in the torus differs for all the routing algorithms. They are briefly explained below.

4.2.1. Permanent Buffer Assignment In A Torus

A 64-node torus is shown in Figure 2.3. The links of the torus are assumed to be bidirectional. The minimum number of buffers required to prevent deadlock in a network using permanent buffer assignment is one greater than the maximum path length. In this example network the maximum path length is 8. Therefore, the minimum number of buffers required to prevent deadlocks is 9. The routing of the packets may be in any of the four directions and the buffers required are in the same order as explained in Section 4.1.1. The routing of the packets is more complex than that in a hypercube. A packet may use any of the four links to reach its destination, but only two of the links will provide the shortest

paths. For example, a packet from the node (0,0) to the node (3,3) may take any number of routes. Some of the routes are:

$$(0,0) \rightarrow (0,1) \rightarrow (0,2) \rightarrow (0,3) \rightarrow (1,3) \rightarrow (2,3) \rightarrow (3,3)$$

$$(0,0) \rightarrow (1,0) \rightarrow (2,0) \rightarrow (3,0) \rightarrow (3,1) \rightarrow (3,2) \rightarrow (3,3)$$

$$(0,0) \rightarrow (0,1) \rightarrow (0,2) \rightarrow (1,2) \rightarrow (2,2) \rightarrow (2,3) \rightarrow (3,3)$$

$$(0,0) \rightarrow (0,1) \rightarrow (1,1) \rightarrow (1,2) \rightarrow (1,3) \rightarrow (2,3) \rightarrow (3,3)$$

There are of course many others including the non-optimal routes. The shortest routes in this case are formed when the packet moves along the directions north and east. If the packet moves along the directions south or west then it follows a non shortest path. In the simulation a packet is allowed to follow any of the shortest paths. For example to go from the node (0,0) to the node (3,3) a packet may go along to the node (1,0) or the node (0,1). If it goes to the node (0,1) then it can go to either (0,2) or (1,1). In other words, a packet may wait on more than one node. This lessens the possibility of a packet being blocked due to the unavailability of a buffer.

4.2.2. E-routing in a Torus

The routing of packets in a torus differs from that in a hypercube. In a hypercube e-routing takes place in order of decreasing dimensions but in a torus the routing of packets in order of decreasing dimensions is not enough to prevent deadlocks. In a hypercube there is only one node in a dimension while in a torus a dimension may have several nodes. If, for example, routing is performed in a torus in order of decreasing dimensions then a packet will be routed in each dimension till the destination address is equal to the current address. If a packet is routed first along the X-axis and then along the Y-axis even then deadlock is possible as the torus forms a ring in each dimension and so deadlock has to be prevented in each dimension as well. For this, upper and lower channels can be used (see

Section 3.13). For example, a packet from the node (5,1) to the node (2,2) is routed along the X-axis and then along the Y-axis, in order of decreasing dimensions. The packet is routed to the node (4,1)→(3,1)→(2,1) by the lower channel as the source's X dimension address is greater than the destination's X dimension address. At the node (2,1) however the packet is routed via the upper channel to the node (2,2). As the routing takes place in order of decreasing dimensions and on fixed channels, no deadlock can take place.

4.2.3. Wormholing in a Torus

Wormhole routing follows the same strategy of routing as the e-routing. The buffer pool, however, differs from that of the e-routing. The partitioning of the buffer pool into several virtual lanes is the same as that for the wormholing in the hypercube, the variations of which are shown in Figures 4.3, 4.4, and 4.5.

4.2.4. Summary

The simulations of the three algorithms are compared under three basic conditions: uniform distribution, or with spatial or temporal locality. Uniform distribution implies that messages are equally distributed to all nodes, i.e., a message can be sent to any node in the network with equal probability. Temporal locality means that the same node is repeatedly chosen for the destination of successive packets, while spatial locality chooses topological proximity for communication, i.e. nodes at a proximity will communicate more often than nodes farther apart.

There are several factors involved in any experiment and it is difficult to draw any useful conclusions if all the factors influence the outcome. The task of analyzing the results is much easier if some limiting assumptions are made. The following are the assumptions made about the environment in which the experiments are conducted.

- 1) Each communication link is assumed to be full duplex and is conceptually replaced by two half duplex links.
- 2) Each link is modeled as a queueing center.
- 3) The time between each successive message generation at each node is exponentially distributed with mean $1/\lambda$.
- 4) The node message generation rate of processes are independent of each other.
- 5) The message service times are exponentially distributed. Each link process messages at rate μ .

These assumptions allow each link to be modeled as an M/M/1 queueing center.

Under these conditions the performance of the various algorithms is compared. The other input parameters that are used are:

- 1) The message generation rate (λ). The message generation rate is varied for both the hypercube and the torus. For the hypercube the message generation rate is varied from 2 onwards. The message generation rate is directly responsible for the network load. A message generation rate of 2 implies that 2 messages, on average, will be generated at each node per unit time. This also gives the maximum performance measure of the network of 128 delivered messages per unit time. The message generation rate also directly influences the average message delay. The greater the message generation rate, the greater the conflict for buffer space and links, so an increase in the message generation rate implies greater average message delays.
- 2) The service rate of links (μ) plays a crucial role in the network performance. The service time of a link defines the amount of time required by a link to transmit a

packet from one node to the other. A lower service time for links implies a greater conflict for links. This leads to a greater message delay as the packets have to wait for links to become free.

- 3) The essential cause of communication deadlock is finite buffer capacity. Buffer size has a marked effect on the performance of the network. A lack of buffer space leads to the blocking of messages and hence an increase in the message delay.
- 4) The performance of wormholing is dependent on the number of virtual lanes per channel. Varying of the number of virtual lanes provides an idea of the performance of the network under different buffer configurations.
- 5) The performance of the algorithms under different network sizes is also studied. In the case of the other simulations the system size was a network of 64 nodes.

The link service rate encompasses the setup time as well as the transmission time for a message. Wormholing allows messages to be transmitted unbuffered at an intermediate node if the next link on its route is free. As this is the case, the two input variables `set_up` and `cut` are also required. the variable `set_up` defines the setup time required before transmisson can take place and `cut` defines the amount of time before the header flit may be ready for transmisson from the node if it is not being buffered at an intermediate node. These values are system dependent and for the simulations the values taken were .05 for the setup time and .000002 for the cut value.

The simulations were not calculated with confidence intervals due to time and resource constraints. The results of the algorithms are graphically shown in the next two chapters. Chapter five discusses the results obtained with uniform locality, while Chapter six presents the results obtained under spatial and temporal locality.

CHAPTER 5

PERFORMANCE EVALUATION UNDER UNIFORM COMMUNICATION

This chapter presents the results of experiments conducted to compare the performance of the three algorithms on the torus and hypercube networks under uniform communication. The performance of the algorithms is discussed under four different conditions:

- 1) Variable buffer capacity.
- 2) Variable message generation rate.
- 3) Variable link service rate.
- 4) Variable system size.

The memory in wormholing can be varied in respect to the number of buffers and the number of virtual channels. The various combinations and the best configuration is also discussed in this chapter.

The performance of the algorithms under variable buffer capacity required the algorithms to be tested under equal buffer memory. For the wormholing, this required the partitioning of the flits in different configurations of buffers and virtual channels. The results of the variations are shown in Figure 5.1. The figure shows that for optimal results

the buffers can be configured as one buffer per virtual channel. This result is justifiable under the criteria that only a packet at the head of a virtual channel may allow wormholing, i.e., the non buffering of a packet at a node. The increase in the number of virtual channels allows a greater number of cuts to take place and hence decreases the number of times that a packet has to be buffered in its route from the source to the destination node. This in turn

Figure 5.1 Throughput in a hypercube using wormholing under uniform communication with variable number of virtual-channels

leads to a low average message delay. In the comparison for wormholing the number of buffers thus represents the number of virtual channels.

Theoretically the throughput of a network in the steady state is defined by the formula:

$$T = n \cdot \lambda \quad (3)$$

Where n is the number of nodes in the network and λ is the message generation rate of each node in the network. From this formula we deduce that the throughput of a 64 node hypercube with a message generation rate of 2 should be 128, but this is seldom the case

because of buffer limitations, i.e., some messages are rejected at the source node as there is no empty buffer to accept the message. The link utilization of such a network is given by the formula:

$$\rho = l_{\text{link}} / \mu \quad (4)$$

Where ρ is the link utilization, μ the link service rate and l_{link} is the effective message arrival rate at a link. The effective message arrival rate at a link is the product of message generation rate of the network and the average distance traversed by a message from its source to its destination, divided by the number of links in the network. This is given by:

$$l_{\text{link}} = \lambda \cdot d \cdot 2^d / 2 \cdot n_{\text{link}} \quad (5)$$

Here d and n_{link} denote the dimensions and the number of links in the network. The number of links in the network n_{link} in a hypercube are defined by the formula:

$$n_{\text{link}} = d \cdot 2^d \quad (6)$$

As each channels is assumed to be bidirectional, the number of channels is twice the number of links. The effective message arrival rate is equal to:

$$l_{\text{link}} = \lambda \cdot d \cdot 2^d / 2 \cdot d \cdot 2^d = \lambda / 2 \quad (7)$$

The link utilization, from (4) and (7), is thus equal to:

$$\rho = \lambda / 2 \mu \quad (8)$$

The message delay at a link with uniform communication is given by the formula:

$$T_h = 1 / (\mu - \lambda_{\text{link}}) \quad (9)$$

The average message delay for a message is then equal to:

$$R_h = T_h \cdot d / 2 \quad (10)$$

5.1. Performance Evaluation in a Hypercube

For the performance of the algorithms with varying buffer capacity the values of the message generation rate and the links service rate are fixed at 2 and 4 respectively. The number of buffers is varied from 4 to 24. In Merlin's scheme the number of buffers required to prevent deadlock is one greater than the maximum number of hops hence its

Figure 5.2 Throughput in a hypercube under uniform communication with variable buffer capacity

performance can be measured only from 7 onwards. In the e-routing and wormholing each link is divided into two channels (an upper and a lower channel). As the links between any two nodes are assumed to be full duplex links, the buffers in e-routing and wormholing increase in groups of four, hence the results for these algorithms are only shown for these values. The throughput of Merlin's scheme reaches its peak with an increase of only a few buffers from the minimum required. Any further increase has no appreciable impact on the throughput of the network. For the wormholing, the throughput increases with the increase in the number of buffers at a node. This however is true only upto a certain value of buffers, above which the increase in the buffer size has no appreciable impact on the

throughput of the system. As shown in Figure 5.2 the performance of the e-routing algorithm trails the performance of the other two algorithms. This can be explained by the fact that e-routing uses the same store-and-forward packet switching techniques of Merlin's scheme which leads to the buffering of packets at all intermediate nodes. As the routing is also more severe than that of Merlin's scheme, the algorithm requires a larger number of buffers to reach a steady state.

Figure 5.3 Average message delay in a hypercube under uniform communication and variable buffer capacity

The average message delay for each of the routing algorithms is shown in Figure 5.3. The results were obtained with the same parameter values as those for Figure 5.2. The average message delay, in the steady state, for Merlin's scheme was found to be much greater than that for the wormholing. E-routing had the largest average message delay. The average message delay for wormholing is lower than that for Merlin's due to the fact that wormholing uses a form of circuit switching that allows packets pass intermediate nodes without buffering if the next link in its route is free. E-routing, on the other hand uses the

same routing technique as that used in the wormholing but store-and-forward packet switching to forward messages. This leads to a much greater average message delay as shown in Figure 5.3.

From the results in Figure 5.2 and 5.3 it is apparent that e-routing requires the maximum number of buffers to provide a reasonable throughput and average message delay and all further simulations in the hypercube were carried out with 24 buffers per node as these are the minimum required by the e-routing.

Figure 5.4 Throughput in a hypercube under uniform communication with variable system size

The system size can have an appreciable impact on the performance of a network. Figure 5.4 and 5.5 show the effect of system size on the performance of a network using the three given algorithm. The throughput of the system is given by Eq. (3). The system size was varied from 4 to 128 nodes and the algorithms showed no great variation in their throughput, though Merlin's scheme provided the best results while e-routing provided the worst. The throughput of the network is directly proportional to the number of nodes and

Figure 5.5 Average message delay in a hypercube under uniform communication with variable system size

as the number of nodes is increased, the throughput for all the algorithms increases proportionally. The average message delay, however, showed considerable variation for the three algorithms. An increase in the system size leads to an increase in the number of hops for a message and therefore an increase in the message delay. Wormholing, however, allows packets to be forwarded without buffering if the next link on its route is free, hence it is least affected by the increase in system size. Merlin's scheme and e-routing buffer packets at each intermediate node. An increase in the system size leads to an increase in the number of hops and hence to an increase in the average message delay. The message delay for the e-routing is, as in the previous cases, worse than that for Merlin's scheme due to the stricter routing algorithm.

A change in the message generation rate has, as shown in Figure 5.6 and 5.7, a marked effect on both the throughput and the average message delay of the network. The throughput of the network is proportional to the message generation rate. An increase in the message generation rate leads to an increase in the throughput for all the algorithms. The

increase, however, is not that marked for e-routing, as an increase in the message generation rate leads to contention for buffer space. This in turn leads to the hogging of buffers by messages at their source nodes as no buffers are available at intermediate nodes.

Figure 5.6 Throughput in a hypercube under uniform communication with variable message generation rate

In Merlin's scheme the routing allows packets to wait for buffers on a set of nodes. As this is the case and the number of buffers is much greater than the minimum required to prevent deadlock, an increase in the message generation rate does not appreciably effect the average message delay. In wormholing a message may use a number of links at the same time. An increase in the message generation rate leads to not only a contention for buffers (due to an increase in the number of messages generated at each node) but also for links (i.e., the link utilization increases) and as wormholing is very susceptible to the contention for links an increase in the message generation rate increases the average message delay. As shown in Figure 5.6 and 5.7 the performance of e-routing and wormholing deteriorates with an increase in the message generation rate.

Figure 5.7 Average message delay in a hypercube under uniform communication with variable message generation rate

Figure 5.8 Throughput in a hypercube under uniform communication with variable link service rate

Figure 5.9 Average message delay in a hypercube under uniform communication with variable link service rate

Figures 5.8 and 5.9 show the effect of the link service rate on the performance of the algorithms. A decrease in the link service rate leads to an extended buffering of messages due to contention for links. This in turn leads to an increase in the average message delay. The throughput of the network is not appreciably effected by the variation of the link service rate as the rejection rate for messages is not very high. The average message delay for wormholing and e-routing increase very rapidly with the decrease in the link service rate. Wormholing allows packets to pass unbuffered if the next link on its route is free. A decrease in the link service rate leads to a wait on links and as wormholing is very susceptible to the link service rate, the average message delay for wormholing increases rapidly with a decrease in the link service rate. E-routing has a strict routing algorithm, which allows it to wait on only one link at a time. As a decrease in the link service rate leads to a wait on links, the packets are buffered at nodes for longer periods of time and this leads to an increase in the average message delay. Merlin's scheme allows

packets to wait on not just one link but on a set of links and so a decrease in the link service rate affects Merlin's scheme the least. As seen in Figure 5.9 the average message delays of all the algorithms increase with a decrease in the link service rate but wormholing and e-routing are the most seriously effected and at low link service rates the average message delay in wormholing surpasses that of Merlin's scheme.

5.2. Performance Evaluation in a Torus

The simulations on a torus were conducted with the same workload model as for the hypercube. The message generation rate was kept at 1 instead of 2 as in the hypercube. This is because the effective message arrival rate at a link for a torus is twice that in a hypercube. To compare the algorithms under equivalent link utilization (i.e. 25%) the value of the message generation rate is taken as 1.

To study the performance sensitivity of the algorithms to the buffer capacity (or number of buffer), the values of the message generation rate and the link service rates were fixed at 1 and 4 respectively. The number of buffers was varied from 8 to 32. In Merlin's scheme the number of buffers required to prevent deadlock is 1 greater than the maximum number of hops. In a 64-node torus the maximum number of hops is 8; hence the performance of Merlin's algorithm is shown from 9 buffers onwards. The buffers in the e-routing and wormholing, due to the usage of two channels on each of the bidirectional links, increase in groups of four. The results of the three algorithm with variable number of buffers is presented in Figures 5.10 and 5.11. The throughput for Merlin's scheme is not appreciably affected by the increase in the number of buffers. It reaches its peak throughput with an increase of only a few buffers above the minimum required to prevent deadlock. The number of buffers required by e-routing, however, is much greater. The number of buffers required is greater than even in the hypercube. This can be explained by the fact that the average number of hops in a 64 node hypercube (under uniform communication) is 3, while in an equivalent torus is 4, a difference of 33.33%. E-routing requires 32 buffers for

Figure 5.10 Throughput in a torus under uniform communication with variable buffer capacity

Figure 5.11 Average message delay in a torus under uniform communication with variable buffer capacity

routing in the torus while in the hypercube only 24 are required. Note that e-routing is very susceptible to the number of hops and therefore requires a larger number of buffers in the torus. The number of buffers required by wormholing is almost the same as that in the hypercube. This can be explained by the fact that the number of hops does not appreciably effect the performance of wormholing. The performance of the three algorithms is shown in Figures 5.10 and 5.11. It is evident that as in the case of the hypercube the throughputs of all the algorithms are almost the same but the average message delay for wormholing is much better than that of the other two algorithms. This is explained by the fact that in a 64-node torus the number of links that a packet may take at a node is 4 while that in the hypercube is 6. This leads to a restriction on the routing of packets. The average number of hops for a packet is also greater in a torus than in a hypercube. These reasons combined lead to Merlin's scheme having an average message delay almost twice that of the wormholing. The average message delay for the e-routing, as in the hypercube, is much greater than that of both the methods.

For a fair comparison of the routing algorithms, the schemes should be compared under identical conditions. For this reason all the algorithms are simulated under equivalent buffer space. As the e-routing requires the maximum number of buffers, the simulations were conducted with 32 buffers at each node.

The system size has a significant effect on the performance of the network. Figures 5.12 and 5.13 present the performance of the algorithms as a function of network size. The network size was varied from 4 to 100. The throughput of the algorithms scalar with the network size as the message rejection rate is low. The average message delay, however, varies by a great margin. An increase in the network size leads to an increase in the number of hops for a message. The average message delay for the e-routing and Merlin's scheme is dependent to a great extent on the number of hops that a message has to take and so an increase in the network size leads to an increase in the average message delay for reasons

discussed earlier. Merlin's scheme performs better than the e-routing and wormholing gives the best performance

Figure 5.12 Throughput in a torus under uniform communication with variable system size

Figure 5.13 Average message delay in a torus under uniform communication with variable system size

Figure 5.14 Throughput in a torus under uniform communication with variable message generation rate

Figure 5.15 Average message delay in a torus under uniform communication with variable message generation rate

The effect of the message generation rate on the three algorithms is presented in Figures 5.14 and 5.15. An increase in the message generation rate leads to a corresponding increase in the throughput of the system. The message generation rate, however, also leads to a contention for buffers and increases the time packets are buffered at intermediate nodes due to the unavailability of buffers. An increase in the buffering of message leads to an increase in the message rejection rate and hence a loss in throughput. This can be seen with the e-routing which requires the maximum number of buffers. Its throughput drops considerably at high message generation rates and its average message delay also increases. Wormholing is very susceptible to any changes in the link utilization. An increase in the message generation rate leads to an increase in the link utilization and contention for links. This in turn leads to the buffering of messages at intermediate nodes and an increase in the average message delay. Merlin's algorithm is not so susceptible to increase in the message generation rate. As the buffer capacity of the node is much greater than the minimum required, the rejection rate of messages is not significant. As a packet waits on not just one link, but on a set of links or nodes, the message generation rate does not appreciably effect the average message delay. Figures 5.14 and 5.15 present the performance of the three algorithms. As can be seen, Merlin's algorithms performs even better than wormholing for high message generation rate.

Figures 5.16 and 5.17 show the effect of the link service rate on the performance of the algorithms. A decrease in the link service rate causes prolonged buffering of messages at nodes and hence an increase in the average message delay. At high link service rates the throughput of the network is not greatly affected by the algorithm used, but low link service rates results in a decrease in the throughput of the network. In Figure 5.16, varying of the link service rate affects the e-routing the most, for reasons discussed earlier.

Figure 5.16 Throughput in a torus under uniform communication with variable link service rate

Figure 5.17 Average message delay in a torus under uniform communication with variable link service rate

5.3. Summary

The performances of the three algorithms were compared under the uniform communication on the hypercube and torus. E-routing, due to its routing and its communication technique, displayed the worst performance. Of the two other algorithms the performance varies with the workload. Wormholing provides a better performance at low link utilization and large system sizes. At high link utilization and small system sizes, however, Merlin's scheme compared favorably (or better than) with the wormhole routing.

CHAPTER 6

PERFORMANCE EVALUATION UNDER LOCALITY

The results in this chapter compare the results obtained under uniform communication with those under some form of locality. All programs exhibit some form of locality and any routing algorithm that performs well under uniform communication may not necessarily perform well under locality in communication. Section 6.1 presents the performance of the algorithms under spatial locality and Section 6.2 discusses the performance of the algorithms under temporal locality. In both the sections the workload model was kept the same as that in the uniform communication model to give an effective comparison of the performance of these algorithms under different forms of communication.

6.1. Spatial locality

Spatial locality implies proximity of communication, i.e., nodes at a proximity will communicate more often than nodes further apart. The type of locality used is the generalized sphere of locality (G-SOL) [Dandamudi89]. The G-SOL type of distribution allows layers of locality. The total number of nodes is divided into disjoint spheres, which are then each divided into subspheres of locality. A message from a node is destined to some node within its containing layer i locality, but not within its containing layer $i-1$ locality, with a probability a_i (and to each node within this layer with equal probability). For hypercube based networks, it is convenient to assign all nodes with matching most

significant bits to the same layer of locality. For example, in the simulation model for the 64-node hypercube the 3 most significant bits denote the layer number. The use of the 3 significant bits divides the hypercube into 8 layers or clusters with 8 nodes in each cluster. To provide an even network the simulation model uses this 2^3 division, i.e., the number of nodes in a cluster and the number of clusters is equal to 2^3 . The generation of messages is also kept at an equal probability of being an intercluster or intra-cluster message, i.e., the probability of a message being an inter or intra-cluster message is 50%. An inter-cluster message may be destined to any node within the cluster with equal probability (uniform distribution) and to any cluster with equal probability if the message is an intra-cluster message. The results of the simulation using this form of locality in a hypercube and torus are discussed in Sections 6.1.1 and 6.1.2.

6.1.1. Performance Evaluation in a Hypercube

The performance of the algorithms as a function of the buffer capacity is shown in Figures 6.1 and 6.2. Figure 6.1 shows the throughput of the various algorithms when the number of buffers at each node is varied from 8 to 24. As in the case with the uniform communication, Merlin's scheme provided the best throughput and e-routing the worst. The average message delay for the three algorithms, however, was reduced in comparison to that in the uniform communication. The reduction in the average message delay could be attributed to the locality in communication, which reduces the average number of hops of a message. In comparison with the uniform distribution, the e-routing displays the greatest improvement while the improvement for wormholing was the least. This can be explained by the fact that wormholing uses a form of circuit-switching for forwarding messages and its performance is enhanced with a large number of hops between the source and destination nodes. Spatial locality in communication implies proximity in communication and reduces the average number of hops for a message. The advantage of wormholing is reduced and Merlin's scheme compares favorably with wormholing.

Figure 6.1 Throughput in a hypercube under spatial locality in communication with variable buffer capacity

Figure 6.2 Average message delay in a hypercube under spatial locality in communication and variable buffer capacity

Figure 6.3 Throughput in a hypercube under spatial locality in communication with variable system size

Figure 6.4 Average message delay in a hypercube under spatial locality in communication with variable system size

Figures 6.3 and 6.4 show the performance of the various algorithms under varying system sizes. The throughput of all the three algorithms shows very little variation and increase proportionally to the system size. The average message delays, however, inclined heavily towards the store-and-forward packet-switching algorithms. E-routing and Merlin's scheme did not display the large average message delays found in uniform communication. This can be attributed to that fact that, even though the system size increased, the number of hops did not increase proportionally as locality reduced the average number of hops between any source-destination pair of nodes, irrespective of the system size. As shown in Figures 6.3 and 6.4 the performance of the packet-switching algorithms, especially Merlin's algorithm shows considerable improvement in performance.

Figure 6.5 Throughput in a hypercube under spatial locality in communication as a function of variable message generation rate

Figure 6.5 and 6.6 show the performance of the various algorithms as a function of the message generation rate. The average message delay shows a marked difference in performance from that in the uniform communication. As communication is localized, the advantage that wormholing has over the store-and-forward routing algorithms is minimized

and the average message delay of the store-and-forward routing algorithms and wormholing does not differ by much. As the message generation rate is increased the over utilization of local links, i.e., the locality of communication leads to an increase in the average message delay. This is more apparent for wormholing and e-routing than for Merlin's scheme and this leads to a better overall performance of Merlin's algorithm over the other two algorithms even at low link utilization.

Figure 6.6 Average message delay in a hypercube under spatial locality in communication as a function of message generation rate

The performance of the algorithms as a function of the link service rate is shown in Figures 6.7 and 6.8. Merlin's algorithm allows packets to wait on not just one link, as the other two algorithms do, but on a set of links. This makes it less vulnerable to changes in the link service rate. Intuitively, the performance of store-and-forward packet switching algorithms is affected more by a locality in communication than the wormholing. This is because locality in communication implies a decrease in the average number of hops that a message has to take. Wormholing, however, performs well under large distances of communication and hence a decrease in the link service rate compounds the disadvantage of localized communication. This leads to a degradation of the performance of wormholing at

Figure 6.7 Throughput in a hypercube under spatial locality in communication as a function of variable link service rate

Figure 6.8 Average message delay in a hypercube under spatial locality in communication as a function of variable link service rate

2

of/de

2

PM-1 3½"x4" PHOTOGRAPHIC MICROCOPY TARGET
NBS 1010a ANSI/ISO #2 EQUIVALENT

PRECISIONSM RESOLUTION TARGETS

PIONEERS IN METHYLENE BLUE TESTING SINCE 1974

Micro-P

10000 COUNTY ROAD 6 BURNINGWELL, ARN 95007 USA
TEL 012 446 7007 FAX 012 446 7007 Tlx 64000000

a faster pace than that observed in the uniform communication. This can be seen in Figure 6.8 where the average message delay of wormholing surpasses that of Merlin's scheme at a lower value than in the uniform communication.

6.1.2. Performance Evaluation in a Torus

The performance of the algorithms with spatial locality on a hypercube was discussed in Section 6.1.1. The performance of the algorithms on the torus may differ. The difference in the structure of the torus leads to the deterioration in the performance of the store-and-forward algorithms. The torus has a fixed number of links and the average number of hops in a torus is greater than that in the corresponding hypercube. The results in Chapter 5 confirm these observations but with locality in communication the results should differ. The effects of spatial locality on the performance of the algorithms on a torus is discussed in this section.

Figure 6.9 and 6.10 present the performance of the algorithms as a function of the buffer capacity. The performance of all the algorithms is found to be much better than that with the uniform communication. The effect, however, is more pronounced for the store-and-forward algorithms than it is for the wormholing. The reason is the decrease in the average number of hops per message due to locality in communication. This leads to a substantial decrease in the average message delay for store-and-forward routing algorithms and hence a better performance. The number of buffers required by the algorithms to reach a steady state is also found to be less with locality. This can be explained by the fact that messages have to travel a shorter distance and hence are in the system for shorter periods of time, which leads to the freeing of buffers sooner for other messages. Even though a fewer number of buffers may be used for the simulation, the number of buffers will, for uniformity, be the same as those that were used in the torus for the uniform communication.

Figure 6.9 Throughput in a torus under spatial locality in communication as a function of the buffer capacity

Figure 6.10 Average message delay in a torus under spatial locality in communication as a function of the buffer capacity

Figure 6.11 Throughput in a torus under spatial locality in communication as a function of the system size

Figure 6.12 Average message delay in a torus under spatial locality in communication as a function of the system size

Locality also significantly influences the performance of the algorithms in large networks. With uniform communication, an increase in the network size implies an increase in the average distance that a message has to traverse from its source to its destination. This, however, is not the case with locality in communication. This may be explained by the fact that an increase in the system size has a minimal effect on the average number of hops that a message has to take as most of the messages are localized. These observations are confirmed by the performance of the algorithms shown in Figures 6.11 and 6.12.

Figure 6.13 Throughput in a torus under spatial locality in communication with variable message generation rate

Figures 6.13 and 6.14 show the performance of the algorithms as a function of the message generation rate. The throughput of the algorithms show little sensitivity though at high message generation rates the throughput of e-routing distinctly lags behind that of the other two routing algorithms. The average message delay for all the three algorithms is greatly reduced by locality of communication, but the increase in the message generation rate leads to an increase in the link utilization. Wormholing is very susceptible to any

Figure 6.14 Average message delay in a torus under spatial locality in communication and variable message generation rate

Figure 6.15 Throughput in a torus under spatial locality in communication with variable links service rate

changes in the link utilization and an increase in the message generation rate compounded with locality in communication lead to the degradation of the performance of wormholing at a lower link utilization than that in uniform communication.

Figure 6.16 Average message delay in a torus under spatial locality in communication with variable link service rate

The performance of the various algorithms with variable link service rate is shown in Figures 6.15 and 6.16. Merlin's algorithm allows packets to wait on not just one link but on a set of links. This makes it less susceptible to changes in the link generation rate than the other two algorithms. This can be seen in Figure 6.15 where the decrease in the link service rate has the least effect on the performance of the algorithms. The average message delay of the three algorithms increase due to the drop in the link service rate, but the increase in Merlin's algorithm is minimal. The other two algorithms, due to their routing, are forced to wait on just one link. This leads to an increase in the average message delay, due to the prolonged buffering of message at intermediate nodes in e-routing and the forced buffering of messages in wormholing. The effect of decreasing the link service rate is more severe for wormholing with locality in communication as it does not have the advantage of a large number of hops per message. This can be seen in Figure 6.16 where the average

message delay in wormholing surpasses that of Merlin's algorithm at a higher link service rate than in uniform communication.

6.2. Performance Evaluation under Temporal Locality

Temporal locality implies continuity in communication. In a network nodes that communicate do so with some form of locality or continuity. Nodes in a network communicate, but the form of communication may not be uniform or for that matter localized to a set of nodes. If a source node a sends a message to a destination node b , then there is a high probability that a will send another message to node b in the near future. As such communication is frequently the case in multicomputer systems, the performance of routing algorithms under such forms of communications may provide a benchmark for the performance of routing algorithms under real conditions.

Temporal locality in the models is achieved by memorizing the current destination address of a packet for each node. If the next packet generated at that particular node is to communicate with the same node, i.e., temporal locality in communication, then the current packet destination is the same as that of the previous packet. Otherwise any node (with equal probability) may be chosen as the destination node for this packet, i.e., uniform distribution in communication is assumed.

This section presents the performance of the three routing algorithms with temporal locality on the hypercube and the torus. The probability of temporal locality in communication is assumed to be 50%. The section is divided into two parts. Section 6.2.1 discusses the results of the simulations under temporal locality on a hypercube while Section 6.2.2 presents the results on a torus.

6.2.1. Temporal locality in a hypercube

The performance of the algorithms under temporal locality in a hypercube appears to provide a totally different result from that with the spatial locality. With the spatial locality the nodes were bound into a cluster and the probability of messages being inter- and-intra cluster messages were equal. Furthermore, a source node could communicate with any node within its cluster with equal probability. The case with temporal locality is, however, somewhat different. If the communication with spatial locality was relatively free, i.e. a node could communicate with a set of nodes within its cluster if communication is localized, then with temporal locality a node communicates with only one node. This localized communication leads to the increase in the utilization of links between a pair of source destination nodes.

The result of this form of communication can be seen in Figures 6.17 and 6.18 where the performance and the average message delay of the three routing algorithms are presented as a function of the buffer capacities. As can be seen from Figure 6.17, the throughput of wormholing and e-routing lags behind that of Merlin's scheme. This can be explained by the fact that with temporal locality the communication is very localized. The localization of communication leads to high utilization of some links as the same links are used for communication. The increase in the utilization of some links leads to the forced buffering of packets at intermediate nodes for wormholing and a prolonged buffering for the e-routing. Merlin's scheme allows packets to wait on not just one link but on a set of links at each node, the number of links in the set depends on the number of hops that remain for a message to reach its destination node. If two or more packets are bound for the same destination node then they can take different routes to reach their destination, the more the dimensions in the hypercube the larger the set of links on which a packet waits. As this is the case, temporal locality effects Merlin's scheme the least. Figure 6.18 shows the effect of temporal locality on the three algorithms. As can be seen from the figure, the average

Figure 6.17 Throughput in a hypercube under temporal locality in communication with variable buffer capacity

Figure 6.18 Average message delay in a hypercube under temporal locality in communication with variable buffer capacity

message delay for wormholing even at a message generation rate of 2 is much greater than that of Merlin's scheme. This can be compared with uniform and spatial locality in communication where the performance of wormholing is better than Merlin's scheme. The reason for the better performance of Merlin's scheme can be attributed to the fact that wormholing is very susceptible to changes in the link utilization and as temporal locality causes an increase in the link utilization, the performance of wormholing fares badly when compared to Merlin's algorithm. The performance of the e-routing is also seen to degrade with temporal locality. As the e-routing uses store-and-forward packet switching for communication the performance of e-routing is worse than that of the wormholing.

Figure 6.19 Throughput in a hypercube under temporal locality in communication as a function of the network size

Figure 6.19 and 6.20 show the performance as a function of the system size. The performance of all the algorithms is worse under temporal locality in communication than with spatial locality. The effect is not all that visible with system sizes but the average message delay increase proportionally with the increase in the system size. The performance of wormholing, however, fares well with that of Merlin's scheme for

large system sizes. As can be seen in Figure 6.20 the average message delay for wormholing approaches that of Merlin's scheme for large system sizes. This can be explained by the fact that wormholing uses a form of circuit switching while Merlin's scheme uses store-and-forward packet switching. When the system size is small the average number of hops that a packet has to take from its source to its destination is small while for large system sizes the number of hops increases proportionally. For wormholing the average message delay is not that dependent on the number of hops, hence at large system sizes the delay of Merlin's scheme approaches that of wormholing and even exceeds it.

Figure 6.20 Average message delay in a hypercube under temporal locality in communication as a function of the network size

Figure 6.21 and 6.22 show the performance of the algorithms as a function of the message generation rate. The performance for wormholing with a message generation rate of 2 lags behind that of Merlin's scheme and hence the message generation rate in this case is varied from 1 to 3 to give a better idea of how the performance of the algorithms change with varying message generation rate. The throughput of all the algorithms showed very

Figure 6.21 Throughput in a hypercube under temporal locality in communication as a function of the message generation rate

Figure 6.22 Average message delay in a hypercube under temporal locality in communication and variable message generation rate

little variation with e-routing giving the worst performance. The average message delay, however, varied by a great margin for all the three algorithms. For low message generation rate, wormholing has a lower average message delay than that of Merlin's scheme, but with the increase in the message generation rate the link utilization increases. Locality of communication compounds this liability and an increase in the message generation rate leads to the degradation of the performance of the e-routing and wormholing. At low message generation rates the link utilization is low and wormholing allows the packets to be forwarded without buffering at intermediate nodes. Merlin's scheme, which uses store-and-forward packet switching for communication, has to buffer packets at each node and so its average message delay is greater than that of the wormholing.

Figure 6.23 Throughput in a hypercube under temporal locality in communication as a function of the link service rate

The performance sensitivity of the algorithms to the link service rate is shown in Figures 6.23 and 6.24. The locality of communication leads to a degradation of the performance of all the algorithms. Figure 6.23 shows the throughput of the algorithms as a function of the link service rate. As can be seen from the figure at low link service rates

Merlin's algorithm performs much better than the other two algorithms. This is explained by the fact that Merlin's algorithm is least affected by the link service rate. The average message delay for the three algorithms is shown in Figure 6.23. Intuitively the average message delay for all the three algorithms should be much more than that with spatial locality. The performance of wormholing however, deteriorates to a greater extent. This is due to the fact that spatial locality allows a message to be sent to a set of nodes (either within a cluster or outside a cluster), whereas temporal locality sends a message to a given destination node. This increases the utilization of the links and thus leading to an increase in the average message delay.

Figure 6.24 Average message delay in a hypercube under temporal locality in communication with variable link service rates

6.2.2. Temporal locality in a torus

The effect of temporal locality on a hypercube was presented in Section 6.2.1. The effect of temporal locality should be the same in a torus, but this however is not the case. Whereas the performance of wormholing in the hypercube lagged behind that of Merlin

scheme in the hypercube, the relative performance is different in the torus. The reason for this may be explained by the fact that the torus has a fixed number of links. In a hypercube of dimension 6 (a 64-node hypercube) the number of links is 6 while in an equivalent torus the number of links is 4. The difference in the number of links makes a great difference to the performance of the algorithms such as Merlin's scheme, which is very susceptible to any changes in the number of links. Merlin's scheme allows packets to be routed on any link that brings it closer to its destination node. As the number of links in a torus is less than that in a hypercube and the average number of hops that a packet takes in a torus is greater than that in a hypercube, the performance of Merlin's algorithm in a torus will intuitively be worse than that in a hypercube. Wormholing is not that sensitive because the average number of hops has very little effect on the performance of the algorithm.

The performance of the three algorithms with variable buffer capacity is shown in Figure 6.25 and 6.26. The performance of all the algorithms shows a distinct degradation when compared to the performance with uniform communication or communication with spatial locality. The reason for the degradation is the localization of messages to not just a set of nodes as in spatial locality but to a particular node and hence the degradation in performance. This is shown in Figure 6.25. The average message delay is shown in Figure 6.26. The average message delay for all the three algorithms is, as expected, greater with temporal locality than that with spatial locality. This average message delay for wormholing was, however, found to be less in a torus than in a hypercube. This is consistent with the results found with spatial locality. The average message delay for the other two algorithms is greater in the torus than that in the hypercube. The reason for this is the greater number of hops that a message on average has to make in a torus from its source to its destination. As in the case with spatial locality, the number of buffers used for the simulation will be fixed at 32 so that the performance of the algorithms with temporal locality may be compared with that with the uniform communication.

Figure 6.25 Throughput in a torus under temporal locality in communication and variable buffer capacity

Figure 6.26 Average message delay in a torus under temporal locality in communication with variable buffer capacity

Figure 6.27 Throughput in a torus under temporal locality in communication with variable system size

Figure 6.28 Average message delay in a torus under temporal locality in communication with variable system size

The performance of the various algorithms with varying system sizes is shown in Figures 6.27 and 6.28. Wormholing provides a better average message delay than the other two algorithms. The reason is that e-routing uses the same routing algorithm as wormholing and can at best have a performance equal to that of wormholing. Merlin's scheme, due to the use of store-and-forward packet routing and the limited number of links (fixed at 4 for the torus) is unable to achieve a lower average message delay than that of the wormholing.

Figures 6.29 and 6.30 show the performance of the algorithms as a function of the message generation rate. The performance of the algorithms, as in the case with spatial locality in communication, shows the susceptibility of the wormholing to link utilization.

Figure 6.29 Throughput in a torus under temporal locality in communication with variable message generation rate

The throughput of Merlin's scheme was the least affected by the increase in the message generation rate while the e-routing was the most susceptible for reasons explained before. As can be seen in Figure 6.30 the average message delay for wormholing increases very

quickly, at a lower point than it did for spatial locality in communication. This is due to the fact that temporal locality causes the utilization of some links to increase at a much faster rate than others. With spatial locality in communication the effect was to communicate with a set of nodes more often than the rest, (the nodes inside a cluster), but with temporal locality in communication the communication is reduced to one node in particular. This causes the utilization of some links to increase very rapidly and to the forced buffering of packets at intermediate nodes.

Figure 6.30 Average message delay in a torus under temporal locality in communication with variable message generation rate

Performance sensitivity of the three algorithms to varying link service rate is shown in Figures 6.31 and 6.32. The performance degrades at a much faster rate than did with spatial locality in communication. The e-routing provides substantially better performance with increasing link service rates as it lacks the advantage of having a number of links to wait on at a node, that Merlin's scheme allows, nor does it follow the hybrid circuit switching that wormholing allows. Merlin's scheme is the least affected of the three. This is because it allows a packet to wait on a set of links at a node and if a link is busy, it

merely tries to use another link, hence its performance soon surpasses that of the wormholing.

Figure 6.31 Throughput in a torus under temporal locality in communication with variable link service rate

Figure 6.32 Average message delay in a torus under temporal locality in communication with variable link service rate

6.3. Summary

The performance of the three algorithms under locality of communication has been considered in this chapter. The performance of all the algorithms deteriorates under locality but the effect on wormholing and e-routing was considerably stronger than that on Merlin's scheme. This can be explained by the fact that Merlin's scheme allows packets to wait on not just one link but on a set of links. This causes locality to have a lesser effect as packets destined for the same node may take different routes to their destination. E-routing and wormholing however, have fixed routing schemes and any form of locality can have a great impact on their performance. The effect of spatial locality in communication was not as severe as that of temporal locality in communication. This is explained by the fact that spatial locality is not as rigid as temporal locality in communication. An effect of temporal locality in communication in the hypercube is that at even low link utilization Merlin's algorithm performs better than the wormholing. This was due to the fact that hypercubes have a large number of links and the number of hops that a message has to take is small as compared to the torus. In the torus it was seen that the initial message delay for wormholing was less than that for Merlin's scheme as the number of links is limited and the average number of hops per message is also greater.

CHAPTER 7

CONCLUSIONS AND FURTHER RESEARCH

The goal of this thesis was to study the performance of Merlin's algorithm, e-routing and wormholing under different workload models. The change in technology in recent years from store-and-forward packet switching to circuit or wormhole routing has led to an increased interest in the performance of circuit switching routing algorithms. Merlin's algorithm is an example of the former while wormholing is an example of the latter. The routing scheme for wormholing is based on e-routing, the only difference being that e-routing uses store-and-forward packet switching for the communication rather than a form of circuit switching that the wormholing uses. E-routing, therefore, provides a worst case scenario for wormhole routing. This chapter consists of two sections. Section 7.1 provides a summary and conclusions of the thesis work. Section 7.2 gives directions for further research.

7.1. Summary and Conclusions

Chapter 2 describes the interconnection networks and the various forms of deadlock. The interconnection networks were restricted to k-ary n-cube or isomorphic to k-ary n-cubes as most multicomputer systems have been built using such networks. The various forms of deadlock were also reviewed in this chapter as were some simple methods to prevent such forms of deadlock.

Chapter 3 presented various DP algorithms and the means by which they prevent deadlock. Four routing algorithms were used to illustrate how DP methods are used to

implement deadlock-free routing. The various algorithms were compared under several criteria including no message loss, size independence and freedom from livelock. All the algorithms, except for the DAG-based algorithm, provided no message loss in communication. A loss of this property implies that network internal end-to-end protocols requiring retransmission of acknowledgments are necessary. As message loss requires additional overheads and the possibility of other forms of deadlock cannot be disregarded, therefore a criteria of no message loss was imposed on the selection of the algorithms for simulation.

Chapter 4 described the experimental environment for both the torus and the hypercube. Although a 64-node hypercube was used for simulation, the routing schemes were described, for simplicity, on an 8-node hypercube. Examples of how routing might take place in more complex networks (i.e. e-routing in the cube-connected-cycle) were also explained. Wormholing does not allow message to be interleaved as the destination address of the message is only at the head of the message. This causes the blocking of message at nodes due to the blocking of messages ahead of them. To alleviate this problem the use of virtual lanes is presented.

In Chapter 5 the performance of the various algorithms under uniform communication was discussed. The performance of wormhole routing was found to be better than the performance of the other routing algorithms at low link utilization and for large system sizes, whereas Merlin's algorithm performed better under small system sizes and at high link utilization. E-routing performance was comparable to that of the other two algorithms only for small system sizes. Wormholing's main advantage lies in its ability to forward a message to the next node in it's route if the required link is free. At low link utilization the chances of a message being forwarded from its source node to its destination node without being buffered at an intermediate node is high. As the link utilization increases, the chances of a message being buffered at an intermediate node increases and

hence the performance of wormholing degrades. The system size has no appreciable impact on wormholing. In fact wormholing fares better for large system sizes than the other two algorithms. If a message is not buffered at an intermediate node then the increase in hops has no appreciable impact on wormholing. An increase in one hop is essentially an increase in the transmission time over one link hence wormholing performs well. This can be seen in the torus where an increase in the number of hops leads to no appreciable increase in the average message delay, while the delay for both the e-routing and Merlin's algorithm increases substantially. Merlin's scheme was found to compare favorably with wormholing of small system sizes and high link utilization rates. This was more evident in the hypercube than in the torus. This is because Merlin's algorithm allows a packet to wait on not just one link but on a set of links and the average message delay is dependent to large extent on the number of hops that a message has to make. For small system sizes, the average number of hops is small and hence the advantage that wormholing has over Merlin's scheme (i.e., a large number of hops) is lost. At high link utilization a packet is forced to wait on links. Merlin's scheme allows packets to wait on not just on one link but on any link that will take it closer to its destination node. Hence the effect of high link utilization on Merlin's scheme is minimal. The performance of Merlin's algorithm on a torus, however, differs substantially from that in a hypercube. In a torus, the algorithm has two disadvantages: the average number of hops increases and the number of links upon which a packet may wait is also reduced. These two factors play an important part in the performance of the algorithm and the performance of Merlin's algorithm in a torus is decidedly worse than in a hypercube.

The performance of the e-routing provides a worst case scenario for wormholing. E-routing uses the same DP technique as the wormholing, but uses packet switching to forward messages. Merlin's algorithm allows a message to wait on a set of links at any node, but the e-routing restricts the wait to just one link at any node. These restrictions

affect the performance of e-routing and its performance is comparable to that of the other two routing algorithms only for small system sizes.

Locality in communication can affect the performance of routing algorithms to a great extent. The effect of locality in communication was found to be quite acute. With spatial locality in communication, the performance of all the algorithms degraded, though the effect on Merlin's algorithm was minimal. This can be explained by the fact that Merlin's algorithm allows packets to take different routes to the same destination node. The effect of locality in communication is thus minimized as packets destined to the same destination node do not have to wait on just one link but on a set of links. The effect is more apparent on the torus than on the hypercube. The reason for this is the decrease in the number of links in the torus and the increase in the average number of hops. A decrease in the number of links leads to a decrease in the number of paths that a packet may take from its source to its destination and hence leads to an increase in the average message delay. The increase in the average number of hops also compounds this problem and hence the degradation in the performance of Merlin's algorithm in the torus. The effect of spatial locality in communication on e-routing and wormholing is more severe. This can be attributed to the fixed path that a packet has to follow between any source-destination pair of nodes. This leads to an increase in the utilization of the links between the pairs of nodes and, as both these algorithms are susceptible to any changes in the link utilization, the performance of these algorithms degrades.

Spatial locality in communication allows proximity in communication wherein a source node communicates more often with nodes within its own cluster. Temporal locality is far more rigid in its communication pattern and this is reflected in the performance of the algorithms under temporal locality in communication. A degradation in the performance of the algorithms is reflected in the results of the simulations, with wormholing and e-routing being affected the most. This is explained by the fact that temporal locality in

communication, like spatial locality, leads to an increase in the utilization of links between any source-destination pair of nodes. As temporal locality is more rigid in its communication, the performance of all the algorithms is affected more by temporal locality in communication than spatial.

7.2. Further Research

This thesis compared the performance of the wormholing to that of Merlin's algorithm and e-routing under different workload models and different forms of communication. The models were restricted to the hypercube and the torus. This brings forth several interesting avenues for research, among them the performance of these algorithms in other networks, for example in a hierarchical interconnection network (HIN) [Dandamudi89]. The advantage HIN is that they reduce the degree (i.e. the number of links connected to a node), of a majority of nodes. The reduction in the number of link connections has great implications on system design. It reduces the demand for both chip area and printed-circuit-board area and hence makes it feasible to increase the system size substantially. With locality in communication the communication may, to a great extent, be restricted to a cluster in the HIN (or a single level in the HIN) and the performance of these algorithms in such networks may provide interesting results.

The size of a message in Chapter 3 is restricted to one packet, (i.e. multipacket messages are not allowed). This, however may not always be the case. The performance of these algorithms using multipacket messages may also provide some interesting results. In such a case, the advantage that Merlin's algorithm has over wormholing may be considerably reduced. This may lead to a better performance of wormholing, even under conditions of locality in communication.

REFERENCES

[Arangho85]

Arangho, M., Hussein, B., and Gelenter, D., "Staged circuit Switching, " *IEEE Transactions on Computers*, Vol. C-34, February 1985, pp. 174-180.

[Blazewicz87]

Blazewicz, J., Brezezinski, J., and Gambosi, G., " Time-stamped approach to store-and-forward deadlock prevention," *IEEE Transactions on Communications*, Vol. Com-35, May 1987, pp. 490-495.

[Borkar80]

Borkar, S., Cohn, R., et all, "iWarp: An integrated Solution to high speed Parallel Computing," *IEEE Transactions on Computers* vol. C-29, May 1980, pp. 330-339.

[Chong88]

Chong-Kwan, K., Reed, D.A., "Adaptive Packet Routing in a Hypercube," *Proc. 3rd Conf. on Hypercube Concurrent Computers and Applications*, Pasadena, CA. pp. 625-630, Jan. 1988.

[Dally90a]

Dally, W.J., "Performance Analysis of K-ary N-cube Interconnection Networks," *IEEE Transactions on Computers* Vol. 30, June 1990 pp. 775-785.

[Dally90b]

Dally, W.J., "Virtual-Channel Flow Control," *Proc. 17th Int. Symp. Comp. Arch.* pp.60-68.

[Dallyseitz87]

Dally, W.J., Seitz, C.L., "Deadlock Free Message Routing in Multiprocessor Interconnection Networks," *IEEE Transactions on Computers* Vol. C-36, May 1987, pp. 547-553.

[Dandamudi89]

Dandamudi, S.P., and Eager, D.K., "Some issues in hierarchical interconnection network design," School of Computer Science, Carleton University, Ottawa, 1989. (available as SCR-TR-156).

[Gelernter81]

Gelernter, D., "A DAG-based Algorithm for Prevention of Store-and-Forward Deadlock in Packet-Switched Networks," *IEEE Transactions on Computers*, Vol. C-30, Oct. 1981, pp. 709-715.

[Gunter81]

Gunter, K.D., "Prevention of Deadlock in Packet-Switched Data Transport Systems," *IEEE Transactions on Computers*, Vol. C-29, April 1981, pp. 512-524.

[Karol87]

Karol, M.J., Hluchyj, M.G., and Morgan, S.P., "Input Versus Output Queuing on a Space-Division Packet Switch," *IEEE Transactions on Communications*, Vol. COM-35, December 1987, pp. 1347-1356.

[Kermani79]

Kermani, P., Kleinrock, L., " Virtual-Cut Through: A New Computer Communication Switching Technique," *Computer Networks*, Vol. No.3, 1979, pp. 267-286.

[Kleinrock74]

Kleinrock, L., and Naylor, W.E., "On the behavior of the APARNET Network, " *AFIPS conference proceedings*, NCC Chicago, May 1974, Vol. 43, pp. 767-780

[Marco87]

Marco, A., Emanuel, A., et al, "The Warp Computer: Architecture, implementation and performance," *IEEE Transactions on Computers* Vol. C-36, N.12, December 1987, pp. 1523-1537.

[Maxemchuk90]

Maxemchuk N.F., Magda El Zarki, "Routing and Flow Control in High-Speed Wide-Area Networks," *Proceedings of the IEEE*, Vol. 78, January 1990, pp. 204-221.

[Merlin80a]

Merlin, P.M., and Schweitzer, P.L., "Deadlock Avoidance in Store and Forward Networks-I: Store and Forward Deadlock," *IEEE Transactions on Communication*, Vol. Com-28, March 1980, pp. 345-353.

[Merlin80b]

Merlin, P.M., and Schweitzer, P.L., "Deadlock Avoidance in Store and Forward Networks-II: Other Deadlock Types," *IEEE Transactions on Communication*, Vol. Com-28, March 1980, pp. 353-363.

[Sengupta90]

Sengupta, A., and Dandopadhyay, S., "Deadlock-Free Routing in K-ary Hypercube Networks in presence of Processor Failures," *Information Processing Letters* 34(1990), pp. 323-328.

[Singal89]

Singal, M., "Deadlock Detection in distributed Systems," *IEEE Computer*, November 1989, pp. 37-48.

[Sullivan87]

Sullivan, H. and Brashkow, T.R., " A large scale homogenous machine," *Proc. 4th Annual Symp. Comp. Arch.*, 1977, pp. 105-124.

END

2/3·12·92

FIN