

様式 6

論 文 目 録

報告番号	甲 乙 工 工 工 修 工 修	第 6 8 号	氏 名	Andrew Colin Flavell
学位論文題目	High-Speed Message Routing Mechanisms for Massively Parallel Computers			
論文の目次				
第1章: Introduction				
第2章: Scalable Multicomputer Systems				
第3章: Tokkyu: A High-Performance, Randomizing, Adaptive Message Router with Packet Expressway				
第4章: Restricted Length Hardware Multicasting in Multicomputer Networks				
第5章: Conclusions				
参考論文				
主論文				
1. Flavell, A. C. and Takahashi, Y., "Tokkyū: A High-Performance, Randomizing, Adaptive Message Router with Packet Expressway", <i>IEICE Trans. on Information and Systems</i> , vol. E78-D, no. 10, pp. 1248-1260, October 1995.				
2. Flavell, A. C. and Takahashi, Y., "Restricted Length Hardware Multicasting in Multicomputer Networks", <i>Transactions of the IPSJ</i> , vol. 36, no. 5, pp. 1228-1238, May 1995.				
副論文				
1. Flavell, A. C. and Takahashi, Y., "The Tokkyū Router: A Randomizing Router for k-ary n-cubes", <i>Proc. of the International Symposium on Parallel and Distributed Supercomputing</i> , pp. 127-134, September 1995.				
2. Flavell, A. C. and Takahashi, Y., "Continuum: A Hybrid Time/Space Communications Paradigm for k-ary n-cubes", <i>Proc. of the International Conference on Parallel Processing 1994</i> , vol. I, pp. I38-I41, August 1994.				
3. Flavell, A. C. and Takahashi, Y., "Mandala: An Interconnection Network for a Scalable Massively Parallel Computer", in <i>Proceedings of the 33rd IPSJ Programming Symposium</i> , pp. 79-90, January 1992.				
4. Flavell, A. C. et. al., "Mandala: An Interconnection Network for a Scalable Massively Parallel Computer", <i>Technical Report of the IPSJ</i> , vol. 91, no. 100, pp. 91.101-91.109, November 1991.				

様式 7

論文内容要旨

報告番号	甲 乙 工	工 工 修	第 68 号	氏 名	Andrew Colin Flavell
学位論文題目		High-Speed Message Routing Mechanisms for Massively Parallel Computers			
<p>内容要旨</p> <p>現在超並列処理システム(MPP)は、伝統的なベクトルプロセッサや SIMD マシンの牙城であった多くの分野に進出している。これらのシステムは、入手が容易な高性能 CPU の急激な進歩をうまく利用し、これらを数百～数千個接続して均質なマルチプロセッサのシステムとして構成したものである。しかし、これらのシステムの性能は、現実の問題を解くときは必ずしも良くなく、常に公称の最高性能にははるかに及ばないのが現状である。これらのシステムではプロセッサ間の通信はすべて相互結合網によって行われるので、実現可能な最高性能を決める決定的な要素は相互結合網と、それに使われる通信機構である。</p> <p>本論文ではMPPの相互結合網に使われる、効率的な通信機構を実現する2つの方法を提案する。第1は「特急ルータ」の提案であり、これを相互結合網に用いた場合の適合性を検証する。特急ルータは多重の単方向レジスタ挿入バスを利用して、時間空間混合分割型ネットワークを実現するためのものである。異なる基数や次元数について、特急ルータのスイッチ回路とバッファ回路の性能を予測するための正確なモデルを開発した。この結果、特急ルータは効率的な通信を行うためのすべての条件を満足していることが確かめられた。さらに重要な点は、特急ルータはネットワークに故障のある場合や、通信が錯綜する場合にも、低遅延時間、高スループットを損なわない経路制御が行えることである。シミュレーションによって評価した特急ルータの性能は、これまでに発表された固定経路選択方式のルータより優れており、また他の適応経路制御方式のルータに比べても、同程度あるいはそれを越えていることが確かめられた。</p> <p>第2は経路長制限方式のマルチキャスト通信の提案である。マルチキャスト通信は多くの並列処理問題において速度向上に寄与する通信方式である。そこでワームホール通信方式において問題となるマルチキャスト通信におけるデッドロックの問題について研究した。そしてこの問題を解決する方法として経路長制限方式のマルチキャスト通信を提案し、この方式による通信性能をシミュレーションによって評価し、ユニキャスト方式やマルチパス方式によるマルチキャスト通信の性能と比較した。その結果、提案する経路長制限方式のマルチキャスト通信は、パリア同期のためのクラスタへのマルチキャスト通信や、最近傍ノードへのマルチキャストや全ノードへの放送の場合に、特に優れた解決法となることを明らかにした。</p>					

様式 9

論文審査の結果の要旨

報告番号	甲 工 乙 工 第 68 号 工 修	氏 名	Andrew Colin Flavell
審査委員	主 査 高橋 義造 副 査 島田 良作 副 査 赤松 則男		
学位論文題目	High-Speed Message Routing Mechanisms for Massively Parallel Computers		
審査結果の要旨	<p>超並列計算機は、数百～数千個のプロセッサ要素を接続して並列に動作させ、超高速処理を行わせようとするものである。ここではプロセッサ間の通信はすべて相互結合網によって行われるので、このシステムの総合性能を決める決定的な要素は相互結合網の通信機構と通信制御方式になるが、まだ十分に満足できるものが得られていないのが現状である。</p> <p>本論文では相互結合網の通信機構と通信制御方式について研究し、新方式のルータ機構と、独特の制御を行うマルチキャスト通信方式を提案している。新しいルータ機構を「特急ルータ」と呼んでいるが、多重の単方向レジスタ挿入パスを用いて時分割・空間分割混合型ネットワークを実現し、ネットワークに故障のある場合や著しく通信量が多い場合にも、低遅延時間、高スループットを損なわない経路制御が行えることを特長としている。実際シミュレーションによって詳細な性能評価を行った結果、従来の固定経路選択方式のルータより優れ、他の適応経路制御方式のルータに比べても、遜色のない性能を持つことが確かめられている。</p> <p>次に新しい通信方式としてパケット長制限方式マルチキャスト通信を提案している。マルチキャスト通信は多くの並列処理問題において必要とされる機能であるが、これをできるだけ高速に行う必要がある。しかしワームホール通信の場合にはマルチキャスト通信はデッドロックを起こす可能性があるという問題がある。この問題を研究した結果、パケット長を自動的に制限してマルチキャスト通信を行えば、性能を損なうことなくデッドロックを回避できることを証明した。また、シミュレーションによってこの方式の通信性能を評価した結果、バリエ同期のためのクラスタへのマルチキャスト通信や、最近傍ノードへのマルチキャストや全ノードへの放送の場合に、特に優れた効果を発揮することを確かめられた。</p> <p>以上本研究は高性能の超並列計算機を構成するための重要な要素である相互結合網について、その通信機構と通信制御方式についての新しい提案を行い、その効果を実証したものであり、本論文は博士（工学）の学位授与に値するものと判定する。</p>		

High-Speed Message Routing Mechanisms
for Massively Parallel Computers

March 1996

Andrew C. Flavel

(2)

High-Speed Message Routing Mechanisms for Massively Parallel Computers

March 1996

Andrew Colin Flavell

High-Speed Message Routing Mechanisms for Massively Parallel Computers

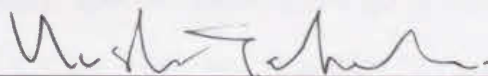
A dissertation submitted to the Department of Information Science and
Intelligent Systems and the Graduate School of the University of
Tokushima in partial fulfillment of the requirements for the degree of
Doctor of Engineering

by

Andrew Colin Flavell

March 1996

Approved as to the style and content by



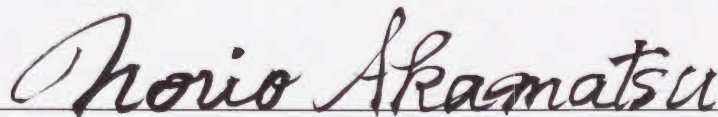
Professor Yoshizo Takahashi

Dept. of Information Science and Intelligent Systems
Dissertation Supervisor



Professor Ryosaku Shimada

Dept. of Information Science and Intelligent Systems



Professor Norio Akamatsu

Dept. of Information Science and Intelligent Systems

Acknowledgments

I wish to express my sincere gratitude to Professor Yoshizo Takahashi, for enabling me to study for a doctoral degree in Japan. His guidance has served me well and has helped to keep me focused on the task at hand. I also wish to thank Professors Ryosaku Shimada and Norio Akamatsu, for their contributions as the members of my defense committee. Thanks must also go to the Japanese Ministry of Education, Science and Culture, for granting me the scholarship which has made studying in Japan a reality.

To Masahiko Sano and Tomio Inoue, many thanks for helping to make my university life, and adjustment to life in Japan, simpler and more enjoyable. Thanks to Dr. Tim Gleeson for his useful and constructive criticism of my written work, especially the comments relating to this dissertation.

Finally, special thanks must go to my wife, Figen Ulgen. Her belief in my ability has been, and continues to be, an inspiration. I couldn't wish for anything more...

Abstract

Massively parallel processing systems (MPPs) are currently making inroads into many areas that are traditionally a stronghold for vector or SIMD processors. These systems leverage the rapid advances being made in readily available high performance CPUs by connecting hundreds or thousands of them together to form homogeneous multiprocessor systems. Unfortunately, the performance of these systems when solving real-world problems has been somewhat disappointing and always falls far short of the theoretical peak performance quoted by system vendors. As all of the communications between processors in these systems rely on the interconnection network, a critical component in determining the maximum achievable performance is the interconnection network and the communications structures supported by it.

This dissertation introduces two solutions to providing effective communications structures for MPP systems. The Tokkyū router is presented and its suitability for use in MPP interconnection networks is demonstrated. The Tokkyū router utilizes multiple, unidirectional, register-insertion buses to provide a hybrid time/space division network. Accurate models are developed to predict the switch and buffer performance of Tokkyū routers for varying radix and dimension. The Tokkyū router meets all of the requirements necessary to be considered effective. Importantly, the support for routing in the presence of faults or network congestion does not compromise the low latency and high throughput of the router. The simulated performance of the Tokkyū router exceeds that of published results for oblivious

routers and is equal to or exceeds those reported for other adaptive routers.

The multicast deadlock problem is investigated, as multicast has been identified as an area which can provide significant speedup to a number of parallel applications. Restricted-length multicast is introduced as a solution to multicast deadlock in wormhole routed networks and the implementation of this multicast scheme is examined. Restricted-length multicast is then compared to unicast and multi-path based multicasts by simulation. The results of the simulations indicate that restricted-length multicast provides a good solution to multicast problems such as multicasting to clusters of nodes found in barrier synchronization, multicasting to nearest neighbors and broadcasting to all of the nodes in the network.

List of Publications

Papers Accepted for Journal Publication

- Flavell, A. C. and Takahashi, Y., "Tokkyū: A High-Performance, Randomizing, Adaptive Message Router with Packet Expressway", *IEICE Trans. on Information and Systems*, vol. E78-D, no. 10, pp. 1248-1260, October 1995.
- Flavell, A. C. and Takahashi, Y., "Restricted Length Hardware Multicasting in Multicomputer Networks", *Transactions of the IPSJ*, vol. 36, no. 5, pp. 1228-1238, May 1995.

Papers Accepted to International Conferences

- Flavell, A. C. and Takahashi, Y., "The Tokkyū Router: A Randomizing Router for k-ary n-cubes", *Proc. of the International Symposium on Parallel and Distributed Supercomputing*, pp. 127-134, September 1995.
- Flavell, A. C. and Takahashi, Y., "Continuum: A Hybrid Time/Space Communications Paradigm for k-ary n-cubes", *Proc. of the International Conference on Parallel Processing 1994*, vol. I, pp. I38-I41, August 1994.

Other Related Papers

- Flavell, A. C. and Takahashi, Y., "Mandala: An Interconnection Network for a Scalable Massively Parallel Computer", in *Proceedings of the 33rd IPSJ Programming Symposium*, pp. 79-90, January 1992.
- Flavell, A. C. et. al., "Mandala: An Interconnection Network for a Scalable Massively Parallel Computer", *Technical Report of the IPSJ*, vol. 91, no. 100, pp. 91.101-91.109, November 1991.

Contents

Abstract	iii
List of Publications	v
1 Introduction	1
2 Scalable Multicomputer Systems	5
2.1 Node Structure	5
2.2 Interconnection Network Topologies	6
2.3 Message Switching	14
2.4 Message Routing	17
2.4.1 Deterministic Routing	18
2.4.2 Adaptive Routing	22
2.5 Deadlock	26
2.6 Multicast Messages	28
2.6.1 Multicast Deadlock	31
3 Tokkyū: A High-Performance, Randomizing, Adaptive Mes-	
sage Router with Packet Expressway	35
3.1 The Register-insertion Bus	36
3.1.1 Register-insertion Bus Operation	36
3.2 Architecture of the Tokkyū Router	40
3.2.1 Router Operation	41
3.3 Switch and Buffer Design	49
3.3.1 Switch Evaluation	49
3.3.2 Buffer Evaluation	56

3.4	Performance	59
3.4.1	Simulation of Uniform Random Traffic	65
3.4.2	Simulation of Hot-spot Traffic	70
3.4.3	Simulation of Traffic in the Presence of Faults	71
3.4.4	Discussion of Results	74
4	Restricted-length Hardware Multicasting in Multicomputer Networks	76
4.1	Preliminaries	76
4.1.1	Definition of Multicast Deadlock Problem	76
4.2	Restricted-Length Multicasting	81
4.2.1	Gate-array Implementation	83
4.3	Simulation	86
4.3.1	Multicast Latency	86
4.3.2	Simulation Results	87
4.3.3	Discussion of Results	90
5	Conclusions	91

List of Figures

1.1	Generic multiprocessor architecture	2
2.1	Generic node architecture	6
2.2	(a) Simple ring network and (b) corresponding spanning sub-graph	7
2.3	(a) Strongly connected digraph and (b) corresponding directed tree, which is also a rooted tree.	8
2.4	Contemporary static network topologies:(a) 2D torus (b) 2D mesh (c) 3D mesh (d) 4D Hypercube (e) Fat-Tree (f) Mandala interconnection network.	10
2.5	Communications channels for a 2-dimensional router	11
2.6	Latency of various switching techniques	14
2.7	Division of information units	16
2.8	Three virtual channels sharing a unidirectional physical channel	17
2.9	e-cube routing on a hypercube	20
2.10	Dimension order routing on a 2D mesh	21
2.11	(a) Dimension order routing (b) Adaptive routing	22
2.12	Physical communication channels divided into routing planes .	23
2.13	Two-dimensional chaos router	25
2.14	(a) Network and (b) its channel dependency graph without virtual channels. (c) Network and (b) its' channel dependency graph with extra virtual channels.	27
2.15	(a) Multicast by unicast (b) Tree based multicast (c) Path based multicast	30
2.16	Multicast deadlock in binary tree	31
2.17	Multipath multicast	33

3.1	Register-insertion bus interface	38
3.2	N -dimensional register-insertion bus port.	40
3.3	Architecture of a two-dimensional Tokkyū router	42
3.4	Global arbiter inputs and outputs	47
3.5	State diagram for determining the distance distribution in an 8-ary 2-cube	53
3.6	Probability of misrouting versus applied load for 16-ary 2- cube. Solid lines are predicted values, points are measure- ments taken by simulation	55
3.7	The discrete-time Markov chain state transition diagram for the output queue size	57
3.8	Performance of output queues. Solid lines are predicted values, points are measurements take by simulation	58
3.9	Dialog for setting simulation variables	60
3.10	(a) Simulation display showing test mode (b) Simulation dis- play key	61
3.11	(a) Simulation display showing random simulation (b) Simu- lation display key	62
3.12	(a) Simulation display showing hot-spot simulation (b) Simu- lation display key	63
3.13	(a) Simulation display showing fault simulation (b) Simulation display key	64
3.14	Performance of queue switches for 256 node 16-ary 2-cube. Solid lines are predicted values, points are measurements taken by simulator	66
3.15	Performance of output queues for 256 node 16-ary 2-cube. Solid lines are predicted values, points are measurements taken by simulator	67
3.16	Latency versus offered traffic for a 256 node 16-ary 2-cube under uniform random traffic	68
3.17	Throughput versus offered traffic for a 256 node 16-ary 2-cube under uniform random traffic	69

3.18 Latency and reduction in latency versus applied load under uniform random traffic with <i>packet expressway</i> enabled and disabled	69
3.19 Latency versus offered traffic for a 256 node 16-ary 2-cube under bit reversal traffic	70
3.20 Throughput versus offered traffic for a 256 node 16-ary 2-cube under bit reversal traffic	71
3.21 Faulty node is bypassed	72
3.22 Average latency versus percent faulty channels at 50% applied load ($m=2$, $L=2$). Mean latency averaged over ten random fault sets	73
3.23 Throughput versus percent faulty channels at 50% applied load ($m=2$, $L=2$). Mean throughput averaged over ten random fault sets	73
4.1 (a) Multicast by node (2,1) and (b) the resulting concurrent resource trees	81
4.2 Organization of a single MEGA router input	83
4.3 Send latency for $L_m = 16$ bytes	89
4.4 Send latency for $\text{Pr}(b) = 0.5$	89

List of Tables

2.1	Routing steps from $s = 0000$ to $d = 1101$	20
3.1	2-tuples defining total distance to travel and Ψ_{d_g} for packets in an 8-ary 2-cube	52
3.2	Probability of j dimensions remaining to be traversed	54
4.1	Resource usage for various buffer structures	85

Chapter 1

Introduction

The peak performance levels of Massively Parallel Processing (MPP) systems have recently begun to rival those which are obtained using traditional vector and SIMD supercomputers. Many therefore believe that MPP systems, constructed by the interconnection of thousands of homogeneous computational nodes, are a promising technology for the construction of computers with teraflops performance. However, the efficiency of multicomputer based MPP systems when solving real world problems tends to be disappointing, especially when compared to vector supercomputers [11, 20].

Although there are many ways in which the nodes of an MPP system can be connected, by far the most popular is the static or direct network. Each node in a direct network has a point-to-point, or direct, connection to its 'neighboring' nodes and these connections form the interconnection network as is illustrated in Fig. 1.1. Direct networks are popular as they are said to scale well, i.e. as the number of nodes in the system is increased, the total processing power, communication bandwidth and memory bandwidth of the system also increases.

Inter-process communication, data-sharing and synchronization in an MPP

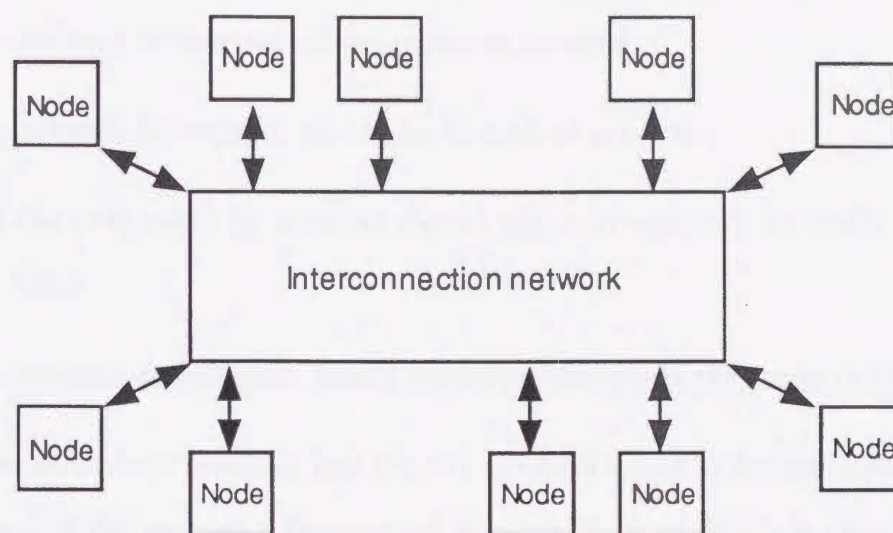


Figure 1.1: Generic multiprocessor architecture

system are all achieved by the passing of messages via the interconnection network (IN), and therefore a critical component in determining the maximum achievable performance of MPP systems is the IN and the communications structures supported by it. A considerable amount of research has therefore been conducted in both the design and evaluation of interconnection networks [1, 2, 46, 5, 42, 47, 22, 23, 24, 25, 26, 27, 28, 29], and this continues to be an active avenue of research.

The interconnection networks of massively parallel systems must provide effective, dynamic and arbitrary connectivity between all of the processors in the system. In order to be considered effective it is desirable that the interconnection network satisfies the following requirements:

- the packet routing algorithm must be free from deadlock
- the network must be free from livelock, i.e. packets must not be infinitely delayed in the network

-
- network latency should be as low as possible
 - network throughput should be as high as possible
 - the path taken by a packet should adapt dynamically to traffic conditions
 - network performance should degrade gracefully in the presence of faults

Freedom from deadlock and livelock are both essential for the correct operation of the network. Guaranteed freedom from deadlock is essential to ensure that there is no potential for the network being brought to a complete halt because of dependencies in the allocation of network resources, and freedom from livelock is essential to ensure that packets do not endlessly cycle in the network, never reaching their destinations. Low latency and high throughput are necessary to allow a good balance of the computation/communication ratio of the system. Adaptive packet routing and graceful degradation of network performance in the presence of faults are both desirable features, provided they do not compromise the latency and throughput of the network[44]. Adaptive routing allows better utilization of communication resources, especially at high network loads or in the presence of hot-spot traffic [31, 9, 32, 35, 41, 15], and networks which are fault tolerant are becoming increasingly important as the size and complexity of massively parallel systems grows. In addition to these requirements, multicast communication, in which a source node transmits a single message to a number of destination nodes in the system, has been identified as being crucial to achieving acceptable performance in a number of application areas[37, 51].

Organization of this Dissertation

This dissertation focuses on simple and effective solutions to meeting the requirements for an IN to be considered effective and is divided into two distinct areas. An introduction to scalable multicomputer systems is given in Chapter 2 and this is followed in Chapter 3 with an examination of adaptive routing in multicomputer networks and the introduction and investigation of the *Tokkyū* interconnection network. In Chapter 4 an examination of multicast deadlock in wormhole routed networks is given and the concept of *restricted-length multicasting* is introduced and investigated. Finally, a summary and conclusions are given in Chapter 5.

Chapter 2

Scalable Multicomputer Systems

2.1 Node Structure

Each node in most current MPP systems contains an off-the-shelf RISC processor, local memory, a number of support units, an interface to a communications network and a message router, as illustrated in Fig. 2.1. Off-the-shelf processors are often chosen for MPP system construction as they are inexpensive and can help to reduce the design time of the system. For example, the Connection Machine CM-5 uses 32-MHz SPARC processors, the NEC Cenju-3 uses 75-MHz NEC VR4400SC processors and the Intel Paragon XP/S uses 50-MHz i860 processors. Support units may include vector processing units, a graphics controller and HIPPI, SCSI, ethernet or some other I/O interface. The role of the network interface unit is to perform message assembly/disassembly and provide flow control for messages entering and leaving the network, while the router provides routing and flow control for messages within the communication network. By removing the functions of message assembly/disassembly, routing and flow control from the CPU, com-

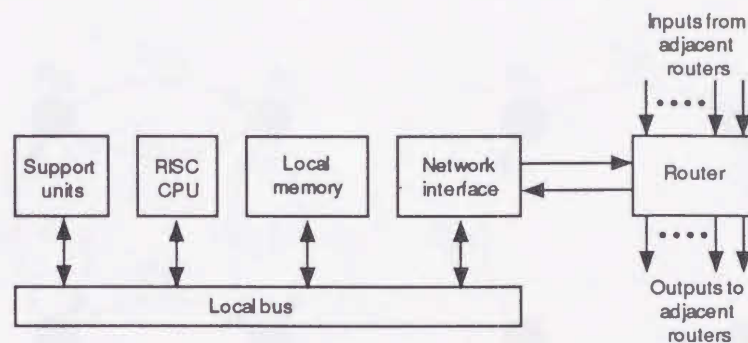


Figure 2.1: Generic node architecture

munication and computation can occur concurrently, significantly increasing the performance of the system.

2.2 Interconnection Network Topologies

The topology of a network defines how the nodes are connected and can usually be represented using graph notation. Therefore, a brief introduction to the relevant graph theory notation is presented before the discussion of static interconnection networks.

Definition 1 A static interconnection network may be represented by the strongly connected directed graph, *digraph*, $I = G(N, C)$, where the vertex set $N(I)$ and the arc set $C(I)$ represent the nodes and channels of the network respectively. The degree of a vertex n , in I , denoted $d(n)$, is the number of edges incident with n . The graph $H = G(N, C)$ is a *subgraph* of I if $N(H) \subseteq N(I)$ and $C(H) \subseteq C(I)$, and H is a *spanning subgraph* of I if $N(H) = N(I)$.

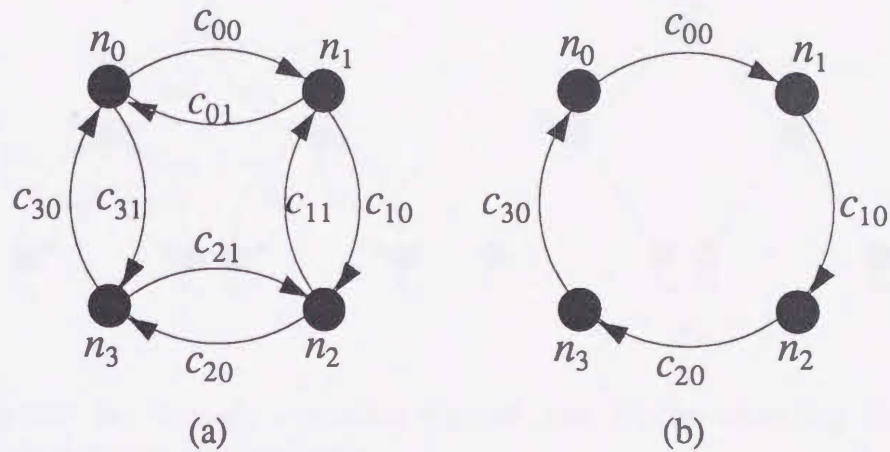


Figure 2.2: (a) Simple ring network and (b) corresponding spanning subgraph

Figures 2.2(a) and (b) illustrate Definition 1. Figure 2.2(a) presents a simple ring network, which is a strongly connected digraph and Figure 2.2(b) represents a spanning subgraph of (a), as it contains the same set of nodes.

Definition 2 A *tree* is a connected graph which contains no cycles, and it follows that a subgraph which is a tree is called a *subtree*, and a spanning subgraph which is a tree is called a *spanning tree*. A *directed tree* is a digraph which becomes a tree when the directions of the edges are ignored and a *rooted tree* is a directed tree with one vertex of *in degree* 0, and all other vertices of *in degree* 1.

Figures 2.3(a) and (b) illustrate Definition 2. Figure 2.3(a) presents a strongly connected digraph and Figure 2.2(b) represents the corresponding directed tree. This tree is a binary tree and therefore it is also a rooted tree.

Some of the more important static evaluative measures of an interconnection network are its degree, diameter, average distance [2], channel bisection

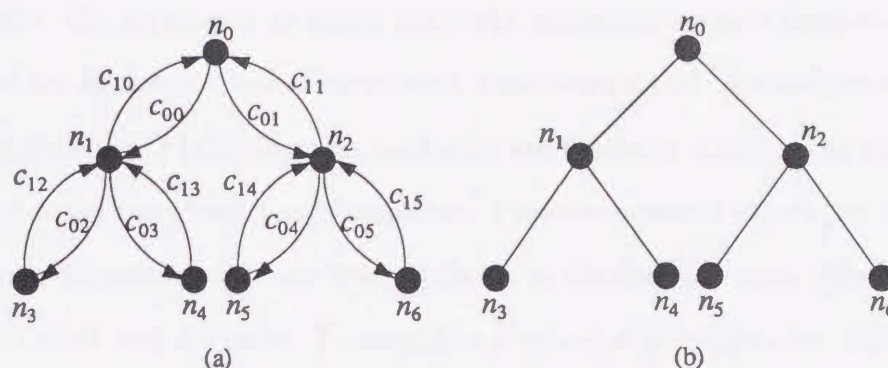


Figure 2.3: (a) Strongly connected digraph and (b) corresponding directed tree, which is also a rooted tree.

width [12], maximum message density, and its ability to be scaled. The degree is defined as the number of channels incident on a node, the diameter as the maximum of the shortest distances between any two nodes in a system, and the average distance as the average number of channels that a message must traverse when traveling from a source node to a destination node. As the degree of a node and the average distance for a given network are often inter-related, the normalized average distance, given by *average distance* \times *degree*, may provide a more useful measure for static evaluation. The channel bisection width, B , is defined as the minimum number of channels that, when cut, separate the network into two equal parts, and the maximum message density is the maximum of the total number of communications paths passing through each node in the system. Scalability is defined as the relative ease with which the number of processing elements in a system can be increased. A system which requires major hardware changes and/or rewiring to increase the number of processors is therefore not considered scalable when compared to a system in which an additional processor can be plugged in. Feng [21]

classified the topologies of static networks according to the dimensions required for layout, i.e. one-dimensional, two-dimensional, three-dimensional, and hypercube. Multicomputer networks are typically constructed from arrays of nodes in at least two-dimensions. Two-dimensional topologies include the ring, 2D mesh, torus and tree, while three-dimensional topologies include the 3D mesh and 3D torus. Presented in Figure 2.4 are a number of contemporary static network topologies.

The networks under consideration here are bi-directional, as these networks allow locality of communication to be employed in the programming model of the parallel machine. Therefore, each arc in Fig 2.4 is divided into two communications channels, one in each direction. A router in a 2-dimensional network will have communications channels in the $+x$, $-x$, $+y$ and $-y$ directions, along with a connection to the local processor, as shown in Fig. 2.5.

Torus

The torus of Fig. 2.4(a) is a member of the general k -ary n -cube family. For the example torus of Fig. reffig:static, $k = 4$ and $n = 2$.

Definition 3 A k -ary n -cube is an n -dimensional cube of radix k , and a node within a k -ary n -cube can be identified by the n -digit radix k address, $(a_0, a_1, \dots, a_{n-1})$. Each node in a k -ary n -cube is connected to every other node whose address differs in exactly one digit by ± 1 modulo k .

The number of nodes in the network, N , is related to n and k by:

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

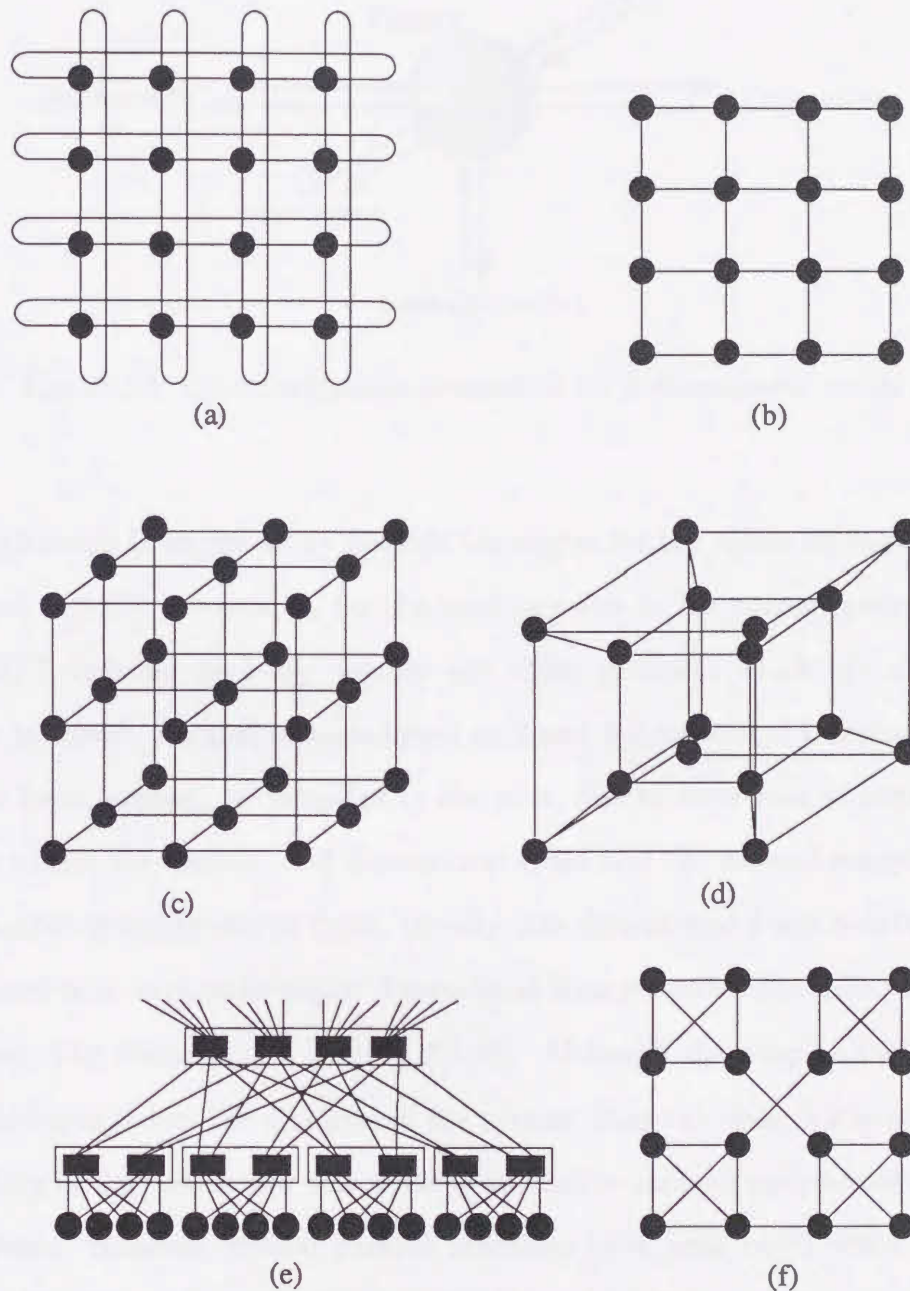


Figure 2.4: Contemporary static network topologies: (a) 2D torus (b) 2D mesh (c) 3D mesh (d) 4D Hypercube (e) Fat-Tree (f) Mandala interconnection network.

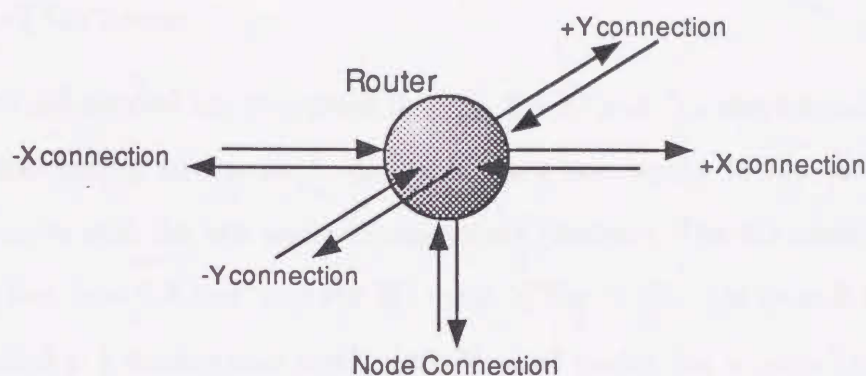


Figure 2.5: Communications channels for a 2-dimensional router

Although there are many possible topologies for the direct networks employed in MPP systems, by far the most popular in the current generation of MPP systems are k -ary n -cubes and those networks which are isomorphic to them¹. Parallel systems based on 2 and 3-dimensional k -ary n -cubes have been intensely investigated in the past, due to their ease of construction within the confines of 3-dimensional space and the natural mapping of a number of algorithms to them. Usually, low dimensional k -ary n -cubes are referred to as *tori*, while higher dimensional binary n -cubes are called hypercubes. The diameter of a torus is $2\lfloor n/2 \rfloor$. Although the wrap-around links of the torus reduce the diameter of the system, they can complicate message routing in the system and may make it difficult to connect peripherals to the network. However, several parallel machines have been constructed using tori. The 2D torus is used in the iWarp[6] and the K2 parallel processor[3], and more recently, the 3D torus has been used in the construction of the Cray Research T3D[43].

¹One notable exception to this is the CM-5, which is based on a fat-tree IN[36]

2D and 3D Mesh

2D and 3D meshes are presented in Figs. 2.4(b) and (c) respectively. The mesh topology is an aperiodic variant of the k -ary n -cube family, and looks like a torus with the end around connections removed. The 2D mesh of Fig. 2.4(b) has $(n = 2, k = 4)$ and the 3D mesh of Fig. 2.4(b) has $(n = 3, k = 3)$. In general a k -dimensional mesh with $N = n^k$ nodes has a node degree of $2k$ and a network diameter of $k(n - 1)$. Several simple routing algorithms have been presented for the mesh, including fault tolerant algorithms, and the unused connections around the edge of the mesh provide an abundance of connections for peripheral devices. A number of commercial parallel computers have been constructed based on the 2D mesh, including the CM-2 and the Intel Paragon [53], and a 3D mesh has been utilized in the J-machine[16] and the Wavetracer Inc. Data Transport Computer[53].

Binary Hypercube

The 4-dimensional binary hypercube of Fig. 2.4(d) is a member of the k -ary n -cube family, with k fixed at two. The hypercube, as it is often referred to, has a network diameter of n , which is one of the lowest known average communications distances of any IN. Many numerical algorithms are suited to this topology, and it is simple to embed other topologies in the hypercube. The main disadvantage of the hypercube is that the number of nodes in the system is increased by increasing the dimension of the network. Thus a large number of connections are required for each node if a large system is to be built. In spite of this, the hypercube topology has been used for a number of commercial and research machines including the Cosmic Cube, CM-2 and

nCube corporations nCube2.

Fat-Tree

The fat-tree takes a somewhat different approach to implementing a static IN. A typical binary tree has a bisection width of only 1, which results in severe message-traffic congestion at the root node of the tree. The number of communications channels, and therefore the communications bandwidth in a fat-tree, increases as you move up the tree hierarchy, thus alleviating the communications bottleneck experienced by a standard binary tree interconnection network. One disadvantage of this scheme is that it requires several different types of routing nodes and the number of communications channels in the hierarchy can become very large. However, the network is quite practical as the Connection Machine Corporation CM-5 is constructed using a 4-ary fat-tree [36]. The 4-ary fat-tree of Fig. 2.4(e) has clusters of four processors located at the leaves of a tree, each of which is connected to two router chips.

Mandala

The Mandala network, presented in Fig. 2.4(f), is a hierarchical network proposed by Takahashi and Flavell [22, 23, 24]. It can be described by the size of its clusters, C and number of levels, L . For example the network in Fig. 2.4(f) is a (4,2) Mandala network. The number of nodes in this system is given by $N = CL$. Each of the nodes in a network of cluster size C , has $C - 1$ communications channels forming a complete connection at level 1, with 1 channel per node reserved for connection to higher levels. The degree

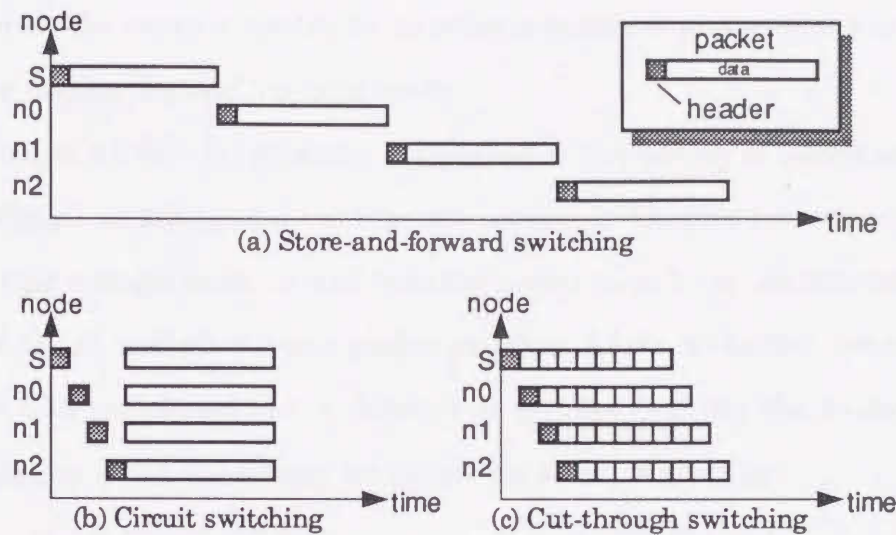


Figure 2.6: Latency of various switching techniques

of each node is given by C and the average distance is given by $\sqrt[3]{N}$.

2.3 Message Switching

The message switching technique, i.e. the method by which data is passed from the input of a router to the output, can have a significant effect on the *latency* of the network. There are a number of possible switching techniques and these include circuit switching, packet switching, virtual cut-through routing and *wormhole* routing. Circuit switching was originally used in telephone networks and involves the formation of a physical channel between the source and destination nodes. In packet switching, or store-and-forward networks, complete packets are buffered at each node between the source and destination and the header of a packet may not leave a node until the tail has been received.

Both virtual cut-through [34] and wormhole routing [12] use *cut-through*

to reduce the network latency by allowing a packet to be forwarded as soon as the routing decision has been made.

Figures 2.6 (a) - (c) present a comparison of the latency of packet switching, circuit switching and cut-through routing techniques respectively. In each case a single packet is sent from the source node S via the intermediate nodes n0, n1 and n2. Given a packet length of L bits, a channel bandwidth of W bits per second and a distance of D hops between the source and destination nodes the latency for circuit switching is given by

$$T_{cs} = T_{setup} + \frac{L}{W} + D \quad (2.1)$$

the latency for cut-through routing is given by

$$T_{ct} = \frac{L}{W} + D \quad (2.2)$$

and the latency for store-and-forward switching is given by

$$T_{sf} = \frac{L}{W}(D + 1) \quad (2.3)$$

If $L \gg D$ then T_{ct} becomes L/W and thus the distance has negligible effect on latency. Clearly the latency of store-and-forward routing is considerably higher than that of both circuit and cut-through routing. Also, in the absence of contention, the network latency of cut-through based switching is similar to that of circuit switching. However, if there is a large amount of contention in the network, the time taken to establish a complete circuit between the source and destination nodes can add a considerable amount to the delay of a circuit switched message.

When channels become blocked, networks using wormhole routing buffer only small units of data called flow control digits or *flits* which are illus-

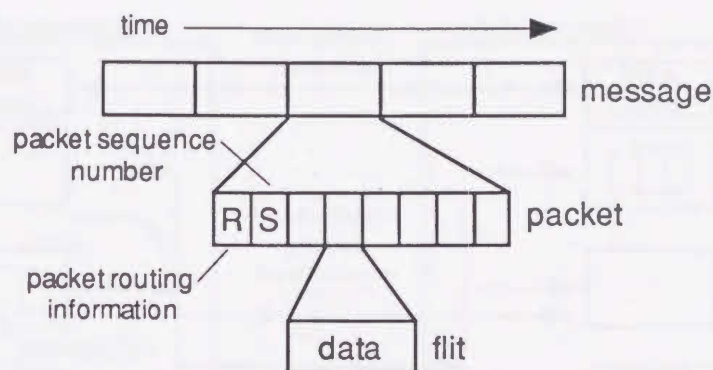


Figure 2.7: Division of information units

trated in Fig. 2.7, whereas networks employing virtual cut-through routing buffer entire packets and therefore requires considerably more buffer resource than wormhole routing. Wormhole routing and virtual cut-through routing provide low latency message delivery and often make use of *virtual channels*, which can significantly improve the throughput of an interconnection network [13]. Moreover, deadlock free routing algorithms for many multicomputer topologies which utilize these switching mechanisms have been proposed [17, 30]. Virtual channels provide excellent channel utilization and allow multiple disjoint logical networks to coexist on a single physical network, which is very useful for adaptive routing. Figure 2.8 presents a physical channel which is being shared by three virtual channels. Even though two of the destination buffers are full, the physical channel can still be utilized as the third destination buffer is free. Thus, the data in the free channel can pass the data in the blocked channels.

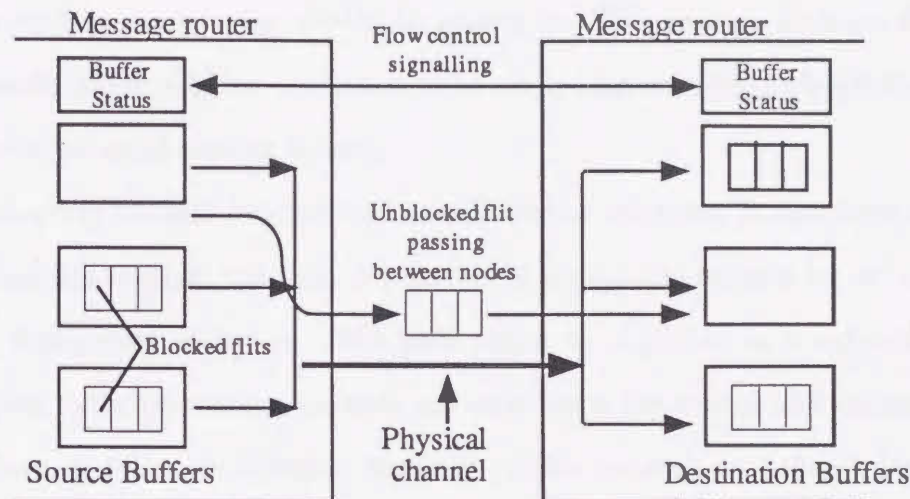


Figure 2.8: Three virtual channels sharing a unidirectional physical channel

2.4 Message Routing

The routing of a message in a direct IN involves the selection of an appropriate path from the source node to the destination node. Routing can be classified in several ways. In *source routing*, as the name implies, the source nodes determines the entire path of a packet prior to injecting it into the network. While this method may reduce the complexity of the message router hardware, it requires that each packet carry the information in its' header, increasing the packet size. Also, the path of the packet is fixed and cannot be changed once it has left the source node. Most current state-of-the-art direct INs employ *distributed routing*. In this case a routing decision is made at each intermediate router which lies on the path between the source and the destination nodes. The decision process determines whether the packet should be delivered to the local processor or forwarded to a neighboring router. If the message is to be forwarded, the routing algorithm decides which of the adja-

cent routers the message should be passed to. This routing decision should be as simple as possible to allow it to be easily implemented in hardware and provide minimal routing latency.

Routing can also be classified as *oblivious* or *adaptive*. In oblivious or *deterministic* routing, the path of a packet is completely defined by its' source and destination addresses. The path taken by a packet in a network employing dynamic routing depends not only upon the source and destination address, but also on dynamic network conditions such as network load, or the presence of faulty channels.

2.4.1 Deterministic Routing

Most current state-of-the-art interconnection networks employ deterministic routing. Although deterministic routers are not fault tolerant and have poor performance in networks experiencing high traffic loads or hot-spots, they are extremely simple and therefore fast. This makes them suitable in the practical implementation of interconnection network hardware[44]. Many multi-computer systems, such as the Cosmic Cube, NCUBE, J-machine, iWarp and Intel Paragon, therefore utilize deterministic routers. The most widely used routing algorithms for these machines are the *e-cube* routing algorithm [49], which is used for routing on hypercubes, and dimension order routing, which is used on n -dimensional meshes.

e-cube Routing

In an n -cube with $N = 2^n$ nodes, each node's address is binary coded as $a = (a_0, a_1, \dots, a_{n-1})$. Given a source address $s = (s_0, s_1, \dots, s_{n-1})$ and a des-

destination address $d = (d_0, d_1, \dots, d_{n-1})$ the routing function should determine a route from s to d with a minimum number of steps. Denoting the n dimensions as $i = 1, 2, \dots, n$, where the i th dimension corresponds to the $(i-1)$ st bit in the node address and letting $v = v_{n-1} \dots v_1 v_0$ be any node along the packet route, the route is determined as follows:

1. Compute the direction bit $r_i = s_{i-1} \oplus d_{i-1}$ for all n dimensions ($i = 1, 2, \dots, n$)
2. Start with dimension $i = 1$ and $v = s$
3. If $r_i = 1$, route from the current node v to the next node $v \oplus 2^{i-1}$, else skip this step.
4. Move to dimension $i + 1$ (i.e., $i \leftarrow i + 1$). If $i \leq n$, go to step 3, else quit.

An example of e-cube routing on a 16 node hypercube is presented in Fig. 2.9. In the example $n = 4$, $s = 0000$ and $d = 1101$. Thus $r = r_4 r_3 r_2 r_1 = 1101$. The routing steps are summarized in Table 2.1. As can be seen in the example, the packet is routed from dimension 1 to dimension 4. If the i th bit of s and d are the same, no routing is needed along dimension i . Thus in the example, no routing is required for dimension 2. If the i th bit of s and d differ then the packet is routed from the current node along dimension i . this process is repeated until the destination is reached.

Dimension order Routing

Dimension order routing is somewhat similar to e-cube routing. As was discussed previously, a k -ary n -cube is an n -dimensional cube of radix k ,

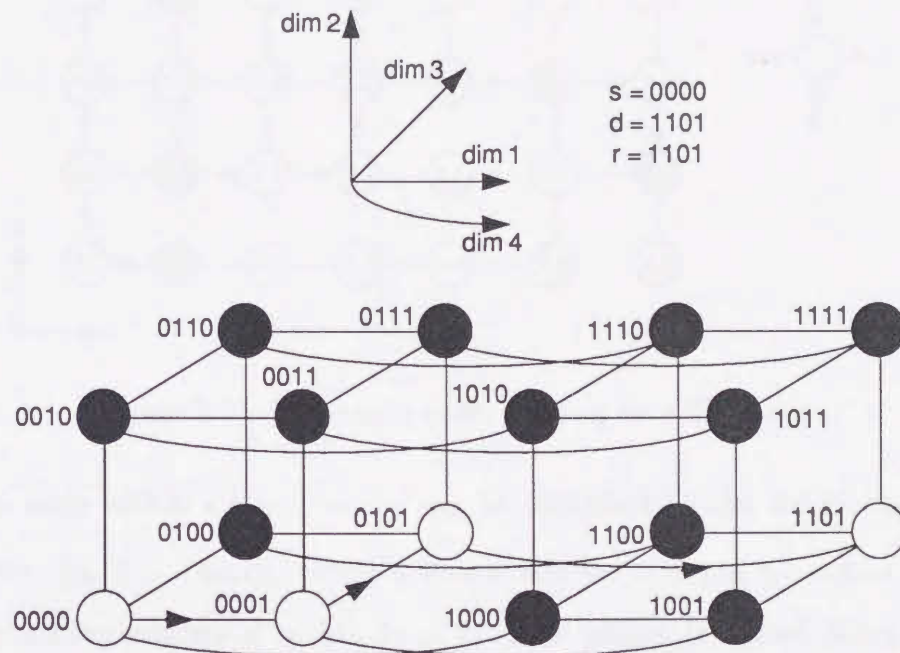


Figure 2.9: e-cube routing on a hypercube

Table 2.1: Routing steps from $s = 0000$ to $d = 1101$

Step	r_i	Operation	Next node
$i = 1$	1	$0000 \oplus 2^0$	0001
$i = 2$	0	skip	n/a
$i = 3$	1	$0001 \oplus 2^2$	0101
$i = 4$	1	$0101 \oplus 2^3$	1101

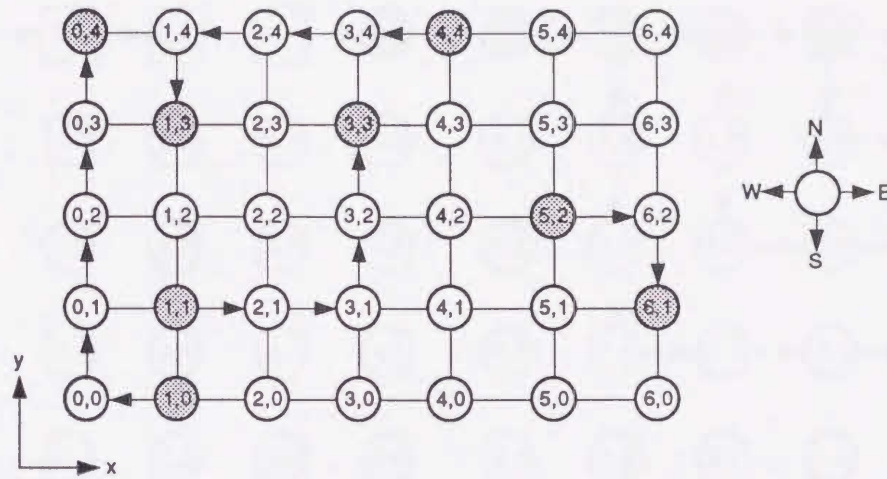


Figure 2.10: Dimension order routing on a 2D mesh

and a node within a k -ary n -cube can be identified by the n -digit radix k address, $(a_0, a_1, \dots, a_{n-1})$. Given a source address $s = (s_0, s_1, \dots, s_{n-1})$ and a destination address $d = (d_0, d_1, \dots, d_{n-1})$, a packet is routed along each dimension $i = 1, 2, \dots, n$, where the i th dimension corresponds to the $(i-1)$ st digit in the node address, until s_{i-1} is equal to d_{i-1} .

This is illustrated in Fig. 2.10, which shows routing between four (source, destination) pairs on a two-dimensional mesh. A packet from any source node $s = (x_1 y_1)$ to any destination node $d = (x_2 y_2)$ will first route along the X-axis until it reaches column y_2 , where d is located. It will then route along the Y-axis until d is reached. A west-north route is taken from node (1,0) to (0,4). An east-north route is traversed from node (1,1) to (3,3). A west-south route is needed from node (4,4) to node (1,3) and an east-south route is required from node (5,2) to node (6,1).

Dimension order routing alone is sufficient to ensure that deadlock does not occur in mesh connected networks, as it prevents a circular wait for

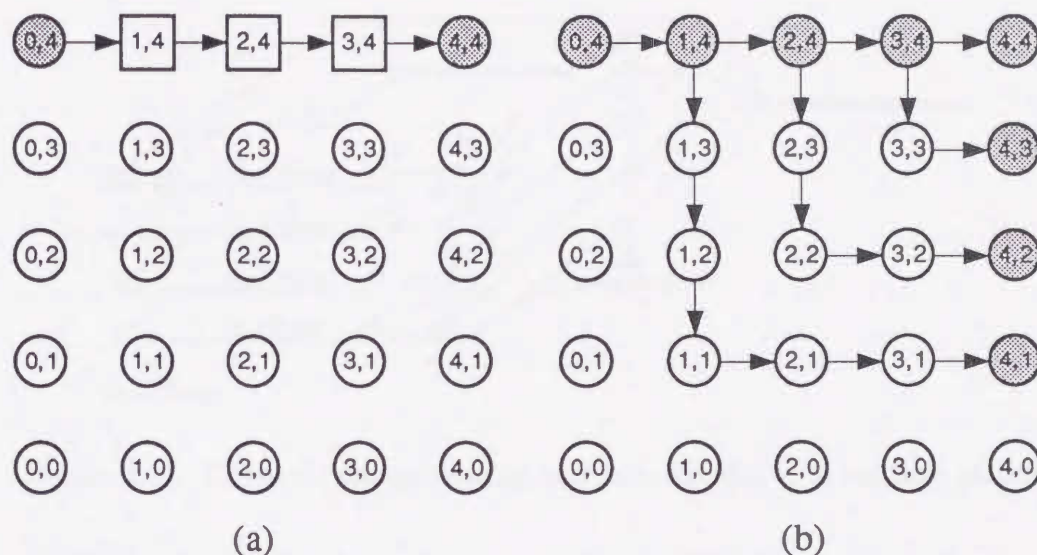


Figure 2.11: (a) Dimension order routing (b) Adaptive routing

channel resources. However, the same dimension ordering scheme will not prevent a deadlock from occurring in a torus network. This is discussed in further detail in Section 2.5

2.4.2 Adaptive Routing

Although deterministic routers are simple to implement and therefore fast, they suffer from poor performance in the presence of hot-spot traffic and are not fault tolerant. Figure 2.11(a) presents a simple example in which dimension order routing may result in poor use of channel resources. Node (0,4) is sending a packet to (4,4), while at the same time node (1,4) has a packet to send to (4,1), node (2,4) has a packet to send to node (4,2) and node (3,4) has a packet to send to node (4,3). As dimension order routing in a two-dimensional mesh requires that the message be sent along the X-axis first, nodes (1,4), (2,4) and (3,4) are unable to send their packets, even though

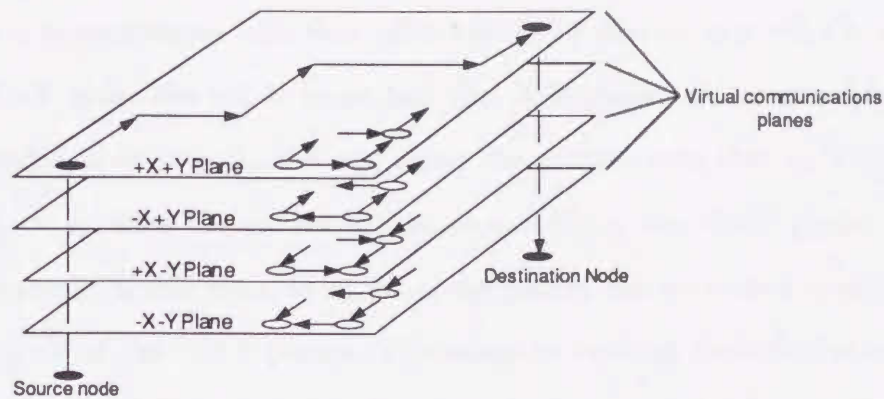


Figure 2.12: Physical communication channels divided into routing planes

a plethora of available channels exist. In Fig. 2.11(b) the routing rules have been relaxed to allow adaptive routing so that the packets from nodes (1,4), (2,4) and (3,4) can be transmitted concurrently with the packet from node (0,4). This allows better channel utilization and lower packet latency.

A number of different approaches have been proposed for the construction of adaptive and fault tolerant routers. Many of these proposals have advocated the use of virtual channels to supply multiple virtual paths between a given (source, destination) pair and thus provide varying degrees of adaptivity and fault tolerance. These include *Planar-Adaptive Routing* [9], *Virtual Networks* [32], *Adaptive Routing with Virtual Channels* [15] and *The Turn Model for Adaptive Routing* [30].

A general technique for providing adaptive routing is to partition the physical network into a number of disjoint subsets, where each subset constitutes a corresponding subnetwork. Packets are routed through different subnetworks depending upon the location of the source and destination nodes.

Figure 2.12 illustrates an application of this method to a 2D mesh. The

network is partitioned into four subnetworks or planes, the $+X+Y$ plane, the $-X+Y$ plane the $+X-Y$ plane and the $-X-Y$ plane. If, for example, the destination node is to the left and above the source node, that is, if $d_x < s_x$ and $d_y > s_y$, then the packet will be routed along the $-X+Y$ plane. If in this example d_x was equal to s_x , then the packet can be routed in either of the $+X+Y$ or the $-X+Y$ planes. This adaptive routing algorithm is said to be *minimal* and fully adaptive, that is, a packet can be delivered through any of the shortest paths between the source and destination. In addition to this, for the 2D mesh it can be proven to be deadlock free. However, providing minimal fully adaptive and deadlock free routing algorithms using this method for the general class of k -ary n -cubes may require additional channels. Linden and Harden [38] have demonstrated that a k -ary n -cube will require 2^{n-1} subnetworks or routing planes and thus the number of channels required increases rapidly with n . The use of virtual channels is also expensive in terms of latency and cycle time[8] and requires that flow control information be sent in the reverse direction to signal the availability of buffering on the receiving node. This flow control information either requires extra wires, or will consume communications bandwidth from the reverse communications channel.

Ngai and Seitz also proposed a non-minimal adaptive mesh router which allows complete freedom of path selection between any (source, destination) pair, by using misrouting to prevent deadlock[41]. However, this approach requires the use of time stamps and prioritization to prevent livelock, requiring that extra state information be stored for each packet and results in a complex router design.

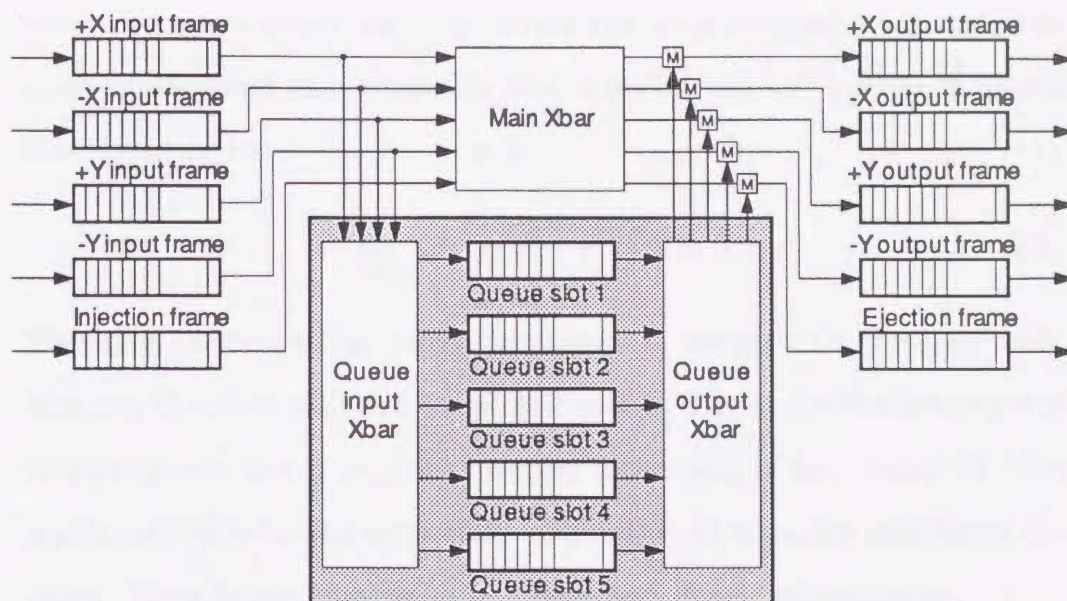


Figure 2.13: Two-dimensional chaos router

Another non-minimal adaptive router which utilizes misrouting to avoid deadlock is the Chaos router proposed by Konstantinidou and Snyder [35]. A block diagram of a two-dimensional router is presented in Figure 2.13. The Chaos router utilizes randomization to provide probabilistic freedom from livelock and therefore does not require any extra state information to make routing decisions. The central queue of Fig. 2.13 is used to store packets which arrive at an input frame and are unable to be routed to an output frame before the entire packet is received. Once the central queue becomes full and a message is specified to be sent to the queue, one of the packets in the queue will be randomly selected and sent to the first available output frame.

Konstantinidou and Snyder have shown that no packet in a router is ever mis-routed with certainty or in other words, every message has a non-zero

chance to avoid misrouting [35]. Using this as a starting point they also demonstrated that the probability that a packet will not have been routed after i routing steps, where $i \rightarrow \infty$ is:

$$\lim_{i \rightarrow \infty} Q(i) = (1 - \epsilon^{\log N})^i = 0 \quad (2.4)$$

Therefore, the longer that a message remains in the network, the more probable that it will be delivered to its' destination. The major disadvantages of this router are that it requires a central misrouting queue, queues at both inputs and outputs, and extra state information to make the misrouting decision. These factors may result in a large and slow implementation.

2.5 Deadlock

Deadlock occurs in an IN of a parallel computer when no packet can advance towards its destination because the queues or channels of the message system are full and no packet can release the queue space that it currently holds. This phenomenon has been studied extensively for wormhole routed networks and a general solution for deadlock avoidance in any wormhole routed network, based on the concept of virtual channels, has been proposed [18]. Deadlock in wormhole routed networks is normally described in terms of a network's *routing function* and *channel dependency graph*.

Definition 4 A *routing function*, $\mathcal{R} : C \times N \rightarrow C$, maps the current channel, C_c , and the destination node, N_d , to the next channel, C_n , on the route from the source node to the destination node. A channel is not allowed to route to itself, $C_c \neq C_n$.

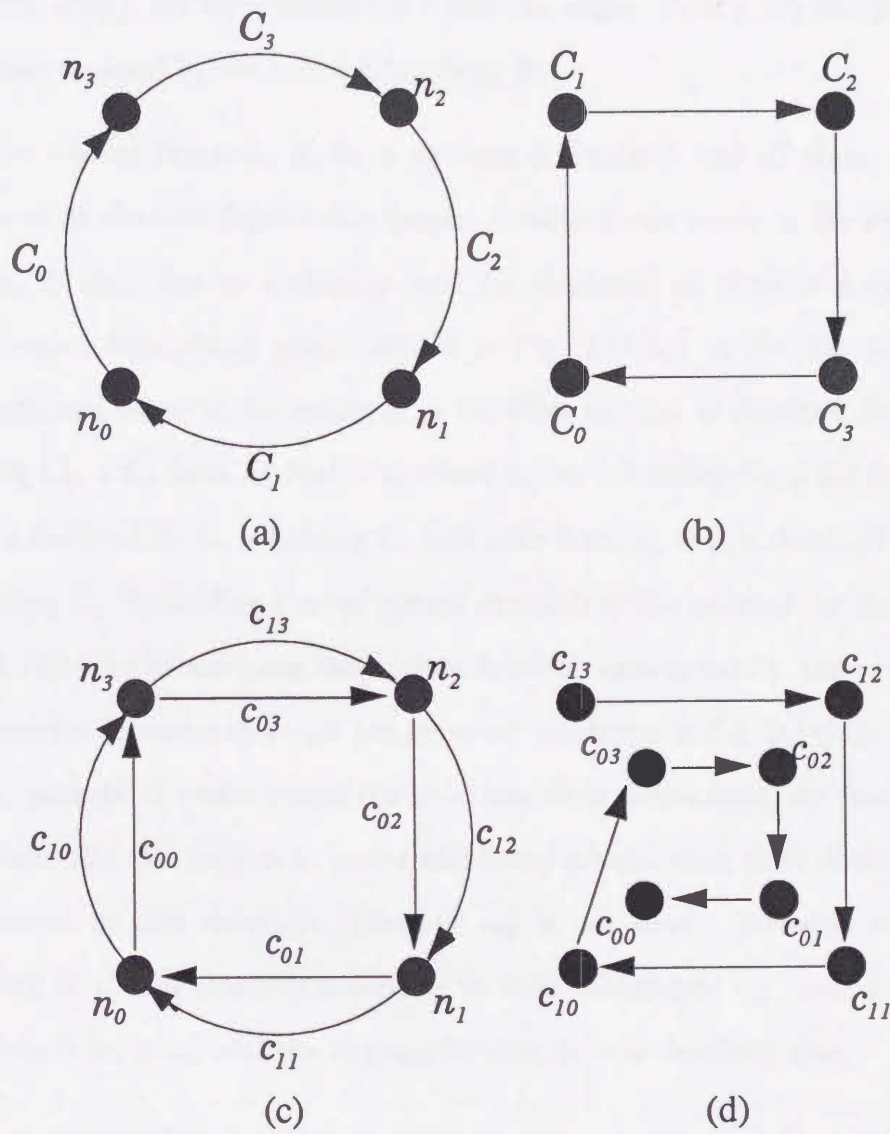


Figure 2.14: (a) Network and (b) its channel dependency graph without virtual channels. (c) Network and (b) its' channel dependency graph with extra virtual channels.

Definition 5 A *channel dependency graph*, D , for an interconnection network, I , and routing function, \mathcal{R} , is the directed graph, $D = G(C, M)$. The vertices, $D(C)$, are the channels of I and the edges, $D(M)$, are the pairs of channels mapped by the routing function, \mathcal{R} .

The routing function, \mathcal{R} , for a network is deadlock free iff there are no cycles in its channel dependency graph. Deadlock can occur in the network of Fig. 2.14(a), due to a circular wait for channels, as there is a cycle in its' channel dependency graph, shown in Fig. 2.14(b). A circular wait for channels can occur if, for example, a flit from n_0 that is destined for n_2 is holding C_0 , a flit from n_3 that is destined for n_1 is holding C_3 , a flit from n_2 that is destined for n_0 is holding C_2 and a flit from n_1 that is destined for n_3 is holding C_1 . By adding a set of virtual channels to the network, as shown in Fig. 2.14(c), and modifying the routing function appropriately, the cycles in the channel dependency graph are removed, as shown in Fig. 2.14(d). In the figure, packets at nodes numbered less than their destination are routed on high channels and packets at nodes numbered greater than their destination are routed on low channels. Channel c_{00} is not used. There is now an ordering of virtual channels according to their subscripts: $c_{13} > c_{12} > c_{11} > c_{10} > c_{03} > c_{02} > c_{01}$ and the routing function is now deadlock free.

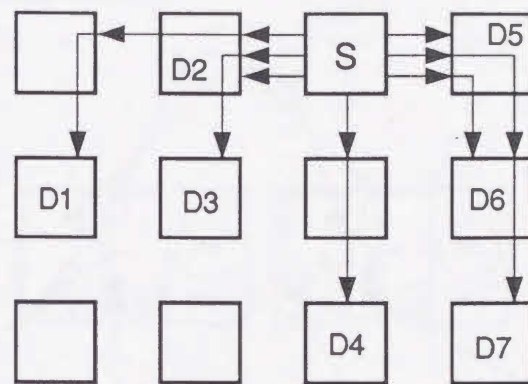
2.6 Multicast Messages

Point to point, or unicast communication, in which a source node sends a message to a single destination node, is the basic structure supported by present multicomputers. Broadcast and multicast communications are the

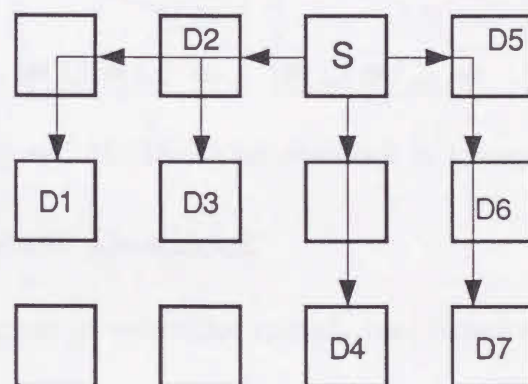
transmission of a message from a source node to all other nodes in the system, and from a source node to a subset of the nodes in a system respectively. Broadcast communication can be viewed as a special case of a multicast communication, in which the same message is delivered to all of the nodes in the system [40].

Two parameters commonly used to measure the efficiency of multicast schemes are *channel traffic* and *communication latency*. Channel traffic is defined as the number of channels used to deliver the message under consideration and latency is defined as the longest packet transmission time involved. These two parameters are somewhat interrelated as is illustrated in Fig. 2.15. The unicast based multicast generates traffic = 14 and has distance = 3, the tree based multicast has traffic = 9 and distance = 3 and the path based multicast has traffic = 7 and distance = 4.

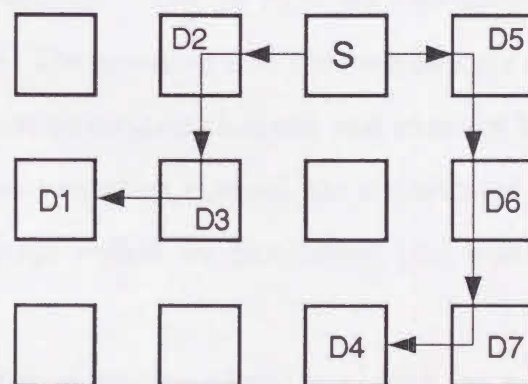
Multicast communications can be implemented using multiple unicasts, software multicast trees, or by hardware multicast facilities. Multiple unicasts, while simple to implement, generate large amounts of unnecessary traffic which can cause blocking and contention in the network [37]. Software multicast trees, in which a worker node will forward the multicast message to its neighbors upon reception of the message, exhibit considerable speedup when compared to multiple unicasts [51], but are still inferior to hardware based multicast schemes. Although hardware based multicast schemes offer the best potential performance for the implementation of multicasting, it has been shown that these schemes may result in deadlock in those networks which employ wormhole routing [37].



(a)



(b)



(c)

Figure 2.15: (a) Multicast by unicast (b) Tree based multicast (c) Path based multicast

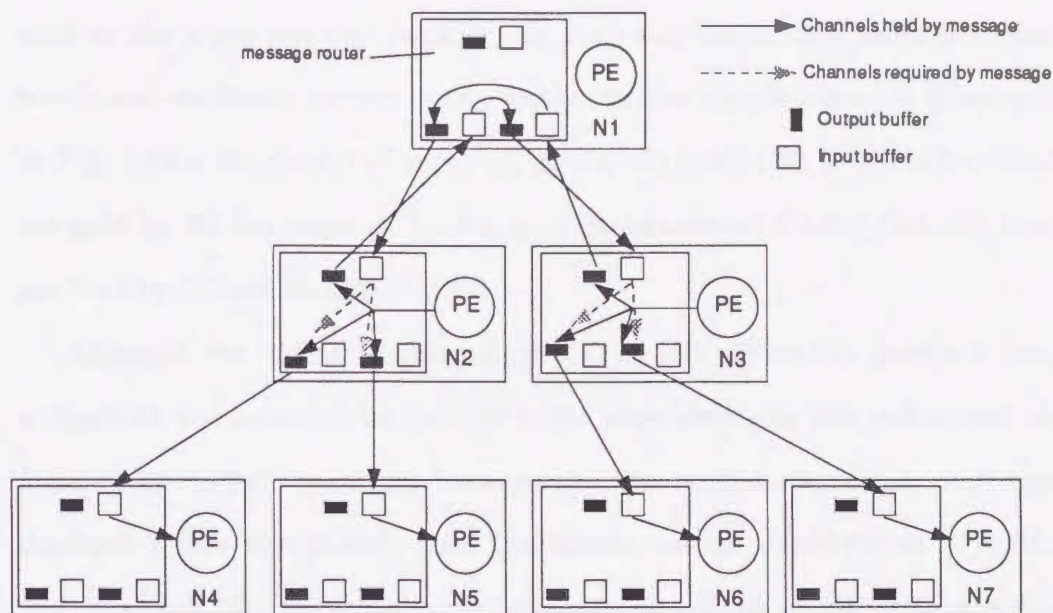


Figure 2.16: Multicast deadlock in binary tree

2.6.1 Multicast Deadlock

One of the properties of wormhole routed, tree based multicast schemes is that, due to the small amount of buffer space at each node, a potentially large number of network resources must be concurrently held by a single multicast message. The resources that the messages are competing for in the network are the communication channels and message buffers of each node. Each physical communication channel has a dedicated message buffer and typically the message buffers are partitioned into separate virtual channel buffers [13].

While a number of routing algorithms, such as *e-cube* routing in hypercubes and *dimension order* routing in meshes, guarantee deadlock free routing of unicast messages, multicast trees based on these algorithms are prone to deadlock. In fact, networks which are inherently free of deadlock,

such as the *n*-ary tree and fat tree [36], may also deadlock if more than one tree based multicast occurs concurrently. In the simple example presented in Fig. 2.16 a deadlock has occurred as the channels (N3,N6),(N3,N7) that are held by N3 are required by N2, and the channels (N2,N4),(N2,N5) that are held by N2 are required by N3.

Although the unicast routing algorithm of this network is deadlock free, a deadlock has occurred because of *cyclic dependency* in the *concurrent allocation of multiple resources* between the two multicasts. Thus, multicast deadlock differs significantly from traditional unicast deadlock, as in multicast deadlock, the resources contributing to the deadlock situation are distributed over a number of nodes. Traditional methods of deadlock avoidance, such as releasing all of the deadlocked resources once deadlock is detected or requesting all of the required resources prior to initiating an operation which might result in deadlock, are not suitable for prevention of multicast deadlock. Releasing the distributed deadlocked resources results in considerable waste of communications bandwidth and may be difficult to implement due to the large number of distributed resources which may need to be released, while requesting all of the necessary channels prior to initiating a multicast would significantly increase the multicast latency. New methods of deadlock avoidance for multicast must therefore be found.

Multicast deadlock avoidance has typically been achieved by limiting the growth of the multicast tree and Lin, McKinley, and Ni have extensively studied the use of multi-path multicasting algorithms utilizing Hamiltonian paths to ensure that deadlock does not occur [40, 37, 51]. In addition to deadlock avoidance, multi-path multicast allows arbitrary multicast destinations

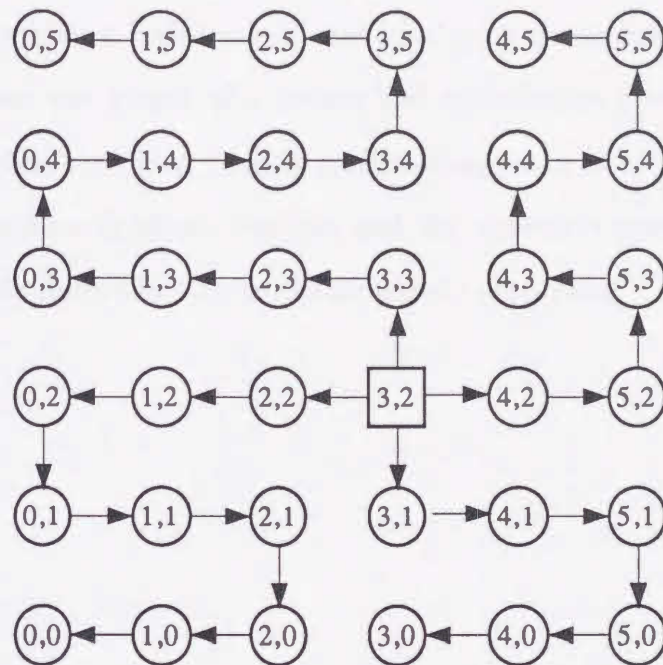


Figure 2.17: Multipath multicast

and they have demonstrated that this technique has the added advantage of reducing the amount of traffic in the network. Figure 2.17 illustrates a multi-path broadcast in an 6×6 mesh network. As can be seen in Fig. 2.17, a multi-path message is broadcast by sending four copies of the message on individual multicast paths. Similarly, Byrd et al. have investigated the restricted branch multicast approach to multicasting [7]. This approach requires that a multicast message can only be split into two paths at any given node, and that one of these paths must be connected to the local processing element.

Multi-path and restricted branch multicasting have a number of disadvantages. For example both restricted branch and multi-path multicasting require that the packet header store multiple destination addresses, as all of

the destinations for a broadcast or multicast must be stored in the header, which increases the length of a packet and complicates router design. In addition to this, restricted branch multicasting requires an extra port resource to guarantee deadlock freedom, and the algorithm used in multi-path multicasting to determine the multicast paths is complex.

Chapter 3

Tokkyū: A High-Performance, Randomizing, Adaptive Message Router with Packet Expressway

The Tokkyū router is a new high-performance message router for k -ary n -cube multicomputer systems[26, 29, 28]. The k -ary rings that make up the interconnection network are constructed using uni-directional register-insertion buses. Tokkyū utilizes misrouting to prevent deadlock and randomization to prevent livelock in a fully adaptive routing environment. Any packet arriving at an input to a Tokkyū router that can not be profitably routed is immediately misrouted. This is significantly different than both the Ngai/Seitz router and the Chaos router which defer the misrouting of a packet that is waiting for an output until it is to be overwritten by a newly arriving packet. The misrouting rate is minimized by utilizing a small number of queues, placed at the outputs of the communication ports. As blocking or buffering flow control is not used, all of the available communications bandwidth can be utilized for sending messages between processors in the system. Finally,

uncongested network performance is improved by the inclusion of the *packet expressway*, which provides a low latency bypass path for packets which need not pass through the core of the router.

3.1 The Register-insertion Bus

High performance ring buses have become a favorable alternative in the implementation of local area networks [45]. However, LAN/WAN structures are not directly applicable to INs due to differences in the node structure and communications patterns [15]. The use of the unidirectional register-insertion bus in the construction of INs does, however, have a number of advantages. These advantages include:

- A packet may propagate through a large number of bus interfaces without being buffered.
- Processors are free to inject packets at any time, subject to available space in the transmit queue. Thus there is no global arbitration, as each processor can decide whether to inject a packet according to information local to its bus interface.
- Active repeaters can be used at the output of each message router, instead of the pulldown structure required for a bi-directional bus, thus making the network more scalable.

3.1.1 Register-insertion Bus Operation

With reference to Fig 3.1 the operation of a register-insertion bus is as follows; Assume that the input and output data is synchronized at the same

transmission rate, so that for each word received, another can be transmitted. The transmit (tx.) buffer is used to temporarily store a packet from the local processor while it is waiting for injection onto the bus. These packets are of variable length and so only a portion of the tx. buffer may be used for a particular packet, however, the packet length must not exceed the length of the tx. buffer. The function of the delay buffer can be explained by first considering the area currently being used. The used, or active portion of the delay buffer, operates as a FIFO queue that delays the incoming packets. Assuming that the entire delay buffer has a capacity of n words and that i words are currently used, $1 \leq i \leq n$, then $n - i$ words remain for the unused or inactive portion. Thus locations w_0, w_1, \dots, w_{i-1} of the delay buffer are active and locations $w_i, w_{i+1}, \dots, w_{n-1}$ are inactive. If, in each time step t , a new word can be received, and a new word is to arrive at time $t + 1$, then the active portion of the delay buffer represents a FIFO queue containing the words which arrived at times $t, t + 1, \dots, t + (i - 1)$. At time $t + 1$ the word stored in w_0 is removed from the queue and sent to the output. Simultaneously, the incoming word is added to the queue such that locations w_0, w_1, \dots, w_{i-1} now contain data which arrived at times $t + 1, t + 2, \dots, t + i$, and the queue length remains unchanged.

It is desirable that in each time step, if $i \geq 1$, the queue size be reduced. A reduction can take place iff the data received at the input is not part of any packet destined for the output. In this case, the previous discussion should be modified so that the incoming word is not stored in location w_{i-1} and also so that i is reduced to $i' = i - 1$. Furthermore, if $i = 0$, then any incoming word need not be stored at all and can pass directly to the output.

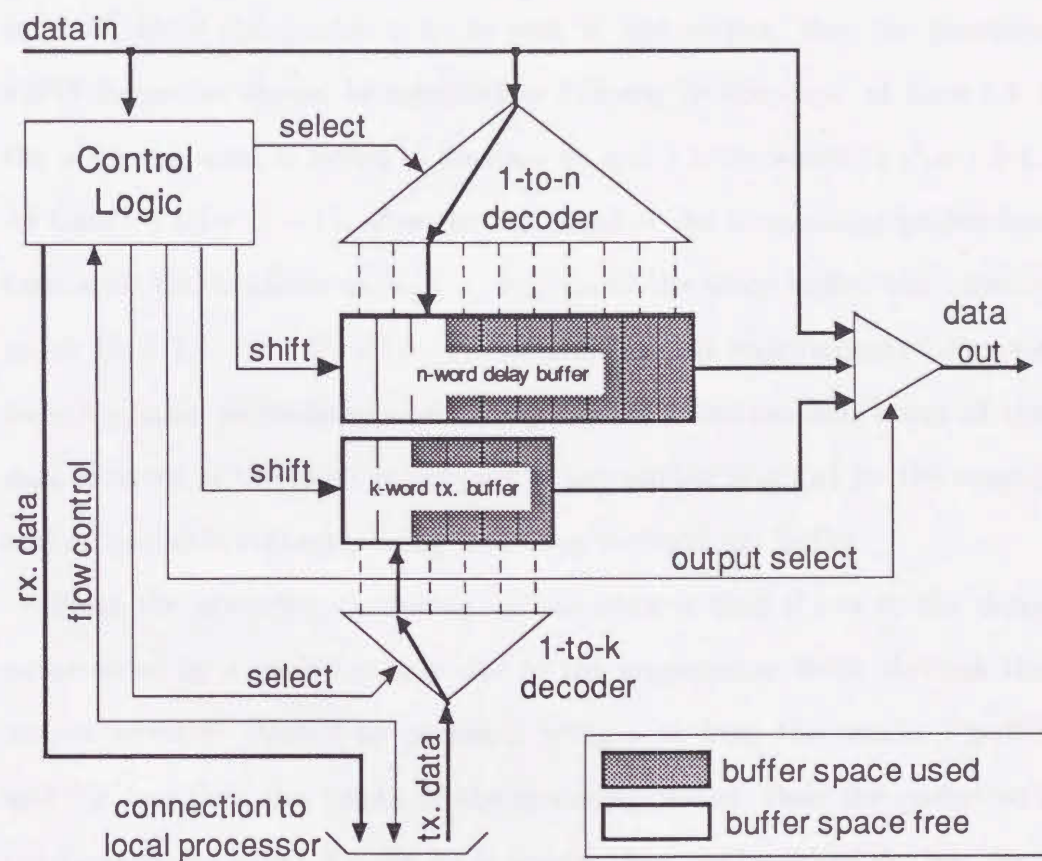


Figure 3.1: Register-insertion bus interface

In this case the incoming word is not stored in location w_{i-1} and i is constant at $i = 0$.

The inactive portion of the buffer is essential for the injection of packets into the network from the tx. buffer. Assuming that the tx. buffer contains a packet of length l , $1 \leq l < (n - i)$, and that at time $t + i$ the first word of the of this packet is to be sent to the output, then the previous FIFO discussion should be modified as follows; In this case, at time $t + i$ the incoming word is stored in location w_i and i is increased to $i' = i + 1$. At time $t + ((i + l) - 1)$, after the last word of the transmitted packet has been sent, the locations $w_0, w_1, \dots, w_{(i+1)-1}$ of the delay buffer now contain words $t, t + 1, \dots, t + ((i + l) - 1)$. In addition, the requirements for queue reduction must be modified such that queue reduction can only occur iff the data received at the input is not part of any packet destined for the output and no packet is currently being sent from the local tx. buffer.

From the preceding discussion we can observe that if $i = 0$, the delay experienced by a packet is only due to the propagation delay through the output selector. Also if no packet is being sent from the transmit buffer and i is less than the length of the incoming packet, then the packet will cut-through the FIFO. Finally if i is greater than the length of the incoming packet, or a packet is being transmitted and l is greater than the length of the incoming packet, then the incoming packet will be completely buffered in the FIFO, in a store-and-forward manner.

The concept of the register-insertion bus can easily be extended to the k -ary n -cube as is shown in Fig. 3.2, which illustrates the structure of a single port of an n -dimensional register-insertion bus router. The delay buffer of

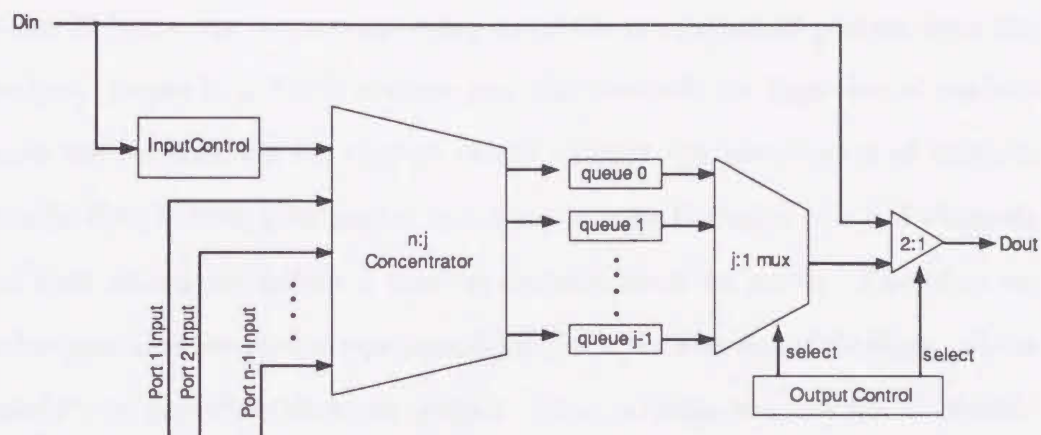


Figure 3.2: N -dimensional register-insertion bus port.

Fig 3.1 is replaced by a group of output buffers. These buffers store packets that are changing dimensions, in addition to those which must be delayed while the local processor injects new packets into the network. Also, the control is now distributed between the input and output control sections to improve performance.

3.2 Architecture of the Tokkyū Router

The architecture of a two-dimensional Tokkyū router is presented in Fig. 3.3. The input queues of a typical oblivious router have been replaced by m queues per output and $n : m$ switches connect the inputs to the queues, where $n = 4$ for a two-dimensional router. A small input frame is also provided in each input controller to temporarily store several words of an incoming packet while a routing decision is made. Each of the output queues is capable of holding multiple, variable length packets and all of the queues support cut-through routing. As the router may buffer complete packets when output contention occurs, it requires the use of comparatively short packets, i.e. less

than 32 bytes. An output controller schedules the output of packets from the output queues in a FIFO manner and also controls the injection of packets into the network via the output switch. Under the assumption of uniform traffic distribution, each packet in a k -ary n -cube traverses $\sigma = k/4$ channels in each dimension before a routing decision *must* be made. Therefore we have provided the *packet expressway* which, in the absence of blocking, allows packets to pass directly to an output. Thus, a single unidirectional channel in any dimension can be viewed as a high speed register-insertion ring[26]. The header of each packet is updated prior to entering the output register, when passing through the *inc* or *dec* modules, to reflect the progress of the packet through the network.

As misrouting is used to prevent deadlock and randomization is used to prevent livelock, correct operation of the router can be guaranteed provided no packet, or part of a packet, is lost due to buffer overflow. The aggregate data rate into any router must therefore never exceed the aggregate data rate out of the router. A simple way for the data rates within the network to remain tightly matched is through the use of a globally distributed clock. Then, by restricting packet injection to only occur when sufficient space exists to completely store any packet which may arrive while injection is taking place, buffer overflow is guaranteed not to occur.

3.2.1 Router Operation

The operation of the router can be understood by examining the control algorithms of its major components. These components are the input and the output controllers of each port, the queue controller associated with each

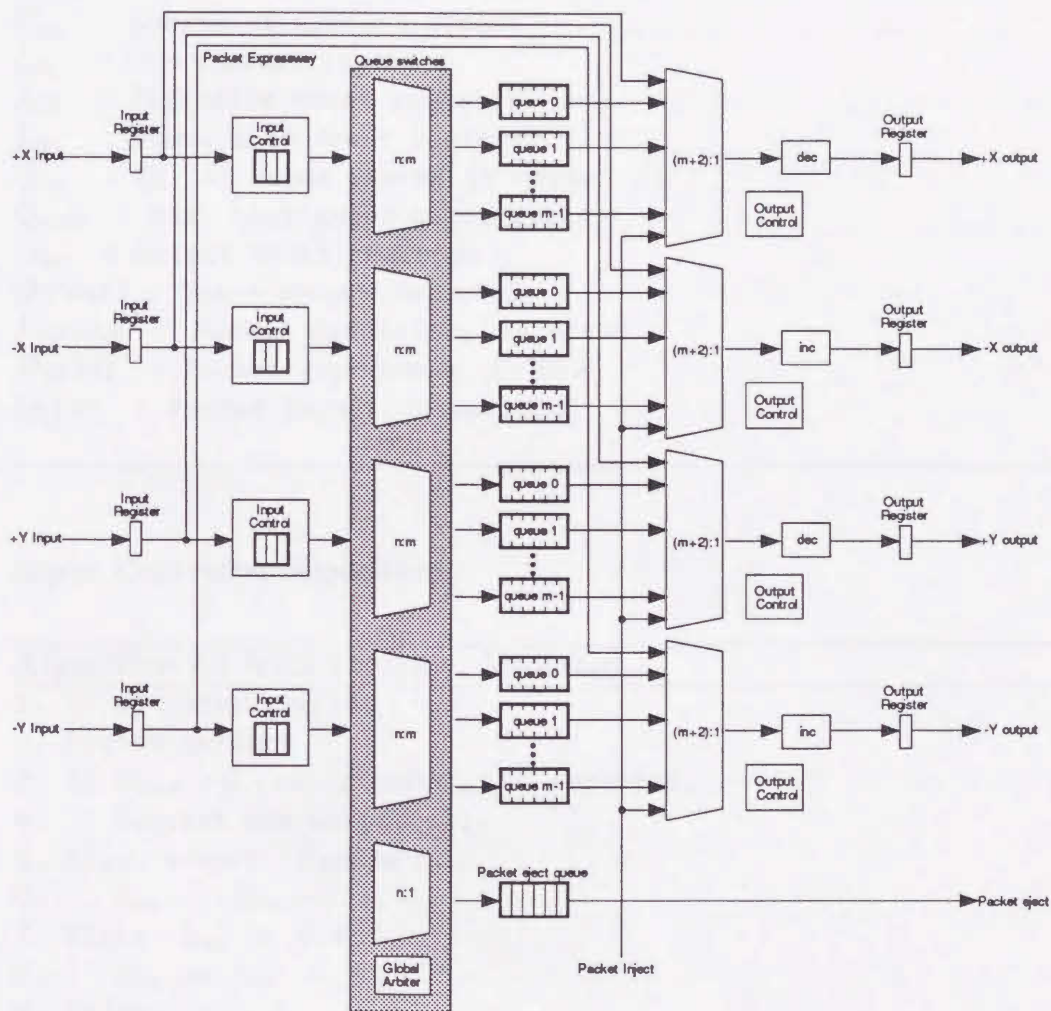


Figure 3.3: Architecture of a two-dimensional Tokkyū router

output queue and the arbiter which controls access to the output queues via the queue switches. Throughout this section the following notation is adopted for convenience:

D_{rem}	: Distance remaining in this dimension
P_{len}	: Length of current packet
I_{len}	: Input count register
J_{len}	: Injection count register
L_{len}	: Queue load count register
Q_{len}	: No. of words stored in queue
Q_{max}	: Max. contiguous queue space
O_{len}	: Output count register
<i>Output</i>	: Queue output selected
<i>Passive</i>	: <i>Packet expressway</i> selected
<i>Bypass</i>	: <i>Packet expressway</i> in use
<i>Inject</i>	: Packet injection selected

Input Controller Algorithm

Algorithm 3.1 Input Controller Algorithm

1. If no packet, wait;
 2. Decode header;
 3. If $D_{rem} = 0$ or *Passive* not asserted,
 4. Request new output(s);
 5. Else, assert *Bypass* ;
 6. $I_{len} = P_{len} - 1$;
 7. While $I_{len} > 0$ do
 8. $I_{len} = I_{len} - 1$;
 9. Enddo;
 10. Reset *Bypass* ;
 11. Goto 1;
-

With reference to Algorithm 3.1 the input controller operation is as follows;
The received data is sampled by the input controller on each clock cycle to

test for a valid packet header. Upon the detection of the first word of a packet, the header is decoded to generate the output request(s). A packet which is j -dimensions from its destination will generate j valid output requests. If the packet has finished traversing the current dimension ($D_{rem} = 0$) or the output switch is not in the *Passive* state, then the output request(s) will be passed to the global arbiter. *Bypass* is asserted if $D_{rem} \geq 1$ and the output switch is *Passive*, to signal that the packet is passing to the output via the *packet expressway*. The packet length is loaded into the input count register and on each subsequent clock cycle I_{len} is decremented as each new word of the packet is received. Once I_{len} has decremented to zero, indicating that the entire packet has been received, *Bypass* is reset and the input controller begins to sample the input for a valid header once again.

Output Controller Algorithm

With reference to Algorithm 3.2 the output controller operation is as follows; Operation of the output controller begins with setting the output switch to the *Passive* state, allowing any packet on the *packet expressway* to pass directly to the output register. Once an output request is detected and no packet is currently bypassing the output, the request is processed and the output switch is set accordingly. If an injection request is being made and there exists sufficient space for any incident packet to be temporarily stored while the new packet is being injected ($Q_{max} \geq J_{len}$), then the switch is set to the injection input. This ensures that there always exists sufficient space to buffer an arriving packet within the node while a new packet is injected so that no information, i.e. no part of a packet, is lost. The packet length

Algorithm 3.2 Output Controller Algorithm

```
1. Assert Passive;
2. If no output requests, wait;
3. If Bypass is asserted, wait;
4. Reset Passive;
5. While output requests do
6.   If injection request,
7.     If  $Q_{max} \geq J_{len}$ ,
8.       Assert Inject;
9.   If Inject not asserted and output request,
10.    Assert Output;
11.    Get first output request;
12.     $O_{len} = P_{len}$ ;
13.    While  $O_{len} > 0$  do
14.      Output word;
15.       $O_{len} = O_{len} - 1$ ;
16.      If Output asserted,  $Q_{len} = Q_{len} - 1$ ;
17.      Else,  $J_{len} = J_{len} - 1$ ;
18.    Enddo;
19. Enddo;
20. Goto 1;
```

is loaded into the output count register and a new word of the packet being output is placed in the output register during each clock cycle. O_{len} and either of Q_{len} or J_{len} are decremented until the entire packet has been sent.

Global Arbiter Algorithm

Algorithm 3.3 Global arbiter algorithm

1. If no requests, wait;
 2. While requests do
 3. Get first request;
 4. If requested output(s) free,
 5. Assign available queue;
 6. Else, Assign random queue;
 7. Enddo;
 8. Goto 1;
-

With reference to Algorithm 3.3 the output controller operation is as follows; The global arbiter processes each output request sequentially, beginning with the request at the head of the request queue. The arbiter examines the output request and the current state of the queue switches and the output queues in an attempt to profitably route the requesting packet. If it is not possible to profitably route the packet, it will be randomly misrouted to any available output queue. Although it may appear that this approach of immediately misrouting blocked packets will result in excessive misrouting of packets, the discussion in Sect. 3.3 and the simulation results of Sect. 3.4 demonstrate that the careful selection of the switch and output queue sizes prevents this from occurring.

The arbiter algorithm presented here processes each input sequentially. At first glance it might appear that it would be beneficial to process all

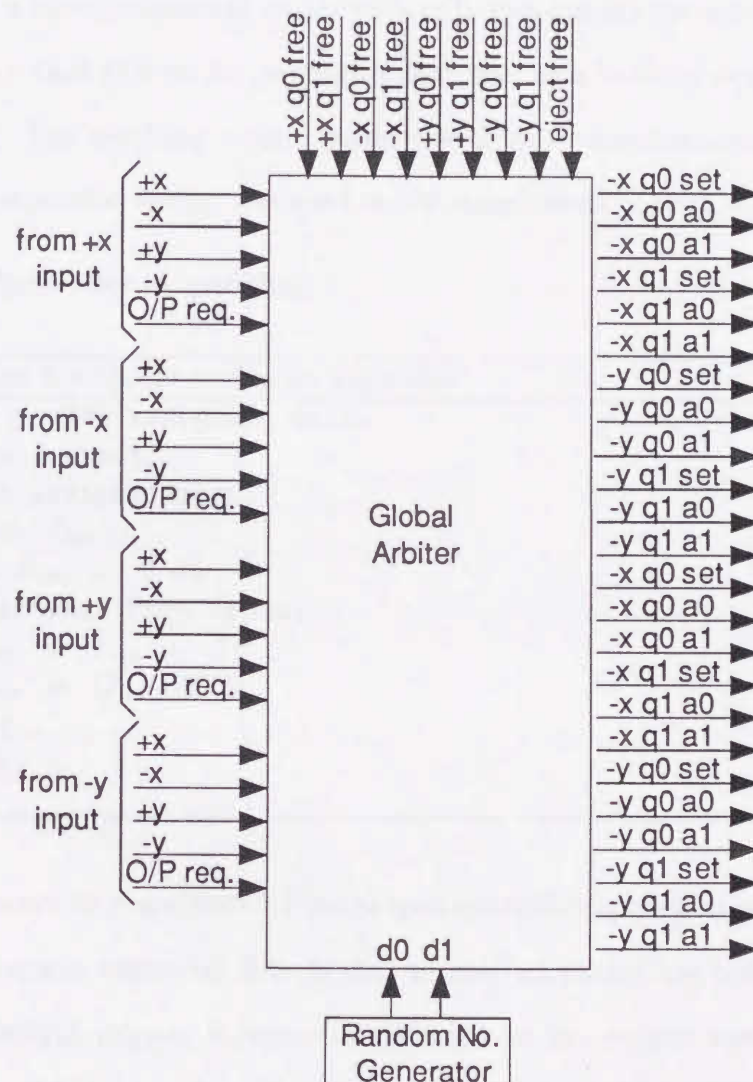


Figure 3.4: Global arbiter inputs and outputs

of the inputs simultaneously using a large combinatorial logic circuit, as this may result in shorter average time to make routing decisions. However, with reference to Fig. 3.4 which presents the inputs and outputs for the arbiter section of a two-dimensional router with only two queues per output port, it can be seen that this would require the solution to a boolean equation with 31 inputs. The resulting circuit would therefore be cumbersome and slow, and so a sequential design was used in the simulations of Sect. 3.4.

Queue Controller Algorithm

Algorithm 3.4 Queue controller algorithm

1. If no packet assigned, wait;
 2. Request output;
 3. Select assigned port;
 4. $L_{len} = P_{len}$;
 5. While $L_{len} > 0$ do
 6. Load word from input;
 7. $L_{len} = L_{len} - 1$;
 8. $Q_{len} = Q_{len} + 1$;
 9. Enddo;
 10. Goto 1;
-

With reference to Algorithm 3.4 the output controller operation is as follows; When the queue controller detects that a received packet has been assigned to it, an output request is immediately made to the output controller and the length of the packet from the assigned port is loaded into the queue load count register (L_{len}). A new word of the packet is loaded into the queue in each clock cycle, (L_{len}) is decremented and the count of the number of words currently stored in the queue (Q_{len}) is incremented, until the entire packet has been received ($L_{len} = 0$).

3.3 Switch and Buffer Design

The misrouting of packets provides a simple solution to the problem of deadlock. However, any packets which are misrouted will remain in the network, requiring channel and buffer resources. This may exacerbate any existing congestion and result in further misrouting. It is therefore desirable that the output switch and buffer sizes be selected so that under normal operation there is a minimal amount of misrouting occurring. Karol *et al* [33] and Yeh *et al* [52] have studied in detail the design and performance of systems employing output queues. However, their analyses have focused on those systems in which an arriving packet can only select one possible output from those available, and where the number of inputs, n , approaches infinity. We extend their work here by examining the switch and buffer requirements for those cases in which an arriving packet may select from a number of outputs, and we focus on small values of n , typically 4 or 6. To simplify the following discussion we assume that all packets are of fixed size.

3.3.1 Switch Evaluation

Assume that fixed size packets arrive at the n inputs to the k -ary n -cube router. In each time slot, packet arrival is governed by independent and identical Bernoulli processes and packets arrive independently at each input with probability ρ . Under the assumption of uniform random traffic in a k -ary n -cube, on average, each packet must traverse $\sigma = k/4$ channels in each dimension and the average distance of a packet, d_{ave} , is $(n \times \sigma)$. Of the arriving packets, $1/d_{ave}$ are destined for the local processor and therefore the

probability that an arriving packet is destined for one of the queue switches associated with an output, which we define as α , is equal to $\rho - (\rho/d_{ave})$. The probability of i packets arriving at the router inputs, all destined for a single output queue switch, a_i^* , has the binomial probabilities

$$a_i^* = \binom{n}{i} \left(\frac{\alpha}{n}\right)^i \left(1 - \frac{\alpha}{n}\right)^{n-i} \quad (3.1)$$

$$i = 0, 1, 2, \dots, n$$

If the probability of misrouting is very low then most arriving packets will be profitably routed, i.e. routed towards their destinations. Arriving packets are therefore equally likely to be destined for only $n - 1$ of the available outputs, as the n^{th} output will send the packets in the opposite direction to which they have just travelled, and thus Eq. 3.1 becomes

$$a_i = \binom{n-1}{i} \left(\frac{\alpha}{n-1}\right)^i \left(1 - \frac{\alpha}{n-1}\right)^{n-1-i} \quad (3.2)$$

$$i = 0, 1, 2, \dots, n-1$$

Packets arriving at the n router inputs to the k -ary n -cube must compete for access to the m queues associated with each output, via the queue switches. If i packets arrive at the inputs at the same time, all destined for the same output, and $i < m$, then all requests can be satisfied by the switch. If $i > m$, then $i - m$ requests will be rejected and these packets will have to be misrouted. It follows then that the probability of an output request being unsuccessful, for the case where a packet can be successfully routed via only

one output, is given by the sum of the probabilities of $i > m$

$$\Pr(M_{j=1}) = \frac{1}{\alpha} \sum_{i=m+1}^{n-1} \left[(i-m) \binom{n-1}{i} \cdot \left(\frac{\alpha}{n-1} \right)^i \left(1 - \frac{\alpha}{n-1} \right)^{n-1-i} \right] \quad (3.3)$$

Extending Eq. 3.3 to the case where a packet can be profitably routed via more than one output: If i packets arrive at the router inputs at the same time, each of which can be profitably routed via j outputs, and $i \leq jm$, then all of the requests can be satisfied by the switches. If $i > jm$, then $i - jm$ requests will be rejected and these packets will have to be misrouted. The probability that $i > jm$ is

$$\Pr(i > jm) = \frac{1}{\alpha} \sum_{i=jm+1}^{n-1} \left[(i-jm) \binom{n-1}{i} \cdot \left(\frac{\alpha}{n-1} \right)^i \left(1 - \frac{\alpha}{n-1} \right)^{n-1-i} \right] \quad (3.4)$$

In order to evaluate the effect of allowing packets to request more than one output, we need to determine β_j , the fraction of arriving packets with j dimensions still to traverse, where $0 \leq j \leq n$. To calculate β_j , we need to determine the distance distribution for newly generated packets. This is given by the number of ways in which the n -tuple describing the total distance to travel in each dimension, $(c_0, c_1, c_2, \dots, c_{n-1})$, can be arranged so that the sum $c_0 + c_1 + c_2 + \dots + c_{n-1}$ is equal to the distance to travel, d_g , where $0 \leq c_l \leq k/2$ for all $l = 0, 1, 2, \dots, n-1$. The number of solutions for the equation $c_0 + c_1 + c_2 + \dots + c_{n-1} = d_g$, which we define as Ψ_{d_g} , is given by the coefficient of x^{d_g} in the generating function, $f(x) = (1 + x + x^2 + \dots + x^{k/2})^n$,

Table 3.1: 2-tuples defining total distance to travel and Ψ_{d_g} for packets in an 8-ary 2-cube

d_g	$j = 1$	$j = 2$	Ψ_{d_g}
1	(0,1)(1,0)	-	2
2	(0,2)(2,0)	(1,1)	3
3	(0,3)(3,0)	(1,2)(2,1)	4
4	(0,4)(4,0)	(1,3)(3,1)(2,2)	5
5	-	(1,4)(4,1)(2,3)(3,2)	4
6	-	(2,4)(4,2)(3,3)	3
7	-	(3,4)(4,3)	2
8	-	(4,4)	1

$0 < d_g \leq nk/2$. Table 3.1 shows the distance distribution of newly generated packets, their corresponding 2-tuples and Ψ_{d_g} for an 8-ary 2-cube.

If P_{d_r} is the probability that a packet is at a distance, d_r , from its destination when it arrives at the input to a router and $P_{(d_g, d_r, j)}$ is the probability of that packet having j dimensions still to traverse, given that it started with a distance to travel of d_g , then β_j is given by

$$\beta_j = \sum_{d_g=1}^{\frac{nk}{2}} \sum_{d_r=0}^{d_g-1} P_{d_r} P_{(d_g, d_r, j)} \quad (3.5)$$

where

$$P_{d_r} = \frac{\Psi_{d_r}}{\sum \Psi_{d_g}} \quad (3.6)$$

and $P_{(d_g, d_r, j)}$ can be determined by considering the state transition diagram for the distance distribution of a given k-ary n-cube. Fig. 3.5 shows the state diagram used for determining distance distribution, P_{d_r} , in an 8-ary 2-cube. The vertices in the figure are the 2-tuples representing the (x, y) distances to travel and the arcs are the probabilities of a transition from distance (x, y) to distance (x', y') .

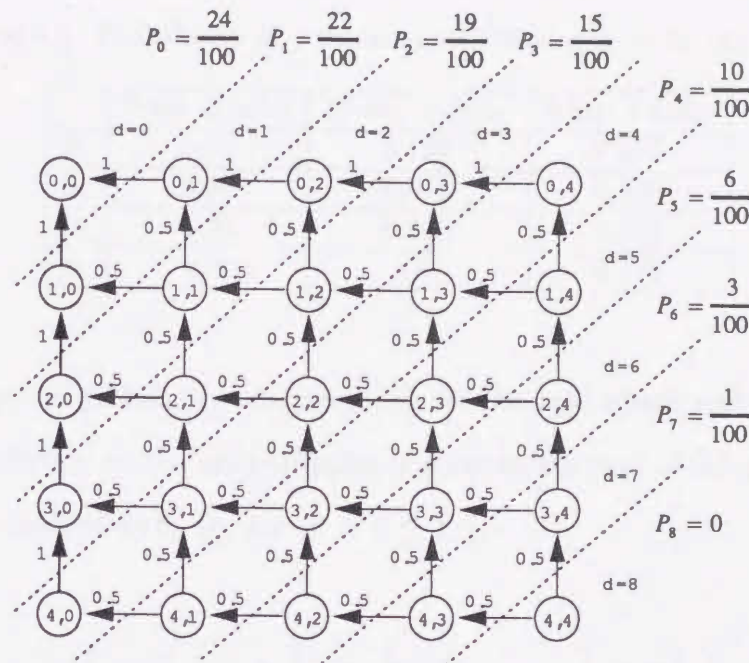


Figure 3.5: State diagram for determining the distance distribution in an 8-ary 2-cube

Using Eq. 3.5 we can predict the probability that a packet arriving at an input has j dimensions still to traverse. This is important since the probability that a packet will be misrouted due to contention for a queue or switch decreases if j is greater than 1, and thus the size of the switches can be reduced if β_j is large for values of j greater than 1. Table 3.2 presents the probabilities of β_j , $0 \leq j \leq 3$, for a 64 node 8-ary 2-cube, a 256 node 16-ary 2-cube and a 512 node 8-ary 3-cube. As can be seen in the table, the probability that a packet can request 2 or more outputs is approximately 29% for the 8-ary 2-cube, 46% for the 16-ary 2-cube and 53% for the 8-ary 3-cube. We can therefore conclude that, under the assumption of uniform traffic, as the radix or dimension of a network is increased, the probability of misrouting due to queue or switch contention decreases.

Table 3.2: Probability of j dimensions remaining to be traversed

	8-ary 2-cube	16-ary 2-cube	8-ary 3-cube
β_0	0.25	0.125	0.167
β_1	0.46	0.413	0.303
β_2	0.29	0.462	0.351
β_3	n/a	n/a	0.179

Finally, the probability of misrouting, for the case where arriving packets can be profitably routed via j outputs, is given by the sum of the probabilities of $i > jm$ multiplied by β_j , for all of $0 \leq j \leq n$

$$\Pr(M_{j>1}) = \sum_{j=1}^n \left[\beta_j \frac{1}{\alpha} \sum_{i=jm+1}^{n-1} (i - jm) \binom{n-1}{i} \cdot \left(\frac{\alpha}{n-1} \right) \left(1 - \frac{\alpha}{n-1} \right)^{n-1-i} \right] \quad (3.7)$$

Applying Eq. 3.7 we can evaluate how the rate of misrouting increases as the load applied to a router increases. Figure 3.6 illustrates how the predicted probability of misrouting varies as a function of the applied load for a single node in a 16-ary 2-cube, along with results obtained by simulation. As can be seen in the figure, the predicted results and simulated results remain in close agreement, indicating that our model is suitable for predicting the switch performance in networks where multiple outputs are available for routing. Although the results of Fig. 3.6 are useful in quantifying the amount of misrouting at a given applied load, any messages which are misrouted will remain in the network and will require channel and buffer resources which may result in further misrouting.

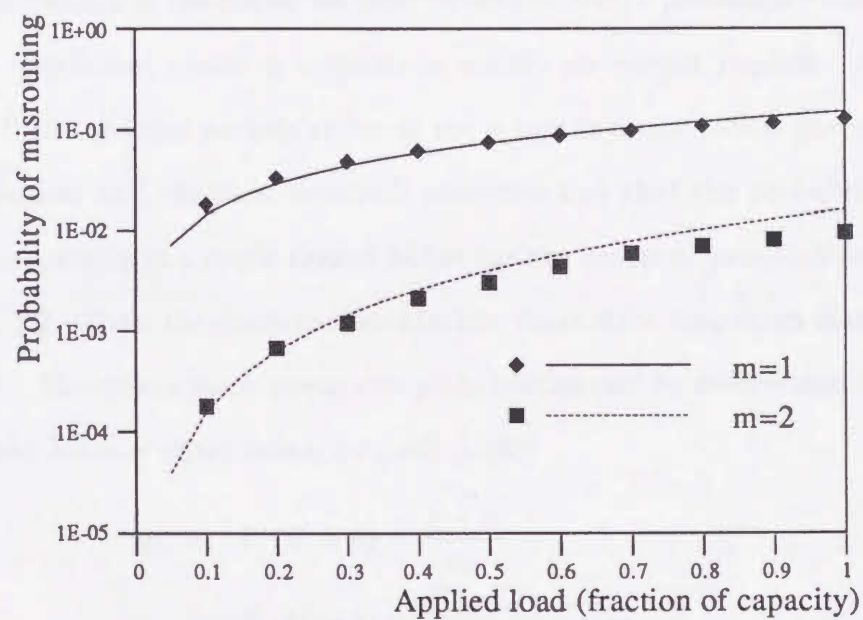


Figure 3.6: Probability of misrouting versus applied load for 16-ary 2-cube. Solid lines are predicted values, points are measurements taken by simulation

3.3.2 Buffer Evaluation

The output of each port has a set of m queues for temporarily storing packets. These FIFO queues operate as a single shared buffer for the associated port, while the output controller ensures that a first-in first-out queuing discipline is maintained for packets arriving at that output. If no packets are lost in the queue switches, then in order to select an appropriate buffer size for each dimension of the router we need to determine the probability that there exists insufficient space in a queue to satisfy an output request. Assume again that fixed size packets arrive at the n inputs to the router governed by independent and identical Bernoulli processes and that the probability of i packets arriving at a single shared buffer has the binomial probabilities given in Eq. 3.2. Given the discrete-time Markov chain state transition diagram of Fig. 3.7, the steady state queue size probabilities can be determined directly from the Markov chain balance equations[33]

$$\begin{aligned}
 q_0 &\equiv \Pr(Q = 0) = \frac{1 - \alpha}{a_0} \\
 q_1 &\equiv \Pr(Q = 1) = \frac{(1 - a_0 - a_1)}{a_0} q_0 \\
 &\vdots \\
 q_n &\equiv \Pr(Q = n) = \frac{1 - a_1}{a_0} q_{n-1} - \sum_{i=2}^n \frac{a_i}{a_0} q_{n-i} \quad (3.8) \\
 n &\geq 2
 \end{aligned}$$

and it follows that the probability that a queue size is greater than or equal to some value, L , is the sum of the probabilities of queue lengths greater than or equal to L

$$\Pr(Q \geq L) = \sum_{i=L}^{\infty} q_i \quad (3.9)$$

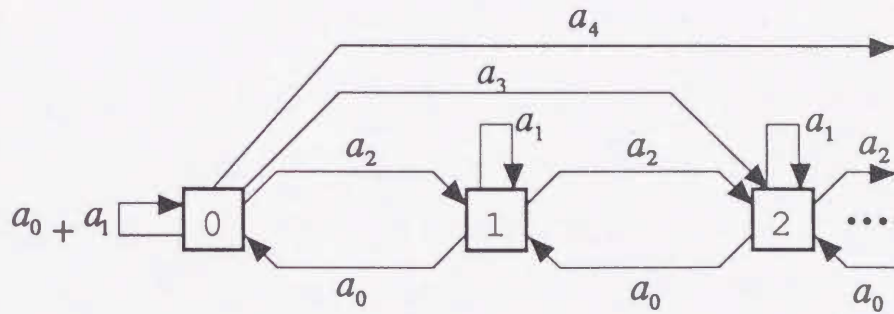


Figure 3.7: The discrete-time Markov chain state transition diagram for the output queue size

As packets are permitted to request more than one output, the probability of misrouting is given by the sum of the probabilities that a packet has j dimensions still to traverse, $0 \leq j \leq n$, multiplied by the probability that the queue sizes of the requested queues exceed L , raised to the j^{th} power

$$\Pr(M_{b=L}) = \sum_{j=1}^n \left[\beta_j \left(\sum_{i=L}^{\infty} q_i \right)^j \right] \quad (3.10)$$

Applying Eq. 3.10, we can evaluate the probability of misrouting if the queue sizes are fixed at L packets. Figure 3.8 illustrates how the probability of misrouting due to buffer overflow varies as a function of the applied load for queue sizes of 2, 4 and 8 packets in a 16-ary 2-cube, along with results obtained by simulation of a single router. As can be seen in the figure, the results predicted by the Markov chain model remain in close agreement with the simulation results, except at high applied loads where the Markov approximation overestimates the overflow rate.

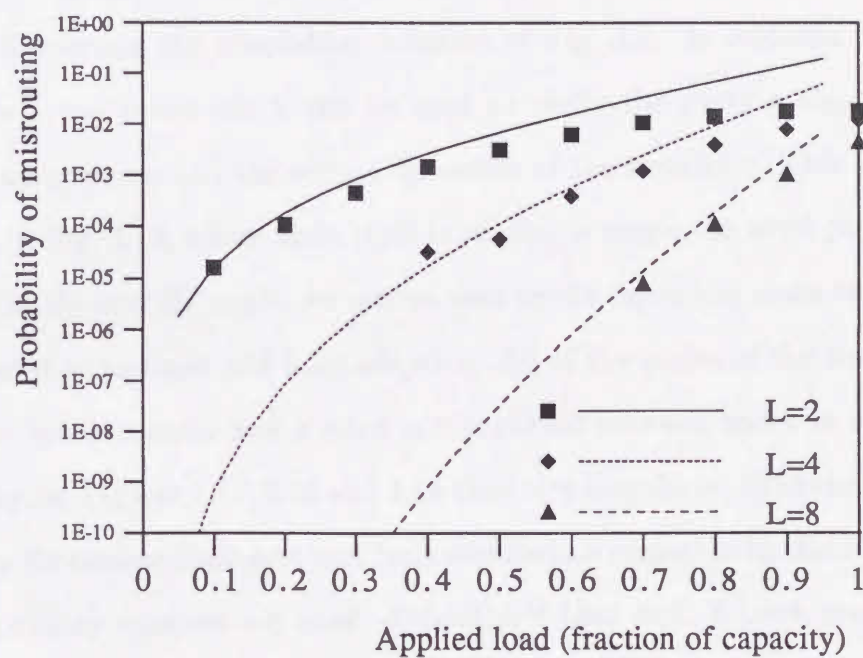


Figure 3.8: Performance of output queues. Solid lines are predicted values, points are measurements taken by simulation

3.4 Performance

In this section, we evaluate the performance of the Tokkyū router under a variety of traffic conditions by simulation. The simulator is a C++ program with a graphical user interface and includes a dynamic display of the simulation progress. The simulator supports programmable network size, buffer size, routing algorithm, traffic pattern and packet length as shown in the dialog for setting the simulation variables of Fig. 3.9. In addition to this there is a test mode which can be used to verify the routing algorithms, buffer assignments and the correct operation of the simulator. This is illustrated in Fig. 3.10, where node (0,0) is sending a single, 16 word packet to node (15,15) in a 2D mesh. As can be seen in the figure the route taken by the packet is minimal and fully adaptive. All of the nodes of the simulator operate synchronously and a word is transferred between nodes in a single clock cycle. Figures 3.11, 3.12 and 3.13 illustrate snapshots of the simulation display for random, hot-spot and fault simulations respectively. Each square in the display windows +X Load, -X Load, +Y Load and -Y Load, represents the buffer load for the given dimension of the corresponding router, while the display window, Ave. Load, shows the average load of the buffers of the corresponding router. The display has proved invaluable in the development of the simulator, as well as providing insight into the results obtained.

Network performance under uniform random traffic, hot-spot traffic and traffic in the presence of router faults has been simulated. Simulations were all performed with two-dimensional tori (16-ary 2-cubes) and a packet size of 16 words. In order to accurately model the performance of a practical router

Network/Routing: <input type="radio"/> 2D Mesh/e-Cube <input type="radio"/> 2D Uni-Torus/RI <input checked="" type="radio"/> 2D Bi-Torus/RI <input type="radio"/> 2D Bi-Mesh/RI	Buffer Size: O/P Buffer: <input type="text"/> words
Simulation Mode: <input checked="" type="radio"/> Random <input type="radio"/> Matrix Multiply <input type="radio"/> Gaussian Elimination <input type="radio"/> Hot Spot <input type="radio"/> Fault <input type="radio"/> Test	Network Size: X Size: <input type="text"/> nodes Y Size: <input type="text"/> nodes
Default Switch Mode: <input checked="" type="radio"/> Passive <input type="radio"/> Active	Max. Packet Length: Length: <input type="text"/> words
Cancel OK	Simulation Settings: Start Load: <input type="text"/> % End Load: <input type="text"/> % Step Size: <input type="text"/> % Res. Space: <input type="text"/> Arb. Delay: <input type="text"/> cycles Sw.Size: <input type="text"/>

Figure 3.9: Dialog for setting simulation variables

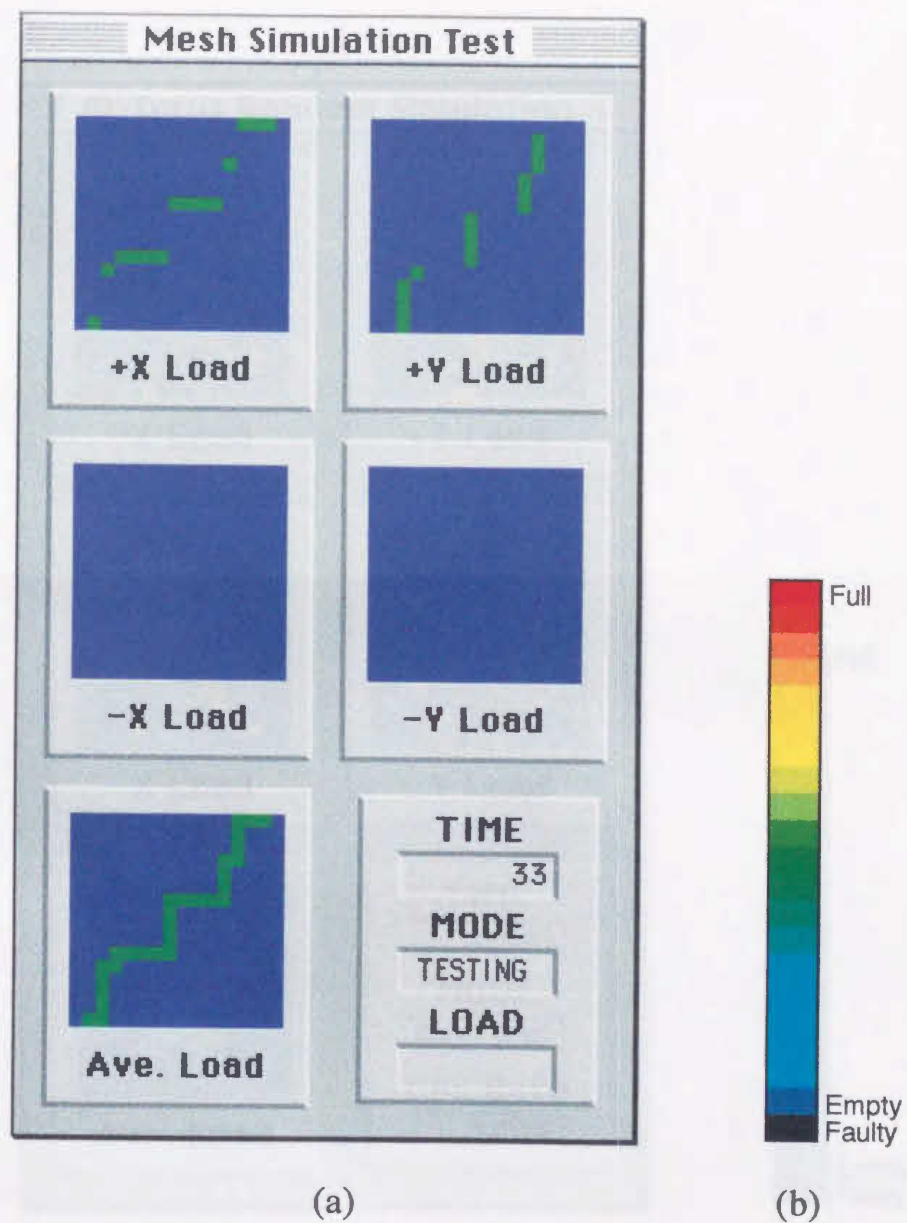


Figure 3.10: (a) Simulation display showing test mode (b) Simulation display key

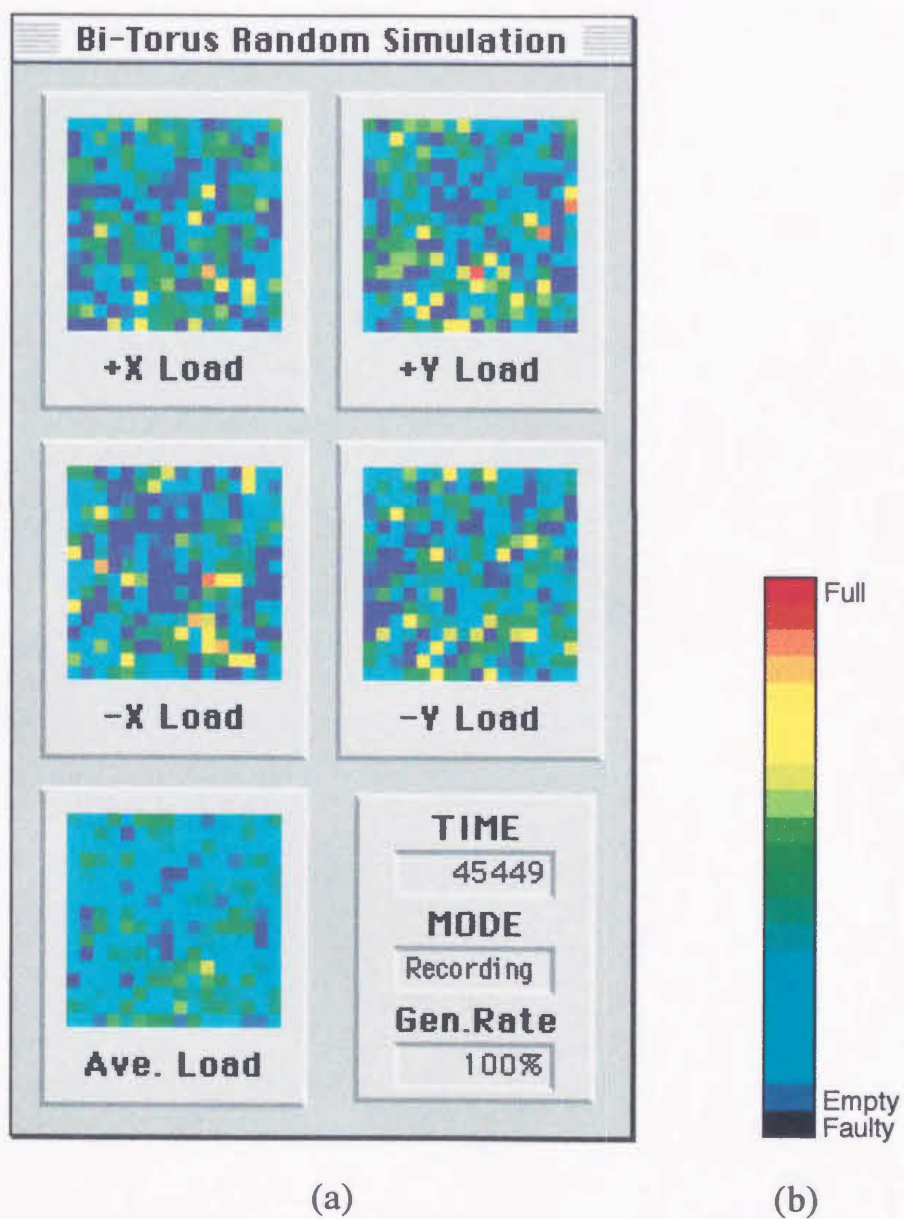


Figure 3.11: (a) Simulation display showing random simulation (b) Simulation display key

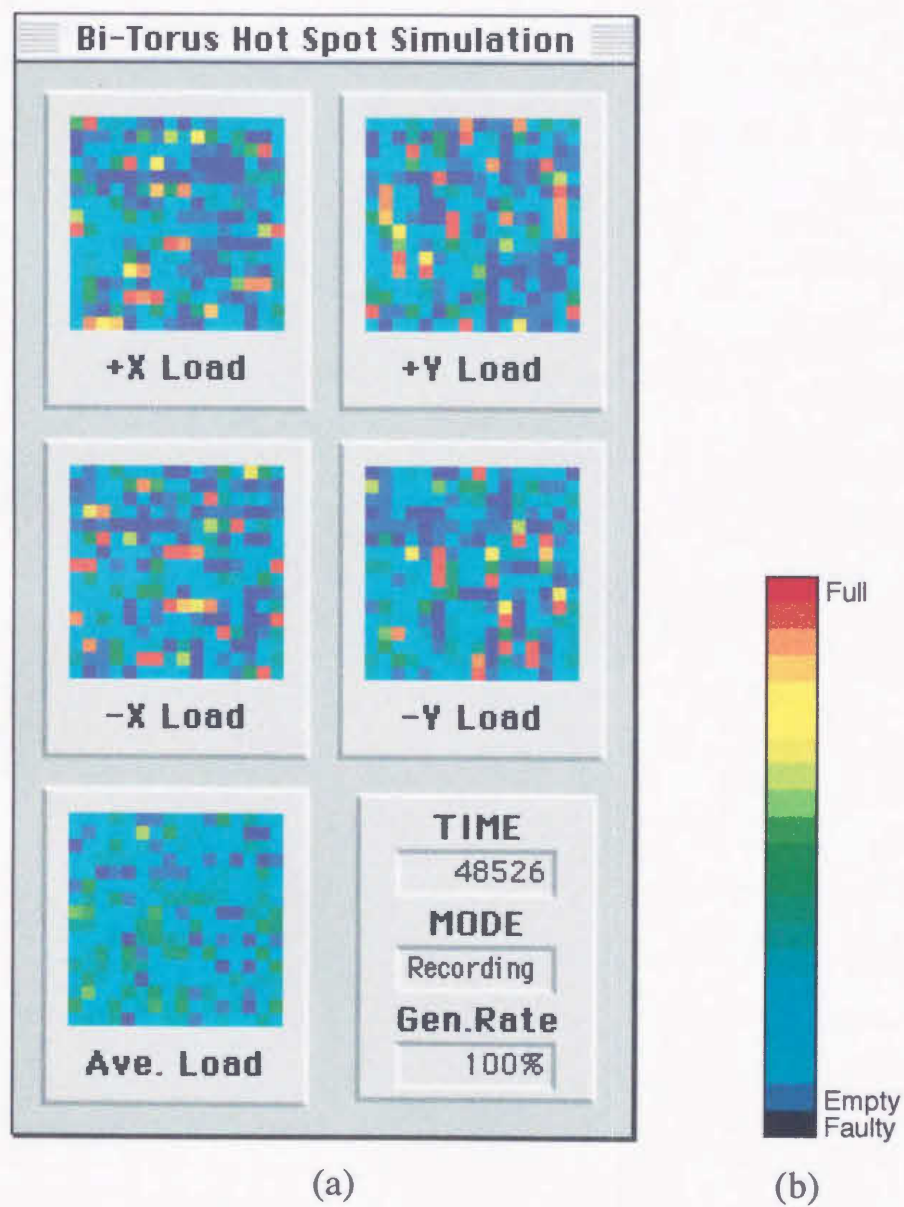


Figure 3.12: (a) Simulation display showing hot-spot simulation (b) Simulation display key

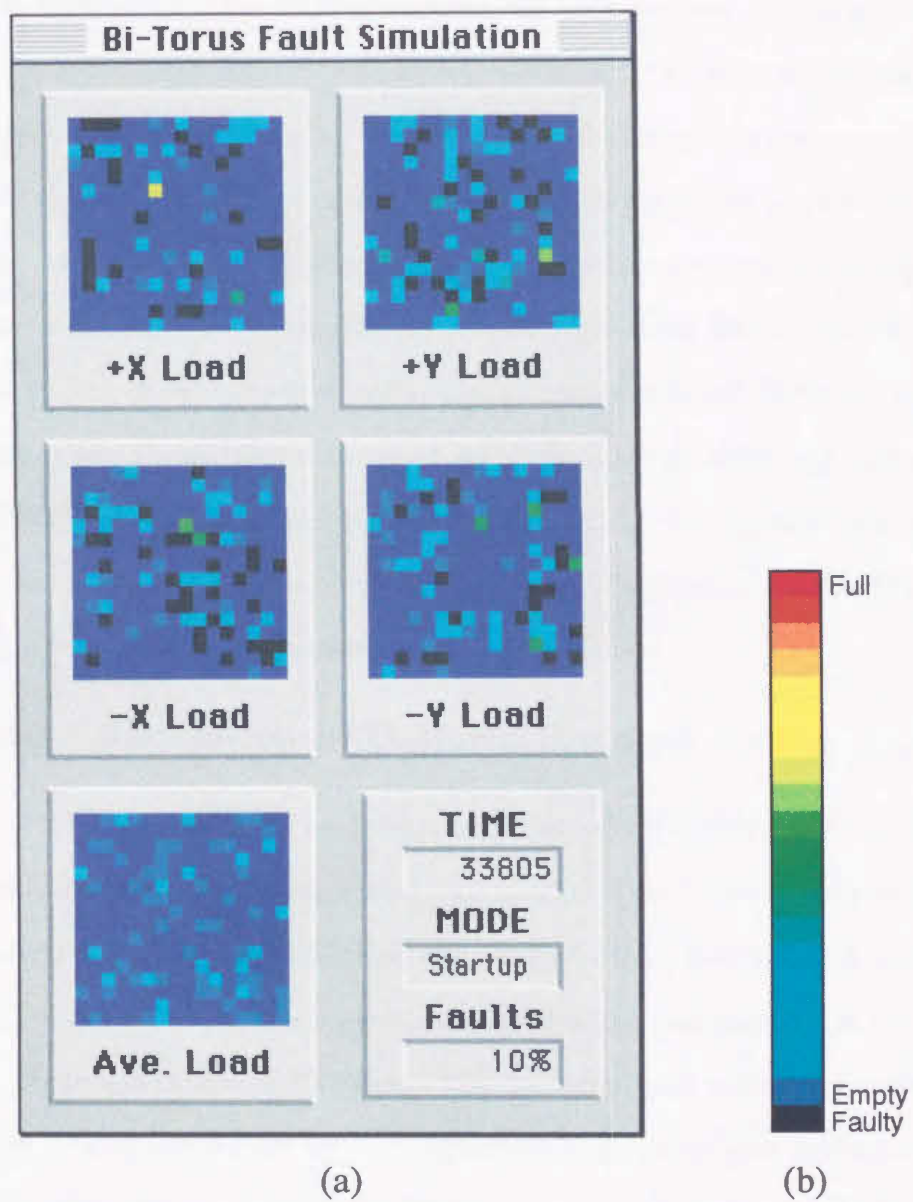


Figure 3.13: (a) Simulation display showing fault simulation (b) Simulation display key

we have fixed the uncongested routing latency of each router at 4 cycles. The assumed cycle-by-cycle operation of the router is as follows; The header of a packet entering the router will be decoded and a routing request made in the first clock cycle. The routing decision will be made and an output assigned in the second and third cycles and the header will be updated and sent to the output in the fourth cycle. This is typical of current generation routers[8]. Packets using the *packet expressway* only require that the header be checked for a value of zero, indicating that the packet has completed routing in the current dimension. Therefore the *packet expressway* has a latency of only one cycle. In all instances, collection of results was not initiated until the latency and throughput measurements of the network under test had reached a steady state. In the presentation of the results, the applied network load of the networks has been normalized such that full load corresponds to all of the network channels transmitting simultaneously.

3.4.1 Simulation of Uniform Random Traffic

In order to evaluate the performance of the network under uniform random traffic a constant rate source with exponential interarrival times was applied to each input and the time from the creation of the first word of the packet until the last word of the packet is accepted at the destination was measured.

Figures 3.14 and 3.15 present the predicted and simulated misrouting rates in a 16-ary 2-cube for varying switch and queue sizes respectively. In these simulations a packet requesting more than one output was randomly assigned to one of those outputs available to it. The simulation result for a queue switch size of 4:1 in Fig. 3.14 is initially higher than the predicted re-

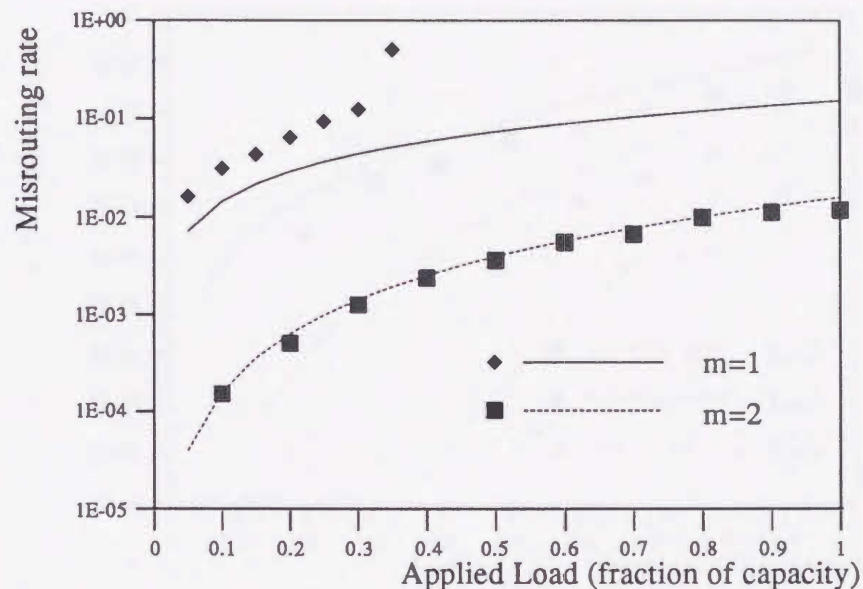


Figure 3.14: Performance of queue switches for 256 node 16-ary 2-cube. Solid lines are predicted values, points are measurements taken by simulator

sult, due to the higher traffic present in the network as a result of misrouting. At 30% applied load the measured network load is 45% and the misrouting rate is 13.4%. At approximately 35% applied load the extra traffic produced by misrouting causes network operation to become unstable and results in a misrouting rate of 50%. A switch size of 4:2 is sufficient to maintain stable network operation and the simulation and predicted results remain in close agreement.

The predicted misrouting due to buffer contention in Fig. 3.15 overestimates the measured rate for buffer sizes of 2, 4 and 8 packets. All of the simulations remained stable, with the misrouting rate rising steadily as the applied load was increased. A minimum buffer size of only 2 packets per port is sufficient to guarantee stable network operation. Figures 3.16 and

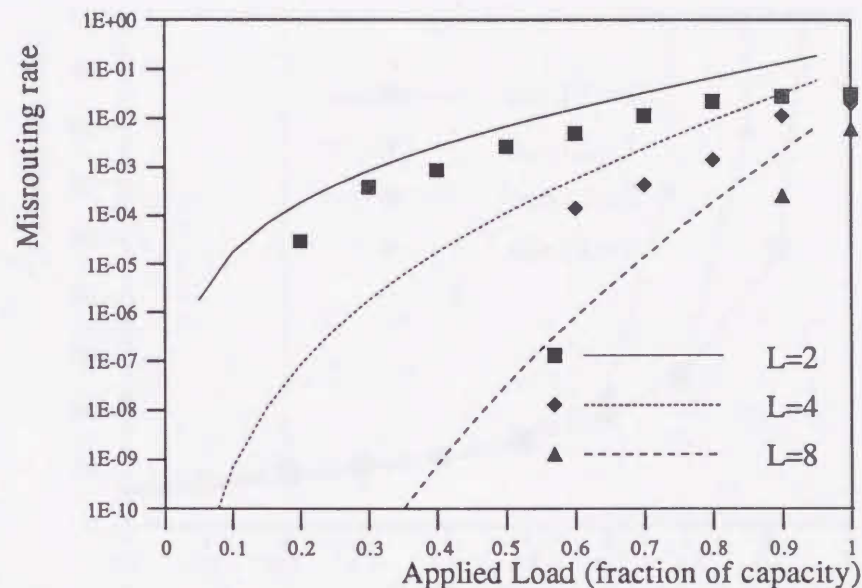


Figure 3.15: Performance of output queues for 256 node 16-ary 2-cube. Solid lines are predicted values, points are measurements taken by simulator

3.17 show the average packet latency and network throughput as a function of applied network load respectively, for a 256 node 16-ary 2-cube and a number of different switch and buffer configurations. With a single switch output and buffering for one packet per port, ($m=1, L=1$), the misrouted traffic causes the network to saturate at 35% applied load, and the throughput is reduced to just 3%. Increasing the number of switch outputs to two and the buffer size to two packets, ($m=2, L=2$) gives a significant improvement in performance with a saturation throughput of 80%. Increasing the switch and buffer sizes to three outputs and three packets respectively, ($m=3, L=3$) further increases the saturation throughput to 90%, while further increases in buffer size give diminishing returns. This is highlighted by the plot for a switch size of three outputs and a buffer size of 16 packets, which saturates

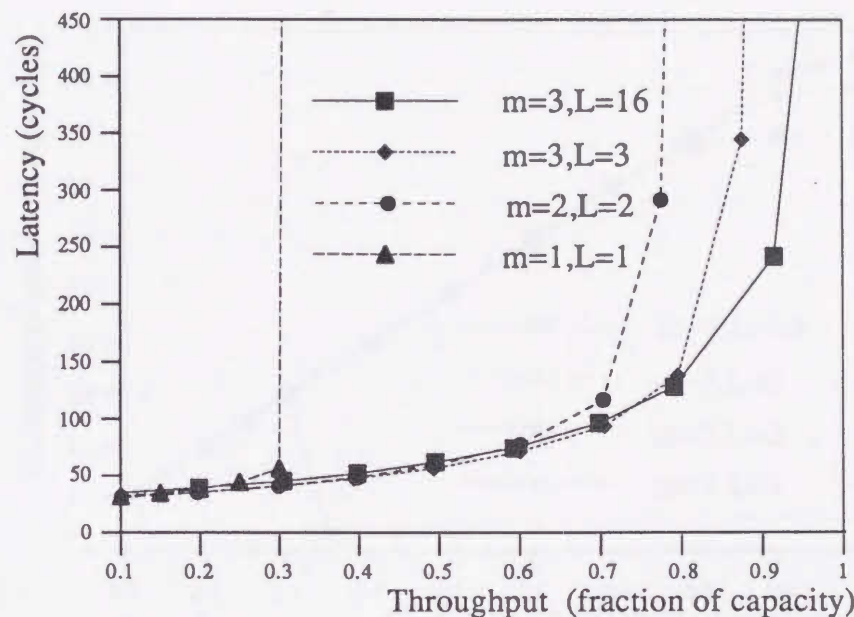


Figure 3.16: Latency versus offered traffic for a 256 node 16-ary 2-cube under uniform random traffic

at 95% throughput.

Figure 3.18 illustrates the effectiveness of the *packet expressway* by comparing a network in which packets make use of the *packet expressway* with a network in which all packets are forced to pass through the core of the router. The average latency of packets in the network which utilizes the *packet expressway* is reduced significantly when compared to the network in which the *packet expressway* is disabled. This decrease in latency occurs at all applied loads and varies from a maximum of 43%, which occurs at 10% applied load, to 23% at an applied load of 95%. The maximum throughput of the network utilizing the *packet expressway* is also slightly higher, 95% versus 92%, due to packets in the network spanning a greater number of channels at any given time.

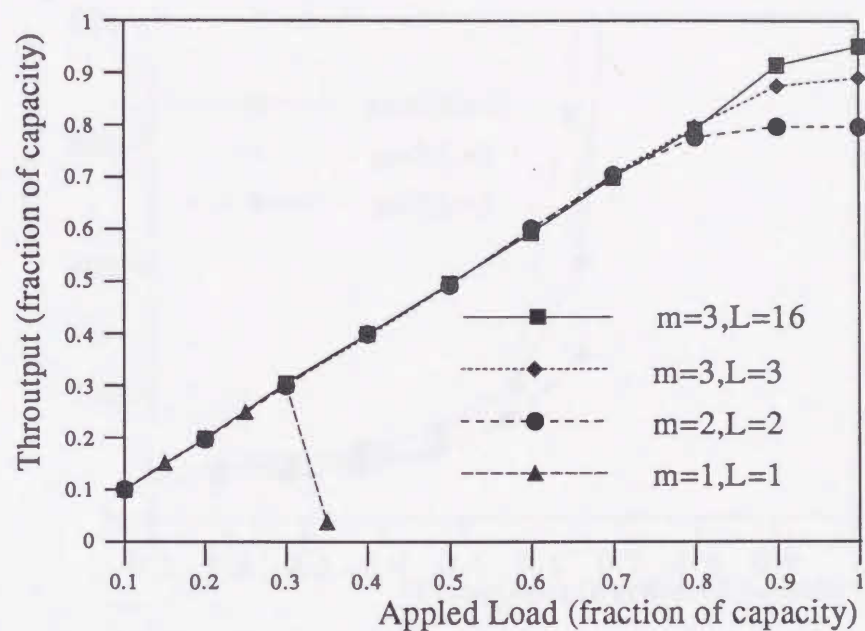


Figure 3.17: Throughput versus offered traffic for a 256 node 16-ary 2-cube under uniform random traffic

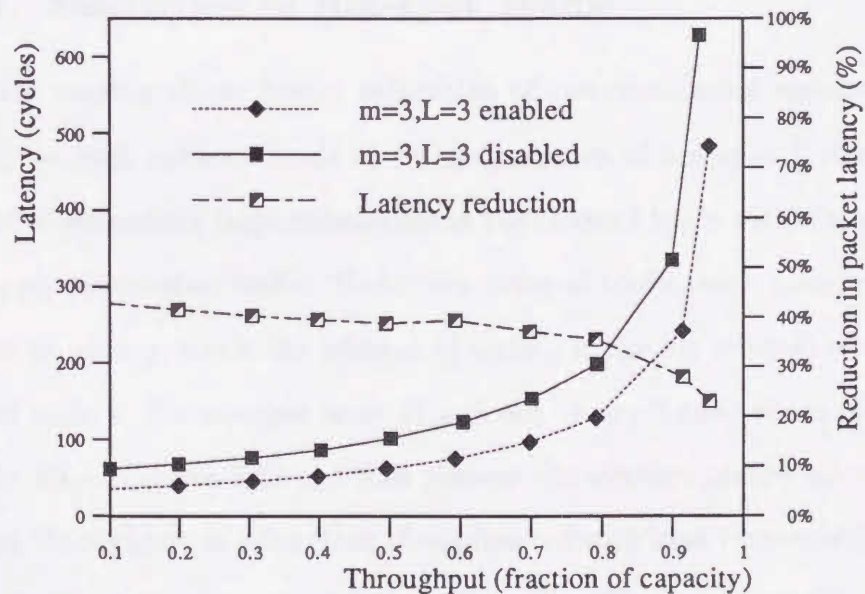


Figure 3.18: Latency and reduction in latency versus applied load under uniform random traffic with *packet expressway* enabled and disabled

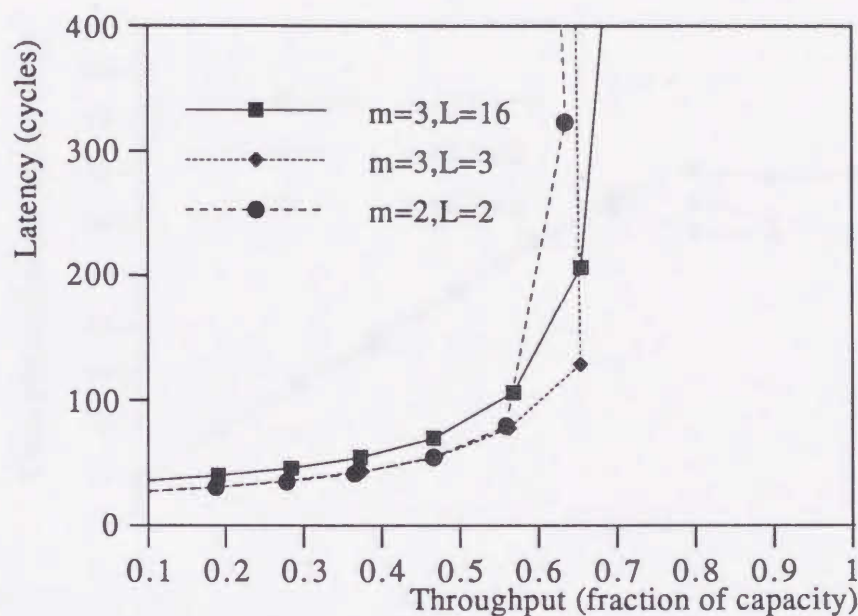


Figure 3.19: Latency versus offered traffic for a 256 node 16-ary 2-cube under bit reversal traffic

3.4.2 Simulation of Hot-spot Traffic

Adaptive routing allows better utilization of communication resources, especially at high network loads or in the presence of hot-spot traffic. One method of generating large imbalances in the channel loads within a network is to apply bit-reversal traffic. Under bit-reversal traffic, each node, p , sends packets to node q , where the address of node q is the bit reversal of the address of node p . For example node 27_{16} in our 16-ary 2-cube sends messages to node $E4_{16}$. Figures 3.19 and 3.20 present the average packet latency and network throughput as a function of applied network load respectively, for a 256 node 16-ary 2-cube under bit-reversal traffic. The maximum throughput for $(m=2, L=2)$ and $(m=3, L=3)$ are 63% and 65% respectively. Increasing the applied traffic rate past these points results in a decrease in the through-

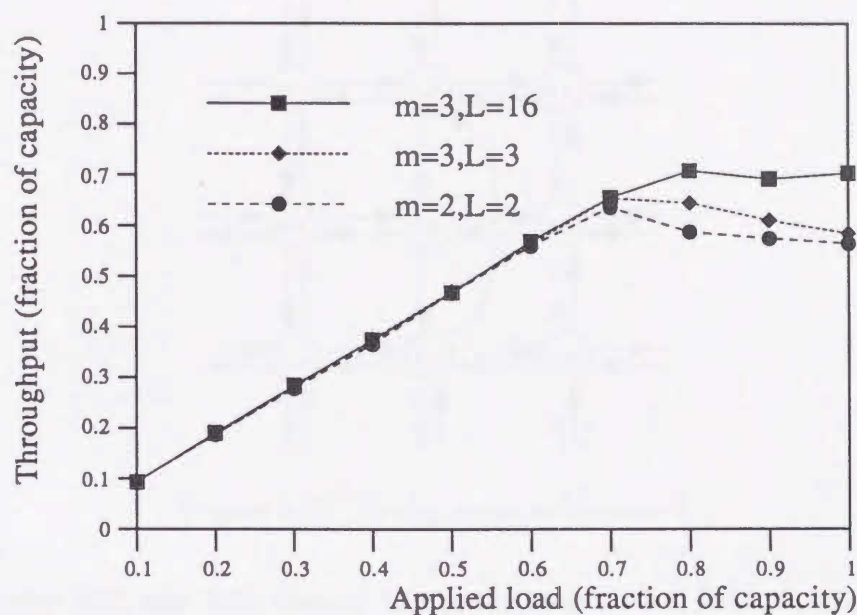


Figure 3.20: Throughput versus offered traffic for a 256 node 16-ary 2-cube under bit reversal traffic

put to 56% and 58%. Increasing the buffer size to 16 packets results in an increase in latency prior to saturation, due to packets queueing in the larger buffers, and an increase in the maximum throughput of 70%.

3.4.3 Simulation of Traffic in the Presence of Faults

The correct operation of the router requires that the aggregate input and output data rates remain balanced. Failure of a single channel of a router will require that the in-degree of the router be reduced by one to maintain the balance in data rates. In Fig. 3.21, the +Y channel of router (4,5) has failed and so it is bypassed, creating a connection between nodes (3,5) and (5,5). Depending upon the nature of the fault it may be possible to use the *packet expressway* of node (4,5) to provide the bypass path.

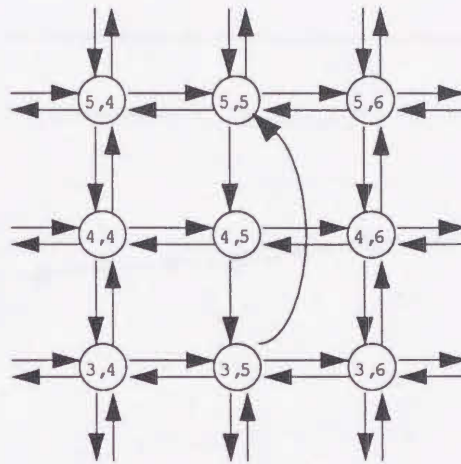


Figure 3.21: Faulty node is bypassed

Figures 3.22 and 3.23 present the performance of a Tokkyū network in the presence of faults. The network on which the faults were simulated had switch and buffer sizes of two outputs and two packets respectively, a constant applied load of 50% and uniform random traffic. Ten fault simulations were carried out, each with randomly generated fault sets and the results were averaged to produce Fig. 3.22 and Fig. 3.23. The network performance degraded only slightly, even with 10% of the available channels faulty, as can be seen in the figures. There was only a 26% increase in the packet latency from a fault-free network to a network with 10% faulty channels and the throughput remained fixed at approximately 50%.

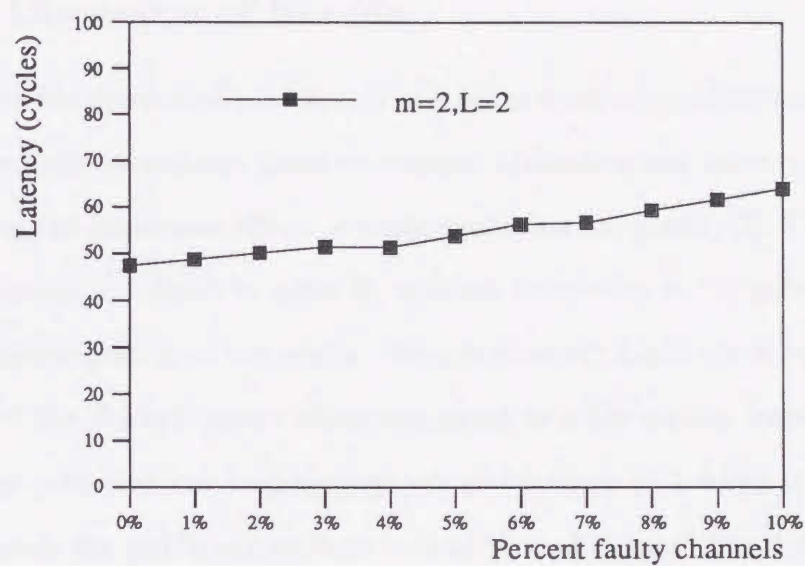


Figure 3.22: Average latency versus percent faulty channels at 50% applied load ($m=2, L=2$). Mean latency averaged over ten random fault sets

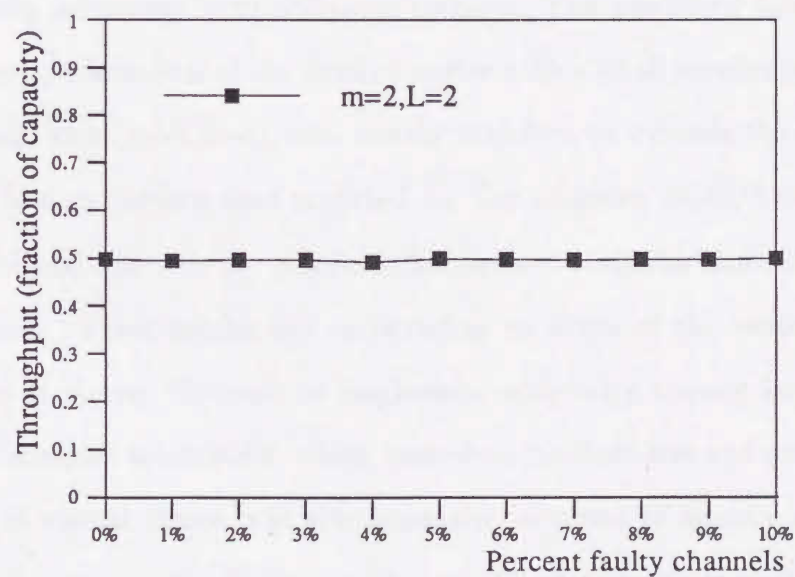


Figure 3.23: Throughput versus percent faulty channels at 50% applied load ($m=2, L=2$). Mean throughput averaged over ten random fault sets

3.4.4 Discussion of Results

A.A. Chien has illustrated the hazards of making comparisons between different router implementations based on channel utilization and latency without considering the important effects of implementation complexity[8]. The effect of these factors is difficult to quantify without simulation at the gate-level or actual implementation of the router. We can however, highlight a number of features of the Tokkyū router when compared to other similar implementations. The predicted low load throughput and latency of Tokkyū is as good as or exceeds the published performance of virtual channel based oblivious routers[50, 48, 16] due to the low latency path provided by the *packet expressway*. In networks experiencing high load, hot-spots or fault conditions, small Tokkyū routers, $(m=2, L=2)$ or $(m=3, L=3)$ have a clear throughput and latency advantage over oblivious routers. The predicted latency and throughput performance of the Tokkyū router with a small number of buffers, $(m=2, L=2)$ and $(m=3, L=3)$, also closely matches, or exceeds the throughput and latency performance reported for the adaptive Dally/Aoki router, with 16 virtual channels per physical channel and a similar amount of total buffer space. These results are encouraging as many of the routers which make use of virtual channels to implement adaptivity require large cross-bars and complex arbitration, which contribute to their size and complexity. The use of virtual channels is also expensive in terms of latency and cycle time[8]. However, as the Tokkyū router must buffer complete packets when output contention occurs, it requires the use of comparatively short packets, i.e. less than 32 bytes. The cost of message disassembly for transmission

and reassembly at the destination, along with the cost of potentially larger packet headers, would have to be included in the latency and throughput measurements to make a direct comparison with virtual channel routers.

Both the Chaos router and the Ngai/Seitz router have similar architectures to the Tokkyū router and thus a more accurate comparison can be made between them. The simulated performance characteristics of these two routers are again similar to the results reported here. The low latency register-insertion ring formed by the *packet expressway* allows the Tokkyū router to achieve lower packet latency than the Chaos and Ngai/Seitz router, especially at low network load. The *packet expressway* achieves lower latency in a manner similar to the *Express Cubes* proposed by Dally [14]. However, unlike Express Cubes, the *packet expressway* does not require additional interchanges and wiring, thereby simplifying the network design and implementation. The simple routing decisions made by the Tokkyū router, which are made using only the message header and current buffer and switch information of the router, will allow for simpler arbiter implementation and therefore faster operation. The simulation results demonstrate that, for a 16-ary 2-cube, two or three queue switch outputs, each with sufficient buffer space for a single packet, are sufficient for a low probability of misrouting, low latency and high throughput.

Chapter 4

Restricted-length Hardware Multicasting in Multicomputer Networks

We begin this chapter by carrying out an in-depth investigation into multicast deadlock in wormhole routed communication networks. This is followed by a presentation of a hybrid virtual cut-through/wormhole routing method for the effective distribution of broadcast and multicast messages in MPP system networks, called restricted-length multicasting [27]. This method uses a single enlarged flit buffer per physical communications channel to provide virtual cut-through routing for multicasts at the nodes where the message is replicated, thus preventing deadlock.

4.1 Preliminaries

4.1.1 Definition of Multicast Deadlock Problem

Multicast deadlock will now be examined in detail using a graph theoretical approach. Any graph theoretical terminology not defined here may be found in [10, 39]. In the following discussion we make the following assumptions:

1. There are no cycles in the channel dependency graph of the unicast routing algorithm, i.e. unicast is deadlock free¹.
2. There are no cycles in the channel dependency graph of the multicast routing algorithm.
3. A destination node will eventually consume a message.

Let the set of nodes, $M = \{n_0, n_1, \dots, n_{k-1}\} \subseteq N$, be referred to as the *multicast set*, M , with $k - 1$ destinations. Let n_0 be the source node and $D = \{n_1, \dots, n_{k-1}\}$ be the destination nodes of the multicast set, and let P be the number of nodes in the set $N(G)$. A unicast is therefore a multicast with $k = 2$, and a broadcast is a multicast with $k = P$.

Definition 6 The multicast routing function $\mathfrak{Rm} : N \times N \rightarrow C$ maps the current node, n_c , and the destination nodes, $n_d \in D$, to the next channel(s), c_n , for the routes from n_c to $n_d \in D$.

Definition 7 The *resource tree* of the multicast set M is the rooted subtree, $RT(N, C)$ of $G(N, C)$, which has n_0 as the root, and where $N(RT) \subseteq N(G)$ and $C(RT) \subseteq C(G)$. The vertices $N(RT)$ and the arcs $C(RT)$ are the nodes and channels of the interconnection network respectively, and are defined by \mathfrak{Rm} for the multicast set M . The resource tree of a unicast is therefore a rooted tree $RT(N, C)$ with only one *branch*.

Let L be the length of the multicast packet P_m in flits, and B_d be the depth of a flit buffer in node n_c . If node $n_c \in RT$ contains the tail of the

¹For a complete discussion of channel dependency and deadlock avoidance for unicast messages in wormhole routed networks refer to [18].

multicast packet P_m in one of its flit buffers, then the *concurrent resource tree* of RT at time t_a is the rooted subtree, $CT(N, C)$ of $RT(N, C)$, whose root is n_c . The nodes $N(CT) \subseteq N(RT)$ and arcs $C(CT) \subseteq C(RT)$ are the set of resources which are required concurrently, before the tail of P_m can leave n_c . The *path length* of a vertex in CT is defined as the number of edges from n_c to the vertex. The *height* of the tree CT , defined $H(CT)$, is the maximum of the path lengths in CT , and the number of nodes in the path of maximum length is equal to $H(CT) + 1$. If $\{L/Bd \geq H(CT) + 1\}$ then $CT = RT$, and if $\{H(CT) = 1\}$ then either $L \leq Bd$, or n_c is adjacent to the destination nodes $n_d \in D$. Let $CT(N, C)$ and $CT'(N, C)$ be the concurrent resource trees of $RT(N, C)$ and $RT'(N, C)$ respectively. The intersection of two concurrent resource trees, $I = CT \cap CT'$, is given by $N(CT) \cap N(CT')$ and $C(CT) \cap C(CT')$. The *number of components* of I , denoted $\omega(I)$, is defined as the number of connected subgraphs of I , that are not contained in any other connected subgraph of I , and let $\omega(I_c)$ be the number of components of I , whose degree ≥ 1 . If a component in I has a degree equal to zero, then the component consists of a single vertex, with no incident arcs.

Theorem 1 Let $R_s = \{CT_0, CT_1, \dots, CT_{l-1}\}$ be the set of concurrent resource trees for ℓ concurrent multicasts at time t_a . Deadlock due to the concurrent allocation of resources may only occur if and only if the following conditions apply:

$$\forall (RT_i, RT_j) \in R_s, \exists \{I | (I = RT_i \cap RT_j \neq \emptyset), (0 \leq i \leq \ell, 0 \leq j \leq \ell, i \neq j)\} \quad (4.1)$$

$$\forall(I \neq \emptyset), \exists\{\omega(I_c) | \omega(I_c) \geq 2\} \quad (4.2)$$

Proof: \Leftarrow

Let CT and CT' be two concurrent resource trees in R_s , where $I = CT \cap CT'$.

- 1.1 $I = \emptyset$, no concurrent resources are shared by the concurrent multicasts in R_s . Therefore assumptions 1, 2 and 3 are sufficient to guarantee deadlock avoidance.
- 1.2 If $I \neq \emptyset$ and $\omega(I_c) = 0$, it follows that $N(CT) \cap N(CT') \neq \emptyset$ and $C(CT) \cap C(CT') = \emptyset$. Therefore only node resources are shared by concurrent multicasts and assumptions 1, 2 and 3, and a fair local arbitration scheme are sufficient to guarantee deadlock avoidance.
- 1.3 If $I \neq \emptyset$ and $\omega(I_c) = 1$ then there is a single rooted subtree in I , which we denote S_u . Let n_u be the root node of S_u , with output ports p_u , and $c_u \in p_u$ be the output channels of n_u defined by \mathcal{Rm} for the packets associated with CT and CT' . If at time t_a , n_u allocates all of the output channels c_u to the packet associated with CT , then CT' will remain blocked until the packet associated with CT releases its resources. Thus, only one multicast is given access to the resources below n_u and assumptions 1, 2 and 3, and a fair local arbitration scheme are sufficient to guarantee deadlock avoidance.

\Rightarrow

2.1 If $I \neq \emptyset$ and $\omega(I_c) = 2$ then there are two rooted subtrees in I that are required concurrently by CT and CT' , which we denote S_u and S_v . Let n_u and n_v be the root nodes of S_u and S_v , and $c_u \in p_u$ and $c_v \in p_v$ be the output channels of n_u and n_v defined by \mathfrak{Rm} for the packets associated with CT and CT' respectively. If at time t_a , n_u allocates $c_u \in p_u$ to the packet associated with CT and n_v allocates $c_v \in p_v$ to the packet associated with CT' , a concurrent allocation of dependent resources has occurred, and a deadlock situation has been reached.

Corollary 1 *Deadlock due to the concurrent allocation of resources cannot occur in a network employing virtual cut-through routing.*

Proof: \Leftarrow

3.1 By definition, the length of a packet in a network employing virtual cut-through is $L \leq Bd$. $H(CT)$ is therefore equal to 1, and $\forall(I \neq \emptyset), \omega(I_c) \leq 1$. Thus by proofs 1.1 and 1.2, multicast is free of deadlock due to the concurrent allocations of resources.

Figures 4.1(a) and (b) illustrate a multicast and its associated concurrent resource trees respectively for virtual cut-through routing. In Figure 4.1(b) $\forall(I \neq \emptyset), \omega(I_c) \leq 1$ and therefore deadlock cannot occur due to the concurrent allocation of resources. If a single channel of a branch in the restricted-length multicast tree becomes blocked, it will not result in the rest of the tree holding channel resources, as is the case in conventional tree-based multicasting.

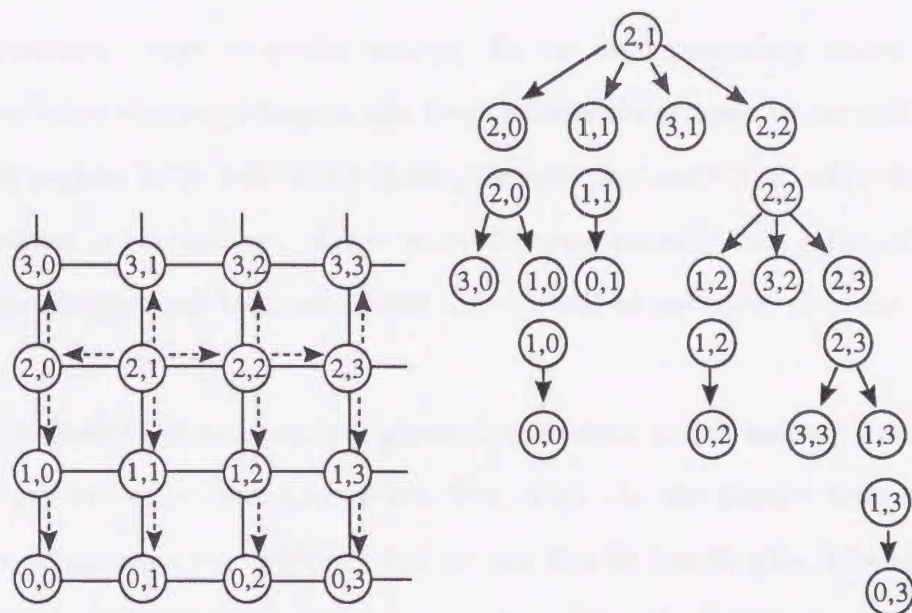


Figure 4.1: (a) Multicast by node (2,1) and (b) the resulting concurrent resource trees

4.2 Restricted-Length Multicasting

Rather than restricting the branching of a multicast, we propose restricted-length multicasting, in which the packets of a multicast message are restricted in length so that they are routed in a virtual cut-through manner in a network which usually supports wormhole routing. Messages are usually divided into one or more packets at the source, prior to injection into the network. Thus, in order to implement restricted-length multicasting in a network which normally supports wormhole routing, the source node must divide a multicast message into packets of length $L \leq Bd$. By ensuring that a flit buffer is sufficiently large to hold a complete multicast packet, or that a packet is sufficiently small to fit in a single buffer, it is therefore possible to implement deadlock free multicasting utilizing existing routing algorithms such

as dimension order, or e-cube routing. As has been previously stated, each router must also implement a fair local arbitration scheme to prevent multicast packets from indefinitely holding output port resources, while waiting for others to become free. However, as this requires only local information a simple timeout and resource release scheme will be sufficient to avoid deadlock.

The buffers of most current generation routers, which employ wormhole routing, can only store one or two flits each. As the header information for a single packet is typically one or two flits in length also, it would be impractical to implement restricted-length multicasting on these systems. A simple solution would be to increase the size of the buffers so that a complete packet could be stored in each buffer, thus implementing virtual cut-through routing. However, this would significantly increase the size of the message router, which would complicate its design and result in lower performance. Another approach would be to increase the size of a single buffer so that it can hold an entire packet. While this approach is preferable to increasing the size of all of the buffers, the size of a buffer capable of storing the maximum length packet employed in the system may still be prohibitively large. Our proposed approach is therefore to increase the size of a single buffer, while restricting the length of multicast packets. When a packet appears at the input to a router, a single bit in the header indicates whether the message is a multicast or a unicast. If the multicast bit is set, then the message must request the enlarged buffer, while unicast messages are free to be placed in any available buffer.

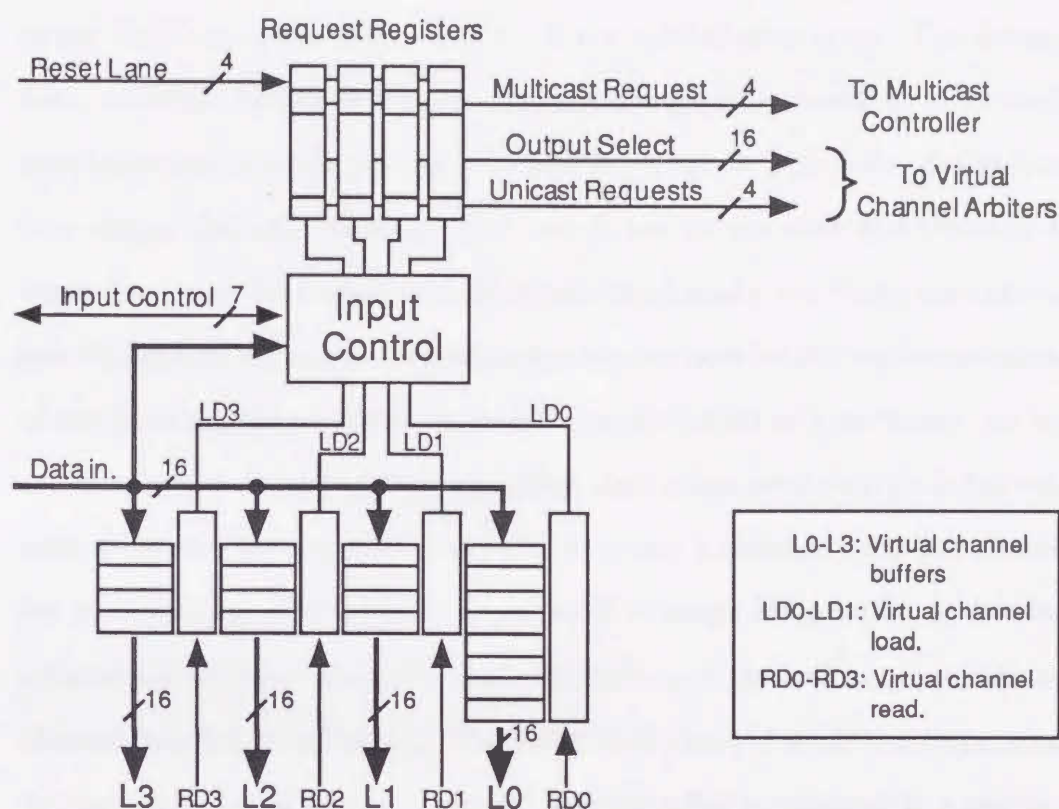


Figure 4.2: Organization of a single MEGA router input

4.2.1 Gate-array Implementation

A number of researchers and commercial enterprises have developed hardware routers for use in multicomputer networks in recent years [17, 19, 50]. These have typically been implemented using full custom VLSI techniques, which have enabled them to achieve high throughput and low switching latency. However, a number of advantages exist in taking a semi-custom approach to the design. These include a shorter design time, lower production costs for small volumes of devices, and well established design and simulation tools [36].

We are therefore undertaking the design of a MESSage passing Gate-

Array (MEGA) router [25], using a $1.2\ \mu\text{m}$ CMOS gate array. The design tools available include schematic capture, design rule checking, functional simulation and critical path analysis. Our second prototype router design has four virtual channel buffers per port, which are 16 bits wide and typically 4 words deep, and the router contains 10 uni-directional ports which are formed into 5 bi-directional pairs. The minimum requirement for the implementation of restricted-length multicasting is that a single packet of a multicast can be accommodated in a virtual channel buffer. As the header of each packet in our system requires 4 bytes, this would result in only 4 message bytes per packet for multicasting. To increase the ratio of message information to header information we have enlarged a single flit buffer per physical communications channel, labeled L0 in Fig. 4.2. The buffer load lines (LD0-LD3) are operated by the input control to load a virtual channel buffer in response to a request on the input control lines. The input control section also controls the request register associated with each virtual channel, placing a new request in a register whenever a new packet is received. These requests are passed to the appropriate arbiters via the request and the select lines. Once a packet has been passed to an output, the output control section (not shown) will assert the reset lane line to indicate that the lane is now free. The output controllers are also responsible for asserting the virtual channel read lines (RD0-RD3), once for each word which is read.

The basic unit for the implementation of digital logic within a gate array is the Basic Cell (BC or cell). Each BC is typically implemented as two pairs of P-channel and N-channel transistors and the logical function performed by each basic cell is determined by the metalization pattern assigned to it. A

Buffer Size	Cell Count	Terminals	Nets
4 lanes \times 4 words	2119	3709	734
3 lanes \times 4 words, 1 lane \times 8 words	2512	3351	847
3 lanes \times 4 words, 1 lane \times 16 words	3440	6408	1322

Table 4.1: Resource usage for various buffer structures

user creates a design using Unit Cells (UCs), such as NAND gates, flip-flops and shift registers, by interconnecting them using wiring networks (nets) and this design is then mapped to the gate array by the design software. Most UCs are made up of a number of BCs and thus these also require interconnection by nets. Terminals are used to provide the connections between BCs and nets, and also between nets on different metalization layers within the device. Although current gate array devices offer BC counts of more than 100,000 cells, the number of nets and terminals can significantly reduce the maximum utilization of these cells. In order to evaluate the effect on the gate array implementation of our router due to enlarging a single virtual channel buffer in each input port, we examined the increase in the cell, terminal and net counts for varying sizes of virtual channel buffer. These results are presented in Table 4.1, which gives the cell, terminal and net counts for the input section of a single port. In each case the number of virtual channels is fixed at four, and the size of one lane is increased from 4 to 16 words in depth. As can be seen in the table, increasing the size of a single buffer per port from 4 to 16 words results in a considerable increase in the number of cells, nets and terminals. However, this increase is significantly less than that which would occur if the size of all of the buffers was to be increased.

4.3 Simulation

4.3.1 Multicast Latency

In order to evaluate the potential benefits of utilizing restricted-length multicasting we have implemented a simulator, which determines the latency of sending a multicast from a single node, based upon the design parameters of our message router. In our simulations we therefore assume a 2D mesh topology with 16 bit data paths, a header length (L_h) of a unicast message of 4 bytes, and that the standard size of a flit buffer is 4 double-byte words. Two bytes of the header contain the destination address, while the remaining two bytes contain the packet length and sequence information etc. Given a multicast message of length L_m bytes, the latency of sending a multiple-unicast based multicast to N destinations is given by:

$$D_u = \sum_{i=0}^{N-1} \frac{(L_m + L_h) D_{flit}(i)}{L_{flit}} \quad (4.3)$$

where D_{flit} is the average delay in sending each flit to destination i and L_{flit} is the size of a flit buffer. Figure 2.17 illustrated that, in a 2D mesh, a multi-path multicast message is broadcast by sending four copies of the message on individual multicast paths. The header appended to each copy of the message must contain a list of all of the destination addresses. Assuming that, as in the case of a unicast, each destination address requires 2 bytes, and that 2 additional bytes of status information are appended to the header, the average number of bytes per header for a multi-path message being broadcast to N destinations in a 2D mesh is given by

$$\overline{L}_h = \left(\frac{N+1}{2} \right) \quad (4.4)$$

and the send latency for a multi-path based multicast with four paths is therefore

$$\begin{aligned} D_{mp} &= \sum_{i=0}^3 \frac{(L_m + L_h) D_{flit}(i)}{L_{flit}} \\ &= \sum_{i=0}^3 \frac{\left(l_m + \left(\frac{N+1}{2} \right) \right) D_{flit}(i)}{L_{flit}} \end{aligned} \quad (4.5)$$

A restricted-length multicast will divide the L_m bytes of the multicast message into a number of flit sized packets. The data content of the each packet is $P_d = (L_{flit} - L_h)$ and the total amount of header information for required to broadcast a message of L_m bytes, assuming each header requires 4 bytes, is $4N_f$, where N_f is given by $N_f = L_m / P_d$ and is rounded up to the nearest whole number. The send latency of restricted-length based multicast is therefore given by

$$L_{rl} = D_{flit} \times \left(\frac{L_m + 4N_f}{L_{flit}} \right) \quad (4.6)$$

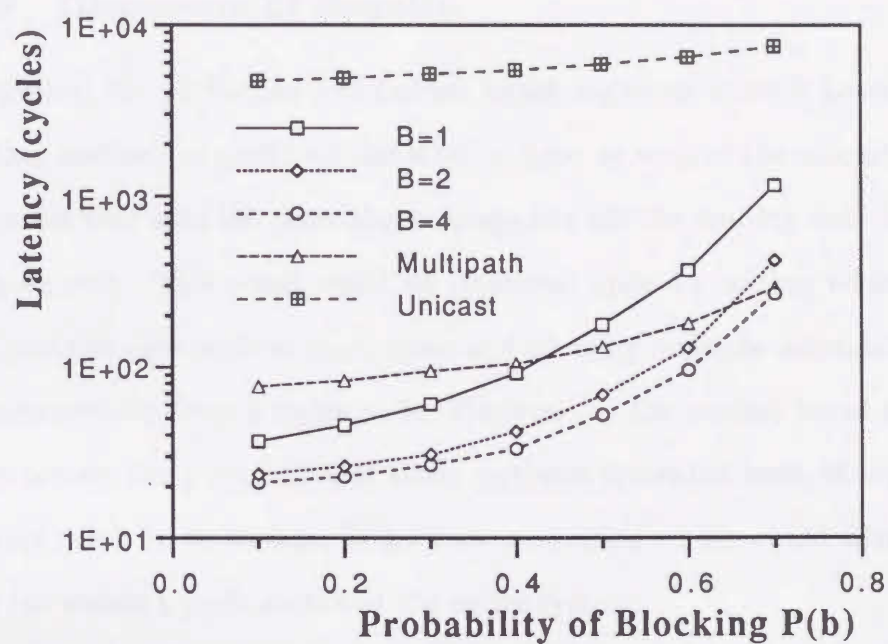
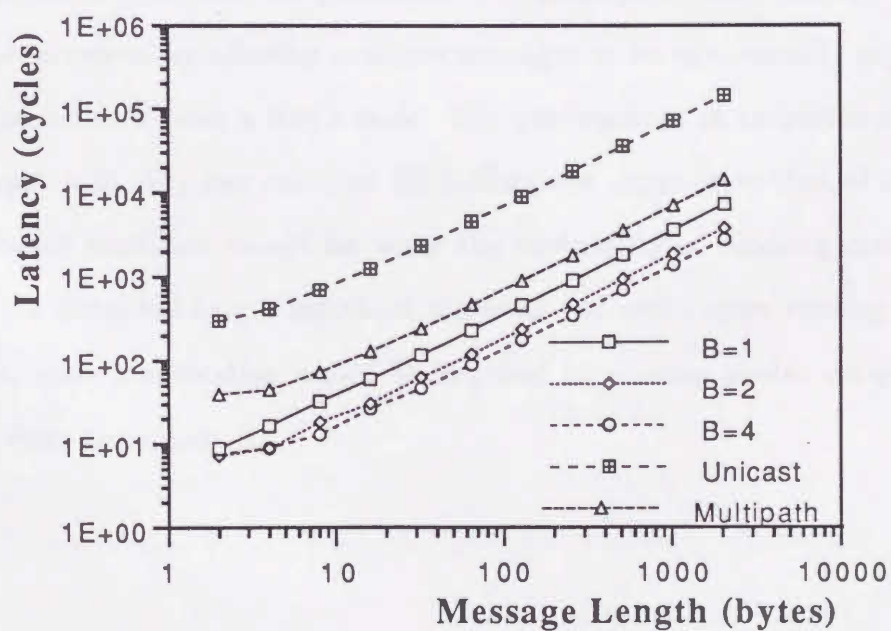
Note that the send latency of restricted-length multicast is independent of the number of destinations of the multicast.

4.3.2 Simulation Results

We have assumed that the multicast set is an 8×8 mesh and the load in the network is simulated by varying the probability of blocking at a single port ($\text{Pr}(b)$) from 0 to 0.7. If multiple outputs are required concurrently, as is the case in restricted-length multicast, then the total probability of blocking (P_t) is given by $P_t = 1 - (1 - P_b)^n$, where n is the number of output ports

requested. A node is chosen at random to initiate the multicast, and the time taken from the initialization of the broadcast until the tail of the last flit arrives at the last node is measured. The results of each multicast method were then averaged over 100 simulations.

Figure 4.3 shows the latency of sending a multicast (in cycles) with L_m fixed at 16 bytes, while varying the probability of blocking from 0 to 0.7. Results for multiple unicast, multi-path, and restricted-length multicasting with buffer sizes of $B=1, 2$ and 4 flits are given. All instances of the restricted-length multicast provided a reduction in latency for $\text{Pr}(b) \leq 0.43$. By increasing the size of one flit buffer so that it can accommodate 2 flits, the probability of blocking must exceed 0.65 before the blocking, due to the requesting of multiple outputs, degrades the performance of restricted-length multicasting to below that of multi-path multicasting. Figure 4.4 illustrates the effect of varying the message length, from 4 bytes to 2048 bytes for a fixed probability of blocking. The header overhead of multi-path multicast is evident in its poor performance for small messages, while unicast performs poorly regardless of message size.

Figure 4.3: Send latency for $L_m = 16$ bytesFigure 4.4: Send latency for $\text{Pr}(b) = 0.5$

4.3.3 Discussion of Results

As expected, the performance of unicast based multicast is much lower than the other methods of multicast investigated here, as each of the unicast messages must wait until the preceding message has left the sending node before it can be sent. This result could be improved upon by adding additional input ports to each node in the system and allowing multiple unicasts to be sent concurrently from a single node. However, as the unicast based multicast generates the most traffic of those methods presented here, this would probably result in an increase in network congestion which would adversely affect the network performance of the entire system.

Both multi-path and restricted-length multicasts exhibited significant speedup when compared to unicast based multicast. As was the case with unicast based multicast, the performance of multi-path based unicast could also be improved by allowing multiple messages to be concurrently injected into the network from a single node. The performance of restricted-length multicast with only two enlarged flit buffers was superior to that of multi-path based multicast except for when the probability of blocking exceeded 0.65. As restricted-length multicast makes use of well known routing algorithms, little modification would be required to existing router designs to allow them to support it.

Chapter 5

Conclusions

Effective communication structures are essential if the full potential of MPP systems is to be realized. The requirements for an interconnection network and its communications structures to be considered effective include freedom from deadlock and livelock, low latency and high throughput, adaptive routing, fault tolerance and support for multicast communication. This dissertation has focused on two solutions to meeting these requirements.

The Tokkyū router was presented and its suitability for use in MPP interconnection networks was demonstrated. Accurate models were developed to predict the switch and buffer performance of Tokkyū routers for varying radix and dimension and these models can be used in the design of routers for networks other than those investigated here. The Tokkyū router meets all of the requirements necessary to be considered effective, as defined in the introduction. Importantly, the support for routing in the presence of faults or network congestion does not compromise the low latency and high throughput of the router. The simulated performance of the Tokkyū router exceeds that of published results for oblivious routers and is equal to or exceeds those reported for other adaptive routers. These performance predictions are es-

pecially encouraging when the simplicity of the control structures required to implement the Tokkyū router are taken into consideration.

The multicast deadlock problem was stated explicitly using a graph theoretical approach which enabled the conditions necessary to avoid deadlock to be defined. Restricted-length multicast was introduced and the implementation of this multicast scheme was examined. Restricted-length multicast was then compared to unicast and multi-path based multicasts. The simulation model allowed the relative merits of restricted-length multicast to be evaluated, and under all but very high simulated congestion conditions restricted-length multicast provided lower latency than unicast or multi-path multicasting. The results therefore indicate that restricted-length multicast provides a good solution to multicast problems such as multicasting to clusters of nodes found in barrier synchronization, multicasting to nearest neighbors and the broadcasting to all of the nodes in the network.

References

- [1] Agarwal, A., "Limits on Interconnection Network Performance", *IEEE Trans. on Parallel and Distributed Computing*, vol. 2, no. 4, pp. 398-412, October 1991.
- [2] Agrawal, D. P., Virenda, J. K., "Evaluating the Performance of Multicomputer Configurations", *IEEE Computer*, vol. 19, no. 5, pp. 23-37, May 1986.
- [3] Annaratone, M., et. al., "The K2 Parallel Processor: Architecture and Hardware Implementation", *Proc. of the 17th Ann. Int. Symp. on Computer Architecture*, pp. 92-101, May 1990.
- [4] Athas, W. C. and Seitz, C. L., "Multicomputers: Message-Passing Concurrent Computers.", *IEEE Computer*, vol. 21, no. 8, pp. 9-24, August 1988.
- [5] Bhuyan, L. N., Yang, Q., Agrawal, D. P., "Performance of Multiprocessor Interconnection Networks", *IEEE Computer*, vol. 22, no. 2, pp. 25-37, February 1989.

- [6] Borkar, S., et. al., "Supporting Systolic Memory Communication in iWarp", *Proc. of the 17th Ann. Int. Symp. on Computer Architecture*, pp. 70-81, May 1990.
- [7] Byrd, G. T. et al., "Multicast Communication in Multiprocessor Systems", in *Proceedings of the 1989 Conference on Parallel Processing*, pp. I196-I200, 1989.
- [8] Chien, A. A., "A Cost and Speed Model for k-ary n-cube Wormhole Routers", In *Proc. of Hot Interconnects 93*, August 1993.
- [9] Chien, A. A. and Kim, J. H. , "Planar-Adaptive Routing: Low-Cost Adaptive Networks for Multiprocessors", *Proc of the 19th Ann. Int. Symp. on Computer Architecture*, pp. 268-277, May 1992.
- [10] Clark, J. and Holton, D. A, *A First Look at Graph Theory.*, Singapore, World Scientific, 1991.
- [11] Cybenko, G. and Kuck, D. J., "Supercomputers: Reinventing the Machine-Revolution or evolution?", *IEEE Potentials*, vol.29, no. 9, pp. 39-41, Sep. 1992.
- [12] Dally, W. J., "Network and Processor Architecture for Message-Driven Computing", in *VLSI and Parallel Processing*, R. Suya and G. Birtwistle eds., Morgan Kaufmann, pp. 140-222, 1989.
- [13] Dally, W. J., "Virtual Channel Flow Control", *IEEE Trans. on Parallel and Distributed Systems*, vol. 3, no. 2, pp. 194-205, March 1992.

- [14] Dally, W. J., "Express Cubes: Improving the Performance of k -ary n -cube Interconnection Networks", *IEEE Trans. on Computers*, vol. 40, no. 9, pp. 1016-1023, September 1991.
- [15] Dally, W. J. and Aoki, H., "Deadlock-Free Adaptive Routing in Multi-computer Networks using Virtual Channels", *IEEE Trans. on Parallel and Distributed Systems*, vol. 4, no. 4, pp. 466-475, April 1993.
- [16] Dally, W. J., et. al., "The Message-Driven Processor: A Multicomputer Processing Node with Efficient Mechanisms", *IEEE Micro*, pp. 23-39, April 1992.
- [17] Dally, W. J. and Seitz, C. L., "The torus routing chip", *Distributed Computing*, vol. 1, pp. 187-196, 1986.
- [18] Dally, W. J. and Seitz, C. L., "Deadlock Free Message Routing in Multiprocessor Interconnection Networks.", *IEEE Transactions on Computers*, vol. C-36, no. 5, pp. 547-553.
- [19] Dally, W. J. and Song P., "Design of a self-timed VLSI multicomputer communication controller", in *Proceedings of the International Conference on Computer Design*, IEEE Computer Society Press, pp. 230-234, October 1987.
- [20] Dongarra, J. J., "Performance of Various Computers Using Standard Linear Equations Software.", *ACM Computer Architecture News*, vol. 20, no. 3, pp. 22-44, June 1992.

- [21] Feng, T., "A Survey of Interconnection Networks", *IEEE Computer*, vol. 14, no. 12, pp. 12-27, December 1981.
- [22] Flavell, A. C., Kanoh, T. and Takahashi, Y., "Mandala: An Interconnection Network for a Scalable Massively Parallel Computer", in *Proc. of the 43rd Annual Convention of the IPSJ*, vol. 6, pp. 91-92, October 1991.
- [23] Flavell, A. C. et. al., "Mandala: An Interconnection Network for a Scalable Massively Parallel Computer", *Technical Report of the IPSJ*, vol. 91, no. 100, pp. 91.101-91.109, November 1991.
- [24] Flavell, A. C. and Takahashi, Y., "Mandala: An Interconnection Network for a Scalable Massively Parallel Computer", in *Proceedings of the 33rd IPSJ Programming Symposium*, pp. 79-90, January 1992.
- [25] Flavell, A. C. and Takahashi, Y., "The MEGA Router: A Hardware Message-Passing Gate Array Router", in *Proceedings of the 45th All Japan Symposium on Information Science*, vol.6, pp. 183-184, October 1992.
- [26] Flavell, A. C. and Takahashi, Y., "Continuum: A Hybrid Time/Space Communications Paradigm for k-ary n-cubes", *Proc. of the International Conference on Parallel Processing 1994*, vol. I, pp. I38-I41, August 1994.
- [27] Flavell, A. C. and Takahashi, Y., "Restricted Length Hardware Multicasting in Multicomputer Networks", *Transactions of the IPSJ*, vol. 36, no. 5, pp. 1228-1238, May 1995.

- [28] Flavell, A. C. and Takahashi, Y., "The Tokkyū Router: A Randomizing Router for k-ary n-cubes", *Proc. of the International Symposium on Parallel and Distributed Supercomputing*, pp. 127-134, September 1995.
- [29] Flavell, A. C. and Takahashi, Y., "Tokkyū: A High-Performance, Randomizing, Adaptive Message Router with Packet Expressway", *IEICE Trans. on Information and Systems*, vol.E78-D, no. 10, pp.1248-1260, October 1995.
- [30] Glass, C. J. and Ni, "Adaptive Routing in Mesh-Connected Networks", in *Proceedings of the 12th International Conference on Distributed Computing Systems*, pp. 12-13, June 1992.
- [31] Hwang, K., *Advanced Computer Architecture*, McGraw Hill, New York, 1993.
- [32] Jesshope, C. R. and Yantchev, J. T., "High Performance Communications in Processor Networks", *Proc of the 16th Ann. Int. Symp. on Computer Architecture*, pp. 150-157, 1989.
- [33] Karol, M. J., et al, "Input Versus Output Queuing on a Space-Division Packet Switch", *IEEE Trans. on Communications*, vol. COM-35, no. 12, pp. 1347-1356, December 1987.
- [34] Kermani, P. and Kleinrock, L., "Virtual Cut-through: A New Communications Switching Technique", *Computer Networks*, vol 3, no. 4, pp. 267-286, 1979.

- [35] Konstantinidou, S. and Snyder, L., "Chaos router: Architecture and Performance", *SIGARCH*, vol. 19, no. 1, pp. 212-221, March 1991.
- [36] Lieserson, C. E., et. al., "The Network Architecture of the Connection Machine CM-5", *Proc. of the 4th Ann. ACM Symp. on Parallel Algorithms and Architectures*, ACM, pp. 272-285, June 1992.
- [37] Lin, X. and Ni, L. M., "Deadlock-Free Multicast Wormhole Routing in Multicomputer Networks." *Proceedings of the 18th Annual International Symposium on Computer Architecture*, pp. 116-125, May 1991.
- [38] Linder, D. and Harden, J., "An Adaptive and Fault-tolerant Wormhole Routing Strategy for k-ary n-cubes", *IEEE Trans. on Computers*, vol. C-40, no. 1, pp. 2-12, January 1991.
- [39] Liu, C. L., *Elements of Discrete Mathematics.*, New York, McGraw Hill, 1977.
- [40] McKinley, P. K., Xu, H., Esafahanian, A. H. and Ni, L. M., "Unicast-Based Multicast Communication in Wormhole-Routed Networks.", Tech. Rep. MSU-CPS-ACS-57, Department of Computer Science, Michigan State University, East Lansing, MI, January 1992.
- [41] Ngai, J. Y. and Seitz, C. L., "A Framework for Adaptive Routing in Multicomputer Networks", *SIGARCH*, vol. 19, no. 1, pp. 6-14, March 1991.
- [42] Ni, L. M., McKinley, P. K., "A Survey of Routing Techniques in Wormhole Networks", *Tech. Rep. MSU-CPS-ACS-46*, Department of Com-

- puter Science, Michigan State University, East Lansing, MI, October 1991.
- [43] Oed, W. and Walker, M., "An Overview of Cray Research Computers including the Y- MP/C90 and the new MPP T3D", *Proc. of the 5th Ann. ACM Symp. on Parallel Algorithms and Architecture*, pp. 271-272, June 1991.
- [44] Panda, D. K., "A Report of the ICPP 94 Panel on - Sea of Interconnection Networks: What's Your Choice?", Department of Computer and Information Science, Ohio State University, Columbus, OH, November 1994.
- [45] Reames, C. C. and Lui, M. T., "A Loop Network for Simultaneous Transmission of Variable-Length Messages", *Proc. of the 2nd Ann. Int. Symp. on Computer Architecture*, pp. 7-12, January 1975.
- [46] Reed, D. A., Fujimoto, R. M., *Multicomputer Networks: Message-Based Parallel Processing*, MIT Press, Cambridge MA, 1987.
- [47] Reed, D. A., Grunwald, D. C., "The Performance of Multicomputer Interconnection Networks", *IEEE Computer*, vol. 20, no. 6, pp. 63-73, June 1987.
- [48] Seitz, C. L., "Concurrent Architectures", in *VLSI and Parallel Computation*, R. Suya and G. Birtwistle eds., Morgan Kaufmann, pp. 1-84, 1990.

- [49] Sullivan, H. and Bashkow, T. R., "A Large Scale, Homogeneous, Fully Distributed Parallel Machine", *Proc. of the 4th Symp. on Computer Architecture*, vol. 5, pp.105-124, Mar 1977.
- [50] Tamir, Y. and Frazier, G. L., "High Performance Multi-Queue Buffers for VLSI Communication Switches", in *Proc. 19th Annual Symposium on Computer Architecture*, IEEE Computer Society Press, pp.343-354, June 1988.
- [51] Xu, H., McKinely, P. K. and Ni, L. M., "Efficient Implementation of Barrier Synchronization in Wormhole-Routed Hypercube Multicomputers", in *Proceedings of the 12th International Conference on Distributed Computing Systems*, pp. 118-127, June 1992.
- [52] Yeh, Y., et al, "The Knockout Switch: A Simple, Modular Architecture for High- Performance Packet Switching", *IEEE Journal on Selected Areas in Communications*, vol. SAC-5, no. 5, pp. 1274-1283, October 1987.
- [53] Zorpette, G., "Supercomputers/Reinventing the Machine - The Power of Parallelism", *IEEE Spectrum* , vol. 29. no. 9, pp. 28-33, September 1992.

