Wormhole Routing Techniques for Directly Connected Multicomputer Systems[1]

Prasant Mohapatra
Department of Electrical and Computer Engineering
201 Coover Hall
Iowa State University
Ames, IA 50011
E.mail: *prasant@iastate.edu*

[1]This research was supported in part by the National Science Foundation through the grants MIP-9628801, CCR-9634547, and CDA-9617375.

Abstract

Wormhole routing has emerged as the most widely used switching technique in massively parallel computers. We present here a detailed survey of various techniques for enhancing the performance and reliability of the wormhole routing schemes in directly connected networks. We start with an overview of the direct network topologies and a comparison of various switching techniques. Next, the characteristics of wormhole routing mechanism are described in detail along with the theory behind deadlock-free routing. The performance of routing algorithms depends on the selection of path between the source and the destination, the network traffic, and the router design. The routing algorithms are implemented in the router chips. We outline the router characteristics and describe the functionality of various elements of the router. Depending on the usage of paths between the source and the destination, the routing algorithms are classified as deterministic, fully adaptive, and partially adaptive. We discuss several representative algorithms for all these categories. The algorithms within each category vary in terms of resource requirements and performance under various traffic conditions. The main difference among various adaptive routing schemes is the technique used to avoid deadlocks. We also discuss a few algorithms based on deadlock recovery techniques. Along with performance, fault-tolerance is essential for message routing in multicomputers, and we thus discuss several fault-tolerant wormhole routing algorithms along with their fault-handling capabilities. These routing schemes enable a message to reach its destination even in the presence of faults in the network. The implementation details of wormhole routing algorithms in contemporary commercial systems are also discussed. We conclude by itemizing several future directions and open issues.

Contents

1 Introduction **3**

2 Direct Network Topologies **5**

3 Wormhole Switching **7**

3.1 Switching Techniques 7

3.2 Virtual Channels 9

3.3 Router Characteristics 10

4 Wormhole Routing Characteristics **12**

4.1 Classifications 12

4.2 Deadlock-Free-Routing Theory 12

4.3 Performance Evaluation 15

5 Deterministic Wormhole Routing **16**

6 Adaptive Wormhole Routing **17**

6.1 Fully Adaptive Algorithms 18

6.2 Partially Adaptive Algorithms 24

6.3 Deadlock Recovery in Fully Adaptive Algorithms 27

7 Fault-Tolerant Wormhole Routing **29**

8 Wormhole Routing in Commercial Systems **33**

9 Conclusions and Open Issues **39**

1 Introduction

Large-scale parallel computers are potential candidates for providing very high computational power. These systems are usually organized as an ensemble of nodes, each with its own processor, local memory, and other supporting devices. The nodes are interconnected using a variety of topologies that can be classified into two broad categories: direct and indirect. In direct networks, each node has a point-to-point or direct connection to some of the other nodes, called neighboring nodes; examples of direct network topologies include hypercube, mesh, and tree. In indirect networks, the nodes are connected to other nodes or a shared memory through one or more switching elements. Examples of indirect networks include crossbar, bus, and multistage interconnection networks.

Direct networks have emerged as a popular architecture for massively parallel computers because of their scalability. The total communication bandwidth, memory bandwidth, and processing capability of the system increases with the number of nodes. Examples of experimental and commercial systems based on direct interconnection network include Intel's iPSC, Touchstone Delta [37] and Paragon [38], Ncube-2/3 [53], Cray T3D [41, 64], MIT J-Machine [56], and Stanford DASH [47]. The nodes of a direct-network-based multicomputer communicate by passing messages through an interconnection network. Neighboring nodes send messages to one another directly while nodes that are not connected directly communicate with each other by passing messages through intermediate nodes. Support hardware is essential to handle the transmission of messages between nodes. In most systems, a router is associated with each node to handle communication-related tasks. Dedicated routers are also used to allow overlapping of computation and communication within each node.

Figure 1 shows the architecture of a generic node consisting of a processor, a local memory, a router, interconnects, and other functional units such as I/O devices. The router has internal channels that connect it to the processor, local memory, or other functional units. The input internal channels are used to absorb messages destined for the host processor. The output internal channels are used by the host processor to send outgoing messages to remote nodes. Some systems use multiple internal channels to avoid communication bottlenecks between the local processor or memory and the router. The multiple internal channels can have either all-port or k-port architecture. In the all-port architecture, every external channel has a corresponding internal channel, thus allowing the node to send and receive on all external channels simultaneously. A k-port architecture has k internal channels, where k is less than the total number of external channels. The internal channels in a k-port router are multiplexed by the external channels, which are used for messages in transit. Usually a crossbar switch is used in the router to connect the input external channels to the output external channels. The control unit is responsible for flow control of the messages traversing the router.

In direct-network-based multicomputers, a task is allocated to a group of nodes that communicate for successful execution of the task. The speed of execution depends on the processor as well as on the communication performance. The latency incurred by a message traversing from

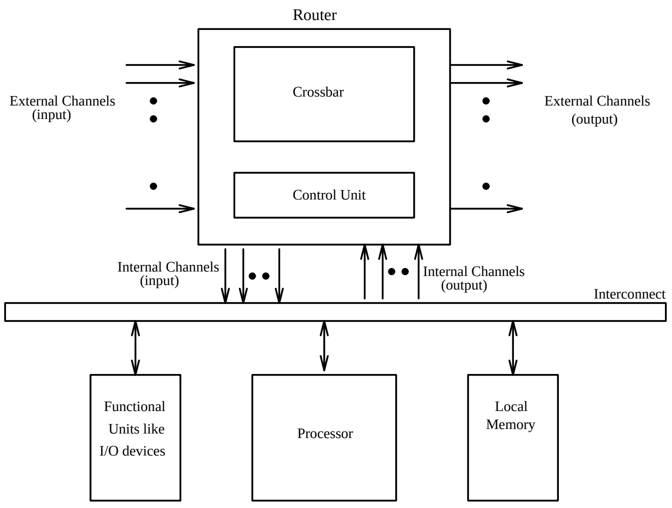

Router
Crossbar
Control Unit
External Channels (input)
External Channels (output)
Internal Channels (input)
Internal Channels (output)
Interconnect
Functional Units like I/O devices
Processor
Local Memory

Figure 1: Node architecture in a multicomputer system.

a source node to a destination node affects the overall performance of the multicomputer system. Because of the interprocessor interactions, the communication latency also affects the granularity of parallelism that can be exploited from the system. Thus it is essential to devise techniques that reduce the communication latency incurred in direct networks.

The communication latency is the most important performance metric in direct networks. It comprises start-up latency, network latency, and the blocking time [55]. The start-up latency is the time required for the system to handle the message at the source and destination nodes and depends primarily on the design of the interface between the local processors and routers. Network latency, defined as the time spent by a message in the network, is computed as the time between the instant when the message head is injected into the network by the source and the instant when the message tail is absorbed by the destination node. Both start-up and network latencies are fixed for a given network. The blocking time of a message is the time spent waiting for a channel currently being used by another message. Thus the blocking time depends on the resource contentions that a message encounters in its path. Blocking time cannot be determined statically, as it depends on the network traffic distribution and the path taken by a message.

The communication latency of direct networks depends on several factors including switching, routing, flow control, and topology. Several switching techniques have been proposed for direct networks. Wormhole switching has emerged as a popular technique and has been used in both commercial and experimental systems. Wormhole switching can be employed in both direct and indirect networks. It is widely used in contemporary multicomputers because of its low latency and requirement of small buffers at the nodes. Theories have been developed for designing cost-effective, efficient, deadlock-free and livelock-free wormhole routing algorithms. Based on these theories, several deterministic and adaptive routing algorithms have been proposed in the literature. In this paper, we survey different techniques of wormhole routing along with the theory behind the design of deadlock-free algorithms. Complementary techniques for deadlock recovery are also described.

We also review fault-tolerant wormhole routing schemes that can route messages in the presence of faults. Details of wormhole routing schemes implemented in several commercial systems are also included.

A preliminary survey on wormhole routing was given by Ni and McKinley [55]. Since then, several advances have been made in wormhole-routed networks. Furthermore, the present report is more comprehensive and discusses several issues not covered by the earlier survey report, such as fault-tolerant routing, deadlock recovery techniques, router designs, and implementation in commercial systems. Topics not discussed in this survey include collective communication and routing in indirect networks. Collective communication could itself be a subject of a survey; indeed, one such work is by McKinley et al. [52]. We focus primarily on direct network topologies such as meshes and k-ary n-cubes because of their widespread adoption in commercial systems. However, we also discuss the implementation of wormhole routing in some recent commercial systems (CM-5 and IBM SP1/SP2) that are based on indirect networks.

The rest of the survey is organized as follows. The properties of direct network topologies are outlined in Section 2. Section 3 discusses various switching techniques along with wormhole switching, which forms the basis of wormhole routing. Virtual channels, flow control mechanisms, and router characteristics are also described in this section. The classification of wormhole routing algorithms and deadlock-free routing theory are presented in Section 4. Section 5 discusses various deterministic wormhole routing algorithms, followed by a discussion of adaptive routing algorithms in Section 6 and of fault-tolerant routing algorithms in Section 7. In Section 8, we discuss the implementation of wormhole routing algorithms in commercial parallel computers, and give concluding remarks and a discussion of open issues in Section 9.

2 Direct Network Topologies

The topology of a network defines how the nodes are interconnected and is generally modeled as a graph in which the vertices represent the nodes and the edges denote the channels. Multidimensional meshes and k-ary n-cubes, the basic topologies used in most parallel computers, are defined as follows [55].

Definition 1: An *n-dimensional mesh* is an interconnection structure that has $k_0 \times k_1 \times \ldots \times k_{n-1}$ nodes, where k_i denotes the number of nodes in the ith dimension. Each node in the mesh is identified by an n-coordinate vector $(x_0, x_1, \ldots, x_{n-1})$, where $0 \leq x_i \leq k_i - 1$. Two nodes $(x_0, x_1, \ldots, x_{n-1})$ and $(y_0, y_1, \ldots, y_{n-1})$ are connected if and only if there exists an i such that $x_i = y_i \pm 1$, and $x_j = y_j$ for all $j \neq i$. Thus the number of neighbors of a node ranges from n to $2n$, depending on its location in the mesh.

Definition 2: A *k-ary n-cube* is defined as an interconnection structure of n dimensions having k nodes in each dimension. Each node in the *k-ary n-cube* is identified by an n-coordinate vector $(x_0, x_1, \ldots, x_{n-1})$, where $0 \leq x_i \leq k - 1$. Two nodes $(x_0, x_1, \ldots, x_{n-1})$ and $(y_0, y_1, \ldots, y_{n-1})$ are connected if and only if there exists an i such that $x_i = (y_i \pm 1)$ *mod* k, and $x_j = y_j$ for all

$j \neq i$. There are wraparound channels in the k-ary n-cubes (specified by the use of modulus in the definition), which are not present in n-dimensional meshes. If k=2, then every node has n neighbors, one in each dimension. If $k > 2$, then every node has $2n$ neighbors, two in each dimension.

The hypercube and torus are two other popular topologies for direct networks. Hypercubes are special cases of an n-dimensional mesh in which $k_i = 2$, for all i, $0 \leq i \leq n-1$; they can be termed 2-ary n-cubes. A k-ary n-cube is called a torus when n=2. Figure 2 shows a three-dimensional (3D) hypercube and a two-dimensional (2D) mesh. A torus can be constructed by connecting the corresponding end nodes of the 2D mesh with wraparound connections.

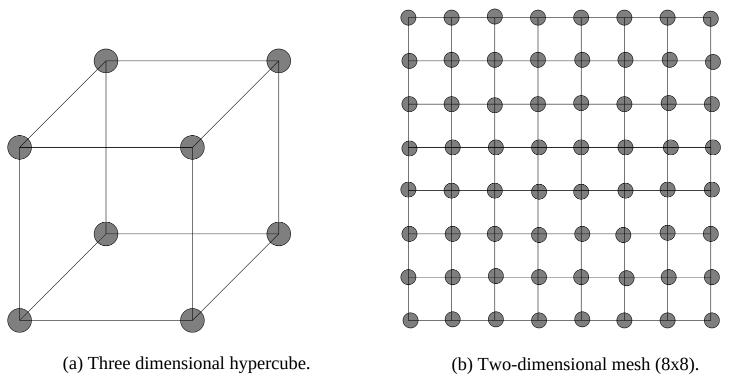

(a) Three dimensional hypercube.
(b) Two-dimensional mesh (8x8).

Figure 2: Topology of a hypercube and a mesh.

Several issues are associated with the mesh, torus, and hypercube topologies. The mesh is an asymmetrical topology in which the node degree depends on its location. Interprocessor communication performance depends on the location of source and destination. The channels near the center of the mesh experience higher traffic density than those on the periphery. The torus and hypercube are symmetrical topologies in which the degree of a node is the same irrespective of its location in the network. Thus, unlike the mesh, all the nodes in tori and hypercubes are identical in connectivity. The network diameter of a mesh is greater than that of the torus, which in turn has a greater diameter than the hypercube for the same number of nodes.

The bisection width of a network is defined as the number of channels that must be removed to partition the network into two equal subnetworks. The bisection width has a significant effect on the interprocessor communication performance [17]. The bisection width (BW) of a $2^n \times 2^n$ 2D mesh, $2^n \times 2^n$ 2D torus, and a $2n$-cube hypercube are $BW_{mesh} = 2^n$, $BW_{torus} = 2^{n+1}$, $BW_{hypercube} = 2^{2n-1}$, respectively. The bisection density, which is the product of the bisection width and the channel width, can be used as a measure of the network cost [55]. For the same cost, the 2D mesh can support wider channels than the 2D torus, which in turn can support wider channels than the hypercube [55]. Thus the channel bandwidth of the three topologies can be expressed as: mesh $>$ torus $>$ hypercube.

In general, low-dimensional meshes are preferred because they have low, fixed-node degrees and fixed length channel wires, which make them more scalable than high-dimensional meshes and k-ary n-cubes. Low-dimensional meshes also have higher channel bandwidth per bisection density and have lower contention and blocking latencies, which results in lower communication latencies and higher hot-spot throughput [17]. Furthermore, two or three topological dimensions are easier to implement in the three physical dimensions. On the other hand, high-dimensional meshes and k-ary n-cubes have lower diameters, which shortens the path lengths. High-dimensional topologies also have more paths between pairs of nodes, which permits more adaptivity and fault tolerance.

A class of shuffle networks known as de Bruijn (dB) graphs have become popular recently. They are suitable for large network and can be defined for any number of nodes, including prime numbers [61]. For a specific node degree, dB networks, in most cases, have the smallest diameter compared to the contemporary network topologies. Formally, a unidirectional dB network can be defined as follows [61].

Definition 3: An r-radix unidirectional de Bruijn digraph dBD(r, r^m) has the total number of nodes $N = r^m$ and the address of a node X is represented as $(x_{m-1}, x_{m-2}, ..., x_0)$ where $x_i \in \{0, 1, ..., (r-1)\}$ for $0 \leq i \leq m-1$. Its neighboring nodes are $(x_{m-2}, x_{m-3}, ..., \alpha)$, where $\alpha = 0, 1, ..., r-1$.

Several other topologies based on Caley graphs have been also proposed [5]. However, here we focus primarily on k-ary n-cubes and multidimensional meshes. Wormhole routing techniques for dB networks and other topologies based on the Caley graphs are reported in [11, 57].

3 Wormhole Switching

Nodes in a direct network communicate by passing messages from one node to another. A message may be divided into one or more equal or variable-size packets. A packet is the smallest unit of information that contains routing and sequencing information. In this section, we discuss various switching techniques used or proposed for multicomputer systems.

3.1 Switching Techniques

In most multicomputer systems, a message enters the network from a source node and is switched or routed towards its destination through a series of intermediate nodes. Four types of switching techniques are usually used for this purpose: circuit switching, packet switching, virtual cut-through switching, and wormhole switching.

In circuit switching, a dedicated path is established between the source and the destination before data transfer initiates. Once the data transfer is initiated, the message is never blocked. As the channels creating the path are reserved exclusively, buffering of data is not required. On the other hand, establishing the path requires significant overhead: during the data-transmission phase, all channels are reserved for the entire duration of message transfer. Circuit switching thus degrades performance and is no longer used in commercial multicomputer systems.

In packet switching, a message is divided into packets that are independently routed towards its destination. The destination address is encoded in the header of each packet. The entire packet

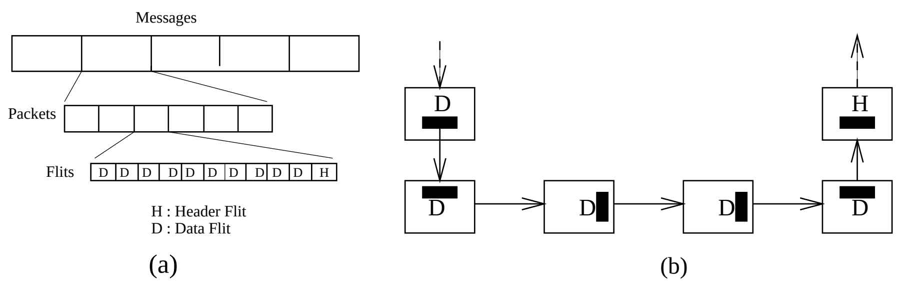


Figure 3: Message format and routing in wormhole switching.

is stored at every intermediate node and then forwarded to the next node in its path. The main advantage of packet switching is that the channel resource is occupied only when a packet is actually transferred. Each packet contains the routing information and alternative paths can be selected upon encountering network congestion or faulty nodes. The major drawback of packet switching is that, since the packet is stored entirely at each intermediate node, the time to transmit a packet from source to destination is directly proportional to the number of hops in the path. Furthermore, at each intermediate node, we need buffer space to hold at least one packet.

In order to reduce the time to store the packets at each node, Kermani and Kleinrock introduced a technique called virtual cut-through [40] in which, while routing toward its destination, a message is stored at an intermediate node only if the next channel required is occupied by another packet. Now, the distance between the source and destination has little effect on communication latency. In an extreme case, when a message encounters blocking at all the intermediate nodes, the virtual cut-through technique reduces to packet switching. The disadvantage of the virtual cut-through technique is its implementation cost: each node must provide sufficient buffer space for all the messages passing through it, and because multiple messages may be blocked at any node, a very large buffer space is required at each node. This implementation constraint limits the use of virtual cut-through technique.

Wormhole switching is a variant of the virtual cut-through technique that avoids the need for large buffer spaces. In wormhole switching, a packet is transmitted between the nodes in units of *flits*, the smallest units of a message on which flow control can be performed. The header flit(s) of a message contains all the necessary routing information and all the other flits contain the data elements. The flits of the message are transmitted through the network in a pipelined fashion. Since only the header flit(s) has the routing information, all the trailing flits follow the header flit(s) contiguously. Flits of two different messages cannot be interleaved at any intermediate node. Successive flits in a packet are pipelined asynchronously in hardware using a handshaking protocol. When the header flit is blocked, then all the trailing flits occupy the buffers at the intermediate nodes. The general format of a message and the units of packets and flits are shown in Figure 3(a). Figure 3(b) shows the routing mechanism using wormhole switching, where the header flit H contains the destination address and the data flits D follow H contiguously in a pipelined fashion.

The main advantage of wormhole switching derives from the pipelined message flow since transmission latency is insensitive to the distance between the source and destination. Moreover, since the message moves flit by flit across the network, each node needs to store only one flit. Some implementations, however, require storage of multiple flits at each node to improve routing performance. The reduction of buffer requirements at each node has a major effect on the cost and size of multicomputer systems.

The main disadvantage of wormhole switching comes from the fact that only the header flit has the routing information. If the header flit cannot advance in the network due to resource contention, all the trailing flits are also blocked along the path and these blocked messages can block other messages. This chained blocking can also lead to deadlock where messages wait for each other in a cycle and hence no message can advance any further. Prevention of deadlock is one of the main issues in wormhole switching, and is usually accomplished by a suitable choice of routing function that selectively prohibits messages from taking all the available paths, thus preventing cycles in the network. Selection of a routing algorithm is thus a major issue in wormhole-switched networks.

3.2 Virtual Channels

The primary problem associated with wormhole routing is the blocking of messages. A message may be blocked behind another message destined for a node in a different direction. In Figure 4(a), message B whose destination is in the east direction, is blocked behind message A, which is blocked while traveling in the south direction. This type of blocking reduces network performance drastically and can also lead to deadlock. Virtual channels can be used in wormhole-switched networks to prevent deadlock and reduce the effects of chained blocking [16]. A virtual channel is a logical abstraction of a physical channel. All the virtual channels associated with a physical channel have individual flit buffers and are time-multiplexed for message transmission using the physical channel. Virtual channels dissociate the buffers associated with the channels from the actual physical channels. Figure 4(b) shows two virtual channels associated with a physical channel in one direction. Even if message A is blocked by some other message down the path, message B can move forward using the other virtual channel. Virtual channels reduce the effect of blocking and are used widely in multicomputer systems to improve performance as well as to design deadlock-free routing algorithms.

Virtual channels are implemented with a single flit or multiple flits along with an appropriate flow-control protocol. The flow-control protocol of a network determines how resources (buffers and channel bandwidth) are allocated and how message collisions are resolved. A message collision occurs when a packet cannot proceed because the buffer it needs is held by another message. The flow control strategy allocates buffer and channel bandwidth to flits. Because flits have no routing or sequencing information, the allocation must be done in a manner that keeps the flits associated with a particular message together.

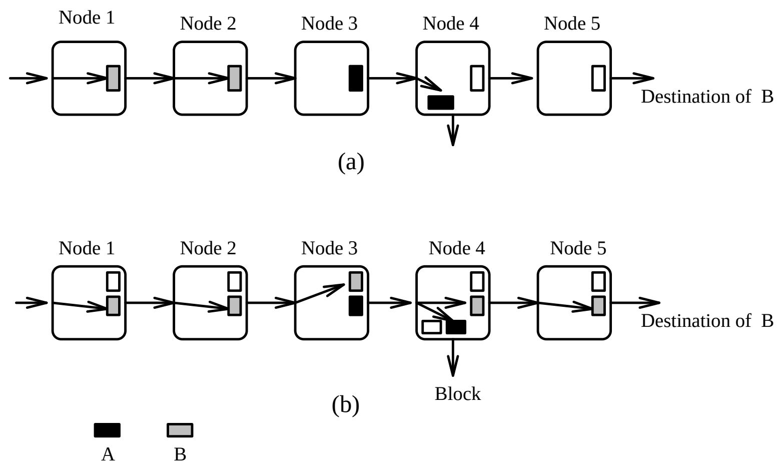
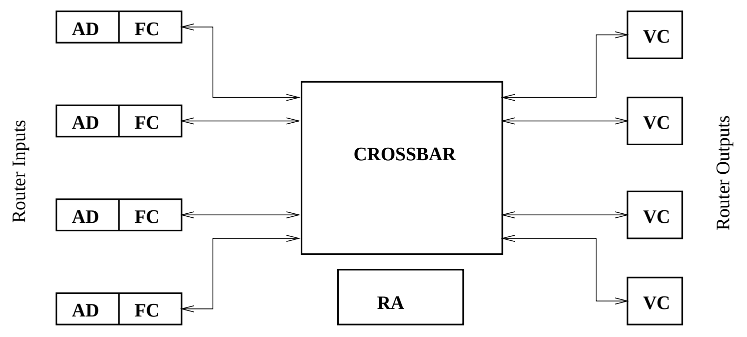


Figure 4: Performance improvement using virtual channels: (a) Packet B is blocked behind packet A while all physical channels remain idle, (b) Virtual channels provide an additional buffer, allowing packet B to pass the blocked packet A.

3.3 Router Characteristics

As wormhole routing is implemented in hardware, the design and characteristics of the router unit are significant in determining routing algorithm performance. A detailed study of various design issues related to the routers is reported in Chien [14]. The primary functions of wormhole routers include switching, routing, flow control, multiplexing physical and virtual channels, inter-chip signaling and clock recovery. Figure 5 depicts various components of a generic wormhole router [14]. The essential components of a wormhole router and their functionality are described below.



AD: Address Decoder, FC: Flow Control, RA: Routing Arbitration, VC: Virtual Channel Controller

Figure 5: Block diagram of a wormhole router.

Crossbar: Crossbar switches enable the switching of router inputs to outputs. Single or multiple crossbar switches are used, based on cost-performance trade-offs. The size of the crossbar may be proportional to the number of inputs and outputs.

Routing Arbitration (RA) Logic: This logic unit does arbitration for the router outputs. It chooses the path and connects and disconnects the input to an appropriate output based on the routing algorithm and network status. The complexity and latency incurred in the routing arbitration logic are proportional to the degree of freedom (number of possible choices) of the routing algorithm.

Address Decoder (AD): This unit checks the message header and generates the set of possible routes based on the routing algorithm. AD also computes and updates the header information.

Flow Control (FC) Units: These units are implemented using control logic and buffers and perform the flow control between the routers. The buffers hold the flits while flow-control signals are being exchanged between the routers. The buffers are also used for holding the flits if a message is blocked.

Virtual Channel (VC) Controllers: Virtual channel controllers are implemented in those routers that support virtual-channel flow control. These controllers multiplex the physical channels to provide a set of independent virtual channels that enables the flow of different messages. The complexity and latency of the controllers increase with the number of virtual channels due to increased buffering and arbitration requirements.

Messages flow through the routers as follows. Messages arriving at the router inputs encounter the address decoder, which checks the message header and generates a set of possible outputs for the message. The routing arbitration logic tries to match all the inputs to the output channels based on the output sets generated by the address decoder. If a suitable match is not found for a message then it is blocked. Once an appropriate output has been selected, the switch connection is made for the entire message. The connection is terminated following the last flit.

The performance of wormhole routers can be characterized by two attributes: internal router latency and bandwidth. Internal router latency, defined as the time to create a valid path through the router, has contributions from the following factors: address decoding, arbitration, updated header selection, crossbar switching, and virtual channel control delay. The selection policies used at the routers also have a significant impact on network routing performance. Several input and output selection policies have been compared by Glass and Ni [33]. A router's channel bandwidth depends on the size of the flow-control unit (flit) and the time required for a flow-control operation. Though higher bandwidth can be achieved by increasing the flit size, such a choice increases router complexity by introducing separate logical and physical signaling rates, requiring additional buffering, slowing backpressure, and increasing routing latency. The internal flow-control latency limits the flit flow rate in the network channels. Flits are units of resource multiplexing, so flow-control latency determines the network's ability to share internal connections and external channels among different packets. The unit of multiplexing directly affects the responsiveness of the network. In addition, flow-control speed determines the amount of buffering needed. The flow-control delay includes the latency of flow-control units, the crossbar switch delay, and the virtual channel controller delay.

4 Wormhole Routing Characteristics

The insensitivity to distance, pipelined flow of messages, and small buffer requirements are some of the main advantages of the wormhole routing scheme. Its primary disadvantage is coupled with the pipelined flow of messages that introduces blockings that can lead to deadlock. In this section, we present the classification of wormhole routing schemes and then formalize the theory behind deadlock-free routing algorithms. We also survey the performance parameters used to evaluate wormhole routing schemes.

4.1 Classifications

Routing algorithms can be classified with respect to several characteristics. They can be classified as source routing or distributed routing according to the location of routing decisions. In source routing, the entire path for message routing is decided at the source node before the message is sent. Each message carries the complete routing information in its header, thus increasing the overall message size. In distributed routing, the routing decisions are made at the intermediate nodes through which the message traverses. Upon receiving a packet, each router decides whether to deliver it to the local processor or forward it to a neighboring router. The routing algorithm helps in deciding which neighbor the packet should be sent to.

Routing can be also classified as deterministic or adaptive based on the path selection process. In deterministic routing, the path is determined by the current and destination addresses. Deterministic routing, also called oblivious routing, provides only one path from a source to a destination. Adaptive routing, on the other hand, provides multiple paths from the source to destination, and the path taken by a particular message depends on network conditions and the routing algorithm.

The routing algorithm can be minimal or nonminimal. In minimal routing, the message is routed through one of the shortest paths between the source and the destination. The message traverses closer to its destination after every hop. In nonminimal routing a message can take any path between the source-destination pair, and thus might take a longer path because of congestion or faults in the minimal paths. While designing nonminimal routing algorithms, care should be taken to avoid livelock situations where a message continues to be routed through the network but never reaches its destination.

4.2 Deadlock-Free-Routing Theory

Deadlock occurs when a set of messages is blocked forever in the network. In such a situation, the packets are holding certain resources and requesting other resources held by other messages involved in the deadlock configuration. Figure 6 shows deadlock in a two-dimensional mesh. Four messages are being routed from sources S1, S2, S3, and S4 to destinations D1, D2, D3, and D4, respectively. All messages are thus waiting for a channel that will never be available, resulting in a deadlock.

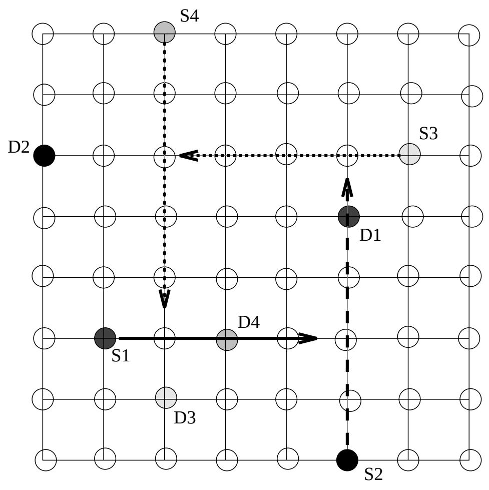


Figure 6: A deadlock situation involving four messages.

Recovery and avoidance are two ways to handle the deadlock problems associated with wormhole routing. Deadlock recovery requires deadlock detection ability as well as preemption of messages. Deadlock detection mechanisms increase the complexity of interprocessor communications, and preemption of messages increases the latency. Thus, most routing algorithms are designed to avoid the possibility of deadlock configurations. Before we discuss the theory behind deadlock-free routing, some terminology is needed. We employ the terms and definitions introduced by Duato [24].

Definition 4: A *routing function* $R : N \times N \rightarrow \rho(C)$, where $\rho(C)$ is the power set of C, supplies a set of alternative output channels to send a message from the current node n_c to the destination node n_d. If size of $\rho(C)$ is always 1, the function R is deterministic, otherwise it is an adaptive routing function. For a given interconnection network, R is *connected* iff, for any pair of nodes x, $y \in N$, it is possible to establish a path $P(x,y) \subset \rho(C)$ between them using channels supplied by R.

Definition 5: A *routing subfunction* R_1 for a given routing function R is a routing function that supplies a subset of channels supplied by R. Thus R_1 restricts the routing options supplied by R. The set of all the channels supplied by R_1 is $C_1 = \cup_{\forall x,y \in N} R_1(x,y)$.

Definition 6: Given an interconnection network, a routing function R, a routing subfunction R_1 and a pair of channels $c_i, c_j \in C$, there is a *direct dependency* from c_i to c_j iff c_j can be used immediately after c_i by messages destined for some node x.

Definition 7: Given an interconnection network, a routing function R, a routing subfunction R_1 and a pair of channels c_i and c_j supplied by R_1 for some destinations, there is an *indirect dependency* from c_i to c_j iff it is possible to establish a path from s_i to d_j for messages destined for some node x. c_i and c_j are the first and last channels in that path and the only ones supplied by R_1. Thus, c_j can be used after c_i by some messages. As c_i and c_j are not adjacent, some other channels not supplied by R_1 are used between them.

Definition 8: A *channel dependency graph* D for a given interconnection network I and routing function R is a directed graph, $D = G(C, E)$. The vertices of D are the channels of I. The arcs of D are the pairs of channels (c_i, c_j) such that there is a direct dependency from c_i to c_j.
Definition 9: An *extended channel dependency graph* D_E for a given interconnection network I and routing subfunction R_1 of routing function R is a directed graph, $D_E = G(C_1, E_E)$. The vertices of D_E are the channels supplied by the routing function R_1 for some destinations. The arcs of D_E are the pairs of channels (c_i, c_j) such that there exists a direct, indirect, direct cross-dependency or indirect cross-dependency from c_i to c_j.

The following assumptions are also required to derive necessary and sufficient conditions for deadlock freedom [19][25]:

1. A node can generate messages destined for any other node.

2. A message arriving at its destination node is eventually consumed.

3. A node can generate messages of arbitrary length. Messages are generally longer than a single flit.

4. An available queue may arbitrate among messages that request that queue, but may not choose among waiting messages.

5. Once a queue accepts the first flit of a message, it must accept the remainder of the message before accepting any flits from another message.

6. A queue cannot contain flits belonging to different messages. After accepting a tail flit, a queue must be emptied before accepting another header flit.

7. In deterministic routing, a route taken by a message is determined by its destination only. For adaptive routing, the route taken by a message depends on its destination and the status of the output channels.

Dally and Seitz have proposed the following necessary and sufficient conditions for deadlock-free deterministic routing [19].
Theorem 1: A deterministic routing algorithm for an interconnection network I is deadlock-free iff there are no cycles in the channel-dependency graph D.

Although Theorem 1 is also a sufficient condition for deadlock-free adaptive routing, it is not a necessary condition. In adaptive routing, even if there are cycles in the channel-dependency graph, the routing can be deadlock-free if there exists at least one escape path with no cyclic dependency. The necessary and sufficient conditions for deadlock-free adaptive routing proposed by Duato [24] can be stated as follows.
Theorem 2: A coherent, connected and adaptive routing algorithm R for an interconnection network I is deadlock free iff there exists a routing subfunction R_1 that is connected and has

no cycles in its extended channel dependency graph D_E. The proof of Theorem 2 and related definitions are discussed in detail in [24].

We analyze an example to illustrate the terminology defined in this section and the application of Duato's theorem [24]. Consider a unidirectional ring with four nodes ni, $0 \leq i \leq 3$ (Figure 7a). There are two channels in each direction except north. Let CAi, $0 \leq i \leq 3$, and CHi, $0 \leq i \leq 2$ be the outgoing channels from node ni. The routing algorithm R can be stated as follows: If the current node ni is equal to the destination node nj, consume the message. Otherwise, use either CAi, $\forall j \neq i$ or CHi, $\forall j > i$. Thus CAi channels can be used to forward messages to all the destinations, but CHi channels can be only used if the destination is higher than the current node.

Consider a routing subfunction R_1 that is equal to R, except that CA0 cannot be used and CA1, CA2 can be used only to forward messages to destinations lower than the current node. The routing subfunction R_1 is connected because messages at node ni destined for a higher node will be forwarded through CHi and messages destined for a lower node will be sent across CAi. Figure 7(b) shows the extended channel-dependency graph for R_1. As there are no cycles in the graph, we can conclude that R is deadlock-free.

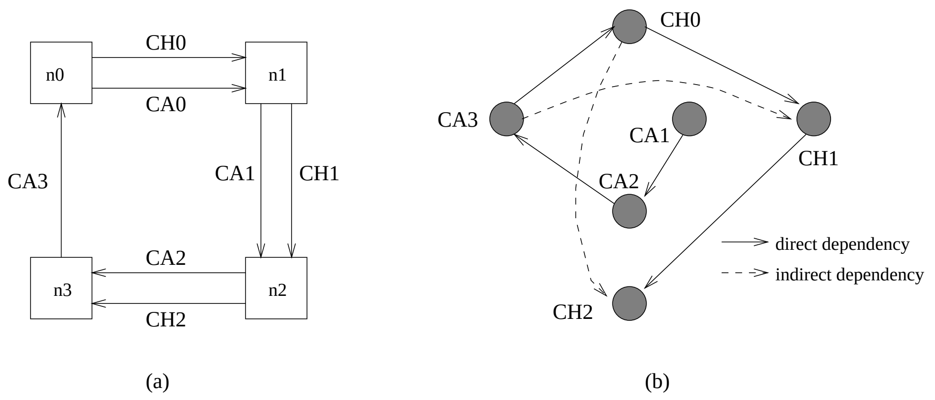


Figure 7: (a) Network for the example, (b) its extended channel-dependency graph.

Alternative theoretical formulations of necessary and sufficient conditions for deadlock-free adaptive routing are proposed by Lin, McKinley, and Ni [49] and Schwiebert and Jayasimha [63].

4.3 Performance Evaluation

Routing algorithms are evaluated primarily by measures of average message latency and average system throughput. The hardware requirements in terms of the buffer size required per node and the number of virtual channels per physical channel are also used in comparing routing algorithms.

Message latency is the time between generation of its header flits and arrival of the tail flit at its destination. Throughput is measured as the average number of packets that finish routing per unit time. Both these parameters depend on the communication pattern, i.e. the distribution of source-destination pairs, which is largely application-dependent. Usually a few predetermined patterns are used to evaluate algorithm performance. The most commonly used communication patterns are uniform, hotspot, transpose or center reflection, and local. With uniform distribution, a source node sends messages to all other nodes with equal probability. In hotspot distribution, the probability that a message is sent to some nodes (called hotspot nodes) is higher than to other nodes. In the center reflection or transpose pattern, messages are directed to the diagonally opposite nodes. In the local traffic pattern, a message is sent only to the nodes within a certain neighboring region; this pattern reflects normal application behavior in which there is a large degree of locality in the internode communication. The overall performance of the algorithm under these varied patterns usually reflects its behavior for any given application.

Routing algorithms are also compared on the basis of their flit buffer and virtual channel requirements . Intuitively, algorithms that use more virtual channels should give better results, but detailed analysis has shown that the overhead associated with the virtual channels is usually very high and that the actual performance may even degrade [8].

Most performance studies on wormhole routing have resorted to simulation and measurements. Development of analytical models for performance evaluation is difficult because of the multiple and simultaneous resource possessions as well as the chained blockings during pipelined routing. However, approximate analytical models based on simplifying assumptions can give reasonable performance estimates [2, 4, 17, 23, 42, 43].

5 Deterministic Wormhole Routing

In deterministic routing, the path from source to destination is determined by the current node address and the destination node address: for the same source-destination pair, all packets follow the same path. Deadlocks are avoided in deterministic routing by ordering the channels a message needs to traverse. Messages traverse the channels either in ascending or in descending order, avoiding cycles in the channel dependency graph.

Dimension-order routing [19] is a deterministic routing scheme in which the path selected traverses network dimensions in sequence. The network dimensions are arranged in a predetermined monotonic order. A message traverses channels in the lowest or the highest dimension with non-zero displacement until that dimension displacement reduces to zero; then it traverses in the next dimension, continuing until it reaches its destination. As the messages never traverse in reverse direction of the dimension ordering, cycles cannot form and deadlock-free routing is guaranteed.

Dimension order routing in hypercubes, also called e-cube routing, is minimal in nature. The nodes of an n-cube are represented by n-bit binary addresses. The destination address for a message is encoded in its header. When a node receives a message, the destination address is bit-XORed with the current node address. If the result of the XOR operation is zero, the message is absorbed by the processor in the current node; otherwise, the message is forwarded in the dimension corresponding to the rightmost (or leftmost if using reverse ordering) 1 in the result. For example, a message originating from source node 0010 in a 4-cube to destination node 1101 traverses the nodes in the sequence $0010 \rightarrow 0011 \rightarrow 0001 \rightarrow 0101 \rightarrow 1101$.

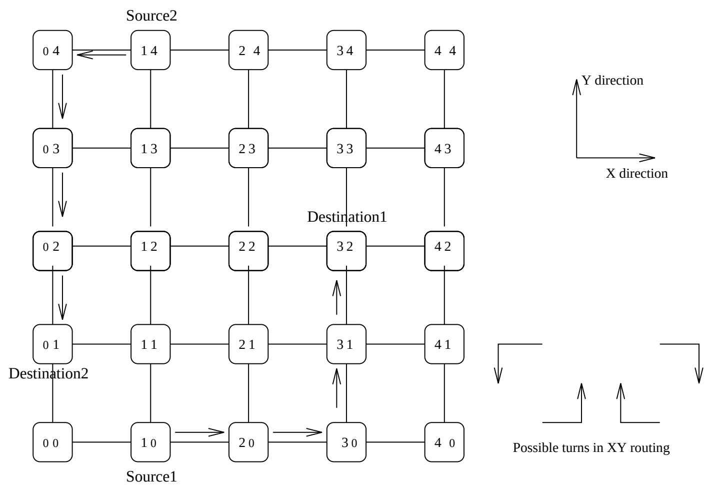


Figure 8: Dimension order routing in a 2D mesh.

Dimension-order routing in two-dimensional meshes is called XY routing, which is minimal in nature. The two dimensions of a mesh are labeled as X and Y. A message is first routed in the X direction completely and then in the Y direction. Figure 8 clearly indicates that cycles cannot be formed with XY routing, and hence it is deadlock-free. Examples of routing paths between two source-destination pairs are also shown in Figure 8. In multidimensional meshes, similarly routing is completed in one dimension before proceeding to the next dimension. All the dimensions are ordered and the routing is done through a predetermined sequence of dimensions till the message reaches its destination.

Dimension-order routing in k-ary n-cubes is not minimal in nature. Because of the wraparound connections, messages may get involved in deadlocks while routing through the shortest paths. In fact, messages being routed along the same dimension (a single dimension forms a ring) may be involved in a deadlock due to cyclic dependency. However, nonminimal deadlock-free deterministic routing algorithms can be developed for k-ary n-cubes by restricting the use of certain edges so as to prevent the formation of cycles [19]. Minimal deadlock-free dimension-order routing can also be implemented in k-ary n-cubes using virtual channels [19].

Dimension-order routing generally distributes the shortest paths throughout the network. It is thus well-suited for uniform traffic distribution. For asymmetrical workloads, some channels are more overloaded than others. As the algorithm restricts message routing to a fixed path, it cannot exploit possible multiple paths between source-destination pairs during congestion or faults.

6 Adaptive Wormhole Routing

Deterministic algorithms provide one and only one path between any source-destination pair. To avoid network congestion and enhance fault tolerance, it is preferred that the routing algorithm provide alternative paths to the message. Algorithms that adapt to the network and traffic conditions are called adaptive routing algorithms and are classified as fully or partially adaptive depending on whether they allow all possible paths between the source and destination or only a subset of them. Although messages can take nonminimal adaptive paths [45], we discuss here only minimal adaptive routing.

6.1 Fully Adaptive Algorithms

Fully adaptive algorithms let a message use all possible physical paths between source and destination. Deadlock is usually avoided in the fully adaptive routing schemes by using virtual channels (VCs). Here we classify fully adaptive routing algorithms on the basis of the number of VCs required per physical channel (PC) for a k-ary n-cube network. Schemes that employ more than two VCs per PC include [50] and [18], the routing algorithms reported in [12, 25, 66, 68] use exactly two VCs per PC, and algorithms using fewer than two VCs are reported in [32, 36, 62].

Algorithms requiring more than two VCs per PC:

Linder and Harden have extended the concept of virtual channels to virtual interconnection networks [50] in order to develop fully adaptive and fault-tolerant routing algorithms. A virtual interconnection network has two components. First, the virtual topology is defined by identifying the virtual nodes and channels of the network and specifying which nodes are the sources and destinations of each channel. Second, the edges of the channel-dependency graph are listed to specify the connections available to the routing functions. The scheme uses a separate virtual interconnection network for each of the possible turns a message might make, and the number of virtual channels required depends on the total number of virtual networks. Message routing within a virtual network is deadlock-free and messages are routed through the virtual networks in a predefined order. The authors have examined three k-ary n-cube topologies: unidirectional, torus-connected bidirectional, and mesh-connected bidirectional.

In a unidirectional k-ary n-cube, the virtual nodes in the virtual interconnection network are identified by their level. To eliminate deadlock due to wraparound connections, the physical networks are split into multiple levels. A flit travels on a new level each time it crosses a wraparound connection. In torus-connected bidirectional k-ary n-cubes, levels are also used to break the cycles formed by the wraparound connections. Because of the bidirectional nature of the physical network, additional cycles due to multidimensional loops can be formed. To break these cycles, the bidirectional networks are split into several virtual networks similar to the unidirectional networks. The deadlock avoidance mechanism in mesh-connected bidirectional networks is similar to that for torus-connected bidirectional k-ary n-cubes, but no levels are required in mesh-connected systems because of the absence of wraparound connections. The disadvantage of the algorithms proposed by Linder and Harden is that they require a large number of virtual channels: in general, the algorithm requires 2^{n-1} subnetworks with $n+1$ levels per subnetwork for a k-ary n-cube network.

Dally and Aoki have proposed an adaptive routing scheme based on the concept of dimension reversal [18]. A message is allocated to virtual channels using a count of dimension reversals (DR). All messages start with a DR of zero. Each time a message goes to a lower dimension, the DR of a message is incremented. Two allocation algorithms, static and dynamic, were proposed. The static algorithm separates the virtual channels into classes numbered zero to r, where r is the maximum number of dimension reversals permitted. Messages with $DR < r$ are allowed to route freely only in a virtual channel of class DR. If a message has DR= r, it must be routed in dimension order in the virtual channels of class r. The dynamic algorithm routes messages in any direction with no limit on the number of dimension reversals. The virtual channels are divided into two classes, adaptive and deterministic. Messages are routed first on the adaptive channels. A message with a higher DR cannot wait for a channel labeled with a lower DR; if all channels with equal or lower DR are occupied, a message must change to the deterministic channels and is not allowed to use the adaptive channels again.

The hop-based adaptive routing scheme recently proposed by Boppana and Chalasani [11] shows that the approach taken by Dally and Aoki is a special case of the hop-based schemes. Boppana and Chalasani have developed new wormhole routing algorithms based on store-and-forward (SAF) hop schemes that can be employed for a variety of network topologies that include k-ary n-cubes, multidimensional meshes, dB networks, and Caley graphs. In hop-based schemes, the class of a message at any time depends on the hops it has taken up to that point. A hop-based adaptive routing scheme called *negative-hop* (NHOP) is based on the NHOP SAF algorithm [35]. In the NHOP SAF algorithm, the network is partitioned into several subsets such that none of the subsets contains two adjacent nodes. All subsets are labeled and the nodes in the subsets are marked or colored with the label number. A hop is considered negative if it is from a node with a higher label to a node with a lower label; otherwise, it is non-negative. A message occupies buffers corresponding to its label number. The algorithm can be modified for wormhole routed k-ary n-cubes as follows [11]:

```
Algorithm NHOP
Initialize: current-class = 0; current-host = source of the message;
If (current-host $\neq$ destination) then {
1. If label of current-host is 0 or is identical to the previous-host, then increment current-class by 1.
2. Select any neighboring node that is a shortest path to destination as next-host.
3. Reserve the virtual channel of class current-class.
4. If the virtual channel is available, change current-host to previous-host, next-host to current-host, and route the message; otherwise go to 2.
} Else consume the message.
```

Figure 9 shows the NHOP routing for a message from (2,2) to (0,0) in a 4x4 mesh using four virtual channels. The second and fourth hops are negative hops, and the message class is incremented after the second hop.

The number of virtual channels needed for a k-ary n-cube network with the NHOP wormhole routing algorithm is $1 + \lfloor n \lceil k/2 \rceil \rfloor$ [11]. Boppana and Chalasani [11] also analyze buffered wormhole routing and describe its implementation in IBM SP1 and SP2.

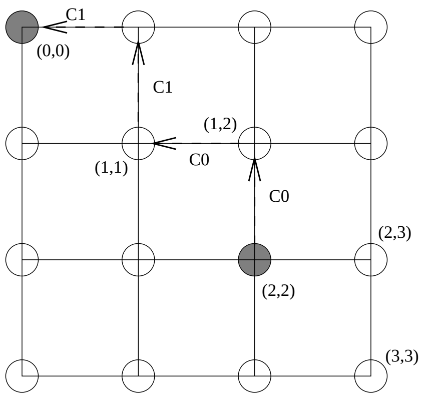


Figure 9: NHOP routing in a 4x4 mesh [11].

Gravano et al. have proposed a fully adaptive minimal routing algorithm called *-Channels [36] that needs only five virtual channels per bidirectional link for n-dimensional torus networks. The *-Channel algorithm involves two subnetworks, one composed of star channels and one of nonstar channels. The star channels are used for dimension-order oblivious routing. The nonstar channels are used when taking any of the turns that are not allowed by the oblivious routing scheme. The star channels implement a complete oblivious subnetwork that acts as a “release valve” or “drain” for the subnetwork built from the non-star channels. Figure 10 shows the paths available in a 2D torus between the nodes (1,0) and (4,2) and between the nodes (6,3) and (1,6) with the *-Channel algorithm. The links have different types of virtual channels associated with them. The star channels with prefix i,+,1 are used for dimension-order routing; channels with prefix i,+,0 are used in correcting the dimension and making a wraparound along the X-dimension.

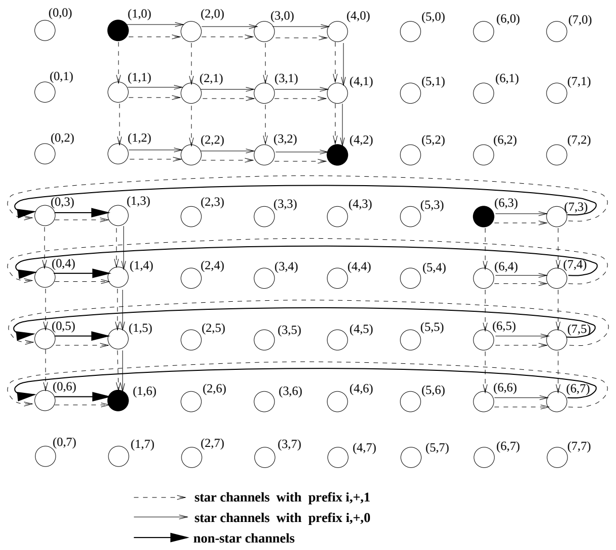

star channels with prefix i,+,1
star channels with prefix i,+,0
non-star channels

Figure 10: Message routing using the *-Channel algorithm in a 2D torus.

Gravano et al. [36] have also proposed an algorithm called 4-classes for bidirectional torus networks. The algorithm divides all source-destination pairs into four classes and creates a virtual network for each class using all minimal paths in the network. It requires eight virtual channels per bidirectional physical link and the routing scheme is as follows.

Consider a two-dimensional torus with dimensions X and Y, and a message with source node (x, y) and destination node (x', y'). A minimal path from (x, y) to (x', y') is built in such a way that the message must travel through at most $\lfloor k/2 \rfloor$ links along dimension X and through at most $\lfloor k/2 \rfloor$ along the dimension Y. The correct orientation along each dimension must be chosen to find such minimal paths. If k is odd, there is only one possible orientation of each dimension for the minimal paths. If k is even and a message is $k/2$ steps away from its destination along dimension X, then either orientation along the X dimension can be taken in order to follow minimal paths toward the message destination. This choice is independent of the dimensions. Therefore, the set of minimal paths between any pair of nodes is determined by a correct choice of the direction in which to change each of the dimensions.

Algorithms requiring exactly two VCs per PC:

Duato has developed a theoretical background for deadlock-free adaptive routing algorithms for wormhole networks [25] and has proposed fully adaptive routing algorithms that split a physical channel into virtual channels. At least two virtual channels are required to support adaptivity. The technique increases the number of valid alternative paths for a message without increasing the number of physical channels. Duato gives the following systematic methodology for designing deadlock-free adaptive routing algorithms.

1. Define a deadlock-free minimal-path-connected deterministic or adaptive routing function R_1 for the network.

2. Split each physical channel into a set of additional virtual channels. Define a new routing function R that can use any of the new channels belonging to a minimal path or the channels supplied by R_1.

3. Verify that the extended channel dependency graph for R_1 is acyclic. If it is, then the routing algorithm is valid. Otherwise, it must be discarded and the steps are repeated.

Using his proposed methodology, Duato has developed a fully adaptive routing algorithm for hypercube computers [25]; the methodology can be also applied to other topologies including meshes and k-ary n-cubes [27]. The steps involved in the design process are explained as follows. In the first step, the conventional static routing algorithm for binary n-cube (e-cube) is used. Messages are forwarded using the channels in decreasing order of dimensions. Thus the routing function is connected and is deadlock-free. In the second step, each physical channel c_i is split into k virtual channels, namely, $a_{i,1}, a_{i,2}, \cdots, a_{i,k-1}, b_i$. Let C_1 be the set of b channels. The new routing function defined in step 2 can be stated as follows: Route over any useful dimensions using any of the a channels. If all of them are busy, route over the highest useful dimension using the corresponding b channel (A useful dimension is one that sends a message closer to its destination).

Su and Shin have proposed an adaptive routing algorithm for mesh networks that also uses two virtual channels per physical channel [66]. They propose three protocols (3P) defining the relationship between messages and channel resources: request-then-hold, request-then-wait, and request-then-relinquish. Their fully adaptive routing scheme is based on the logical division of the set of virtual channels into two sets – waiting channels (fully adaptive channels) and non-waiting channels (deterministic channels). At each intermediate step, the packet is first routed in any direction that it can take to progress toward the destination using the fully adaptive channels. If no such channels are available, it waits for the deterministic channel. The approach is guaranteed deadlock-free as it always provides an escape path from the cycles in the form of deterministic channel routing, which allow deadlock-free routing.

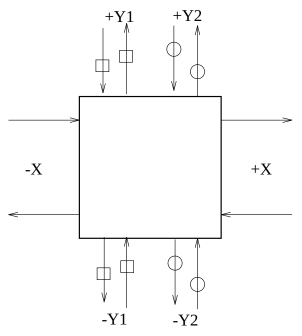
Boura and Das have proposed a wormhole routing technique called mesh_route algorithm similar to the 3P algorithm for n-dimensional meshes [12]. This algorithm also divides the set of virtual channels into waiting channels and non-waiting channels. A packet is first routed along any dimension using a free non-waiting channel. If no non-waiting channel is available, the packet is routed on the lowest positive dimension using the waiting channels. The mesh_route algorithm is efficient in using a greater proportion of virtual channels than the 3P routing scheme.

Upadhyay et al. [68] show that, in addition to adaptivity, the traffic distribution created by the algorithm also plays an important role in the performance of the routing algorithm. Uneven traffic distribution leads to early network saturation. They propose a new algorithm called PFNF (Positive-First, Negative-First) for two-dimensional meshes [68] that uses two virtual channels per physical channel. The physical interconnection network is logically divided into two virtual networks, VN_1 and VN_2, such that the two virtual channels associated with the same physical channel are in different virtual networks. Routing is done positive-first in one virtual layer and negative-first in the other. The PFNF algorithm is not only more adaptive than the 3P and mesh_route algorithms but also creates a balanced traffic distribution in the network.

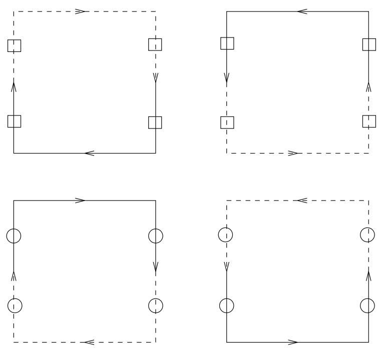
Algorithms requiring fewer than two VCs per PC:

Ni and McKinley propose fully adaptive routing algorithms that require two virtual channels per physical channel along only one of the dimensions, typically taken as the y dimension [55]. These *double-y* routing algorithms are implemented in double-y routers. A typical double-y router structure is shown in Figure 11(a) and the turns allowed by double-y routing are shown in Figure 11(b). Glass and Ni modify these algorithms by eliminating the unnecessary restrictions [32] and propose a new algorithm, called *maximally adaptive double-y* (mad-y), which makes better use of the virtual channels and improves adaptiveness. The turns allowed by the mad-y algorithm are shown in Figure 11(c). The mad-y algorithm is deadlock-free and provides better performance than the double-y routing algorithms [32].

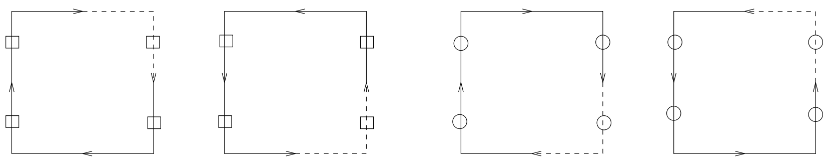
Schwiebert and Jayasimha propose an optimal fully adaptive routing algorithm, *opt-y* [62], that extends the mad-y algorithm. The mad-y algorithm requires an acyclic channel dependency graph, but the opt-y routing algorithm removes this restriction to allow cycles in the channel-dependency graph. The router structure for the opt-y algorithm is the same as in Figure 11(a), and the



(a) A router in a double-y network.



(b) Double-y routing.



(c) Mad-y routing.

Figure 11: A router in a double-y network and the associated routing algorithms. Dashed lines indicate prohibited turns.

turns describing the opt-y algorithm are shown in Figure 12. Restricted turns are those that allow routing only under certain constraints as defined below. The authors show that the opt-y algorithm is deadlock-free and optimal with respect to the number of virtual channels per router and number of routing restrictions on the virtual channels. The optimally fully adaptive routing algorithm can be generalized to n-dimensional meshes by using the following steps.

- Assign a channel to both directions of each dimension.
- Number the dimensions in some order and add a second virtual channel to both directions of all dimensions except the first.
- Allow a message to route along the second virtual channel at any time.
- For each dimension except the last, select one of the two directions as the chosen direction for that dimension. Prohibit a message from routing on the first virtual channel of any direction until it has completed routing in the chosen direction of all the lower dimensions.
- Allow a message to make a 0-degree turn between the two virtual channels of a direction only after the message has completed routing in the chosen direction of all lower dimensions.

The flexibility provided by adaptive routing improves performance for non-uniform workloads, but greater complexity is required to support the additional routing flexibility while assuring deadlock freedom. The increase in hardware complexity can significantly reduce router speed,

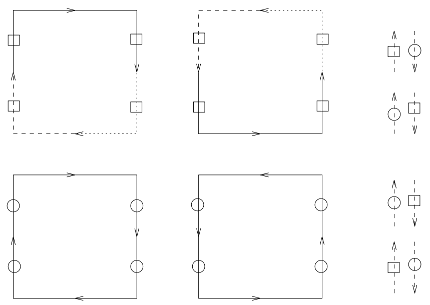


Figure 12: Optimally fully adaptive routing turns.
Dashed lines indicate restricted turns, dotted lines indicate prohibited turns.

thereby decreasing overall network performance. A comparison of various adaptive wormhole routing schemes reveals that adaptivity does not necessarily improve network performance in low-dimensional networks [10, 69], but does improve performance of high-dimensional networks such as hypercubes [25] and for non-uniform traffic patterns [51].

6.2 Partially Adaptive Algorithms

Several approaches based on limited adaptivity have been proposed to reduce the cost of adaptive routing. These *partially adaptive algorithms* allow routing freedom to be traded for router speed while assuring deadlock freedom. Partially adaptive algorithms use only a subset of the physical channels between source and destination.

A seminal work in partially adaptive wormhole routing is the turn model proposed by Glass and Ni [31]. The turn model defines a set of partially adaptive routing algorithms based on finding all the possible turns a message might make and then forbidding some minimum number of turns so as to avoid cyclic dependency. East-First, North-Last, Positive-First, Negative-First are some partially adaptive routing algorithms based on the turn model. The steps for designing routing algorithms using the turn model are:

1. Partition the channels into sets according to the directions in which they route messages. If each node has v channels in a topological direction, treat these channels as being in v distinct virtual directions and divide them into v distinct sets accordingly. Put any wraparound channels in a separate set to be incorporated during step 5.

2. Identify the possible turns from one virtual direction to another, omitting 0-degree and 180-degree turns. (A 0-degree turn is possible only when there are multiple channels in a topological direction. It represents a transition from one set of channels to another, where the two sets are in the same topological directions but different virtual directions.)

3. Identify the cycles that the turns can form. Generally, identifying the simplest cycles in each plane of the topology is adequate.

4. Prohibit a minimum number of turns so that at least one turn is prohibited in each cycle. The turns must be chosen carefully in order to break every possible cycle, including very complex cycles. A useful approach is to first break the cycles in each plane and then check whether doing so allows more complex cycles.

5. Incorporate as many turns as possible involving the set of wraparound channels, without reintroducing cycles. At least one turn involving each wraparound channel can always be incorporated.

6. Incorporate as many 0-degree and 180-degree turns as possible, without reintroducing cycles.

Routing algorithms that route packets along the set of channels identified in Step 1 and use only the turns from one set to another allowed by Steps 4, 5, and 6 are deadlock free, livelock free, and highly adaptive.

We now explain the basic concept of the turn model for 2D meshes. Call the directions -x, +x, -y, and +y west, east, south, and north, respectively. Eight possible turns can be made, shown as the abstract cycles in Figure 13(a). The xy routing prevents deadlock by preventing four turns, as shown in Figure 13(b). However, deadlock can be avoided by prohibiting two turns, one from each abstract cycle. This relaxation allows partial adaptiveness and is the crux of turn model. Prohibiting any two turns will not prevent deadlock. Of the 16 different ways to prohibit two turns, 12 prevent deadlock and three are unique if symmetry is taken into account. These three types corresponds to three routing algorithms - west-first, north-last, and negative-first. The turns allowed in these algorithms are illustrated in Figures 14 (a), (b), and (c), and examples of the west-first and north-last algorithms are shown in Figures 15 (a) and (b). These algorithms can also be extended for deadlock free routing in n-dimensional meshes [31].

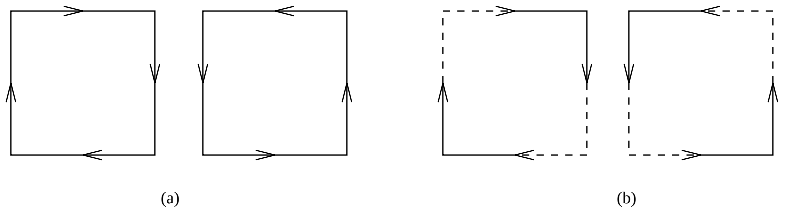
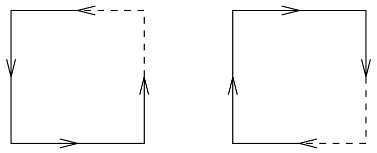
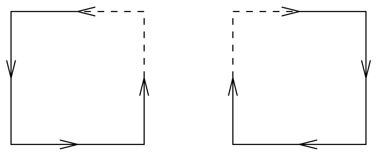


Figure 13: Abstract cycles, and cycles in xy routing algorithm.

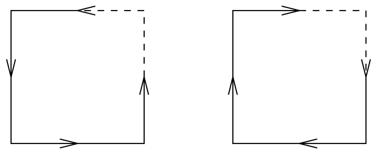
Another efficient partially adaptive routing algorithm, the planar adaptive algorithm, was proposed by Chien and Kim for n-dimensional meshes and torus [15]. Instead of providing adaptivity in all dimensions, it restricts adaptivity to two dimensions at a time. As the message progresses toward its destination, it passes through a series of adaptive two-dimensional planes; eventually the packet completes routing in all dimensions and is delivered to the destination. The adaptive routing planes in three and four dimensions are illustrated in Figure 16. Within each adaptive plane, messages may use any channel leading toward their destination. Because of the restricted routing freedom, the possibility of interdimensional resource cycles is reduced, so that fewer resources are



(a) The six turns allowed (solid lines) by the west-first algorithm.



(b) The six turns allowed (solid lines) by the north-last algorithm.



(c) The six turns allowed (solid lines) by the negative-first algorithm.

Figure 14: Turns allowed in the three routing algorithms.

needed to avoid deadlock. It is shown in [15] that only three virtual channels per physical channel are required to avoid deadlock using planar adaptive routing in a k-ary n-cube with no wraparound paths.

Planar adaptive routing has two phases: high-level routing corresponds to routing between the adaptive planes and the low-level routing corresponds to routing within the adaptive planes. Let A_i represent the adaptive plane between dimension d_i and d_i+1. In the high-level routing, the message is routed successively in adaptive planes. Routing in adaptive plane A_i reduces the distance in d_i to zero. After routing in all of the adaptive planes, the message would reach its destination. In the last dimension there cannot be any adaptivity left for a minimal router, so the packet is routed deterministically to its destination. In low level routing, the scheme is adaptive, as multiple paths can be chosen within each adaptive plane. In each adaptive plane the packet completes its routing in at least one dimension: in plane A_i, the d_{i+1} distance is reduced to zero first, then the routing continues in d_i exclusively until the d_i distance is reduced to zero.

Boura and Das have proposed another model for partially adaptive algorithms for n-dimensional meshes, the direction restriction model [13], that is based on dividing a system into two unidirectional networks, e.g., positive and negative. The message is transmitted in two phases. In the first phase, the message is routed adaptively to an intermediate node using one unidirectional network; in the second phase, it is routed adaptively to its destination using the other unidirectional network. This model thus defines a class of partially adaptive routing algorithms for n-dimensional meshes. There are 2^{n-1} different pairs of complementary networks in an n-dimensional mesh. Depending upon selection of these unidirectional networks, Boura and Das show that 2^n different algorithms are possible for an n-dimension mesh without requiring any virtual channels.

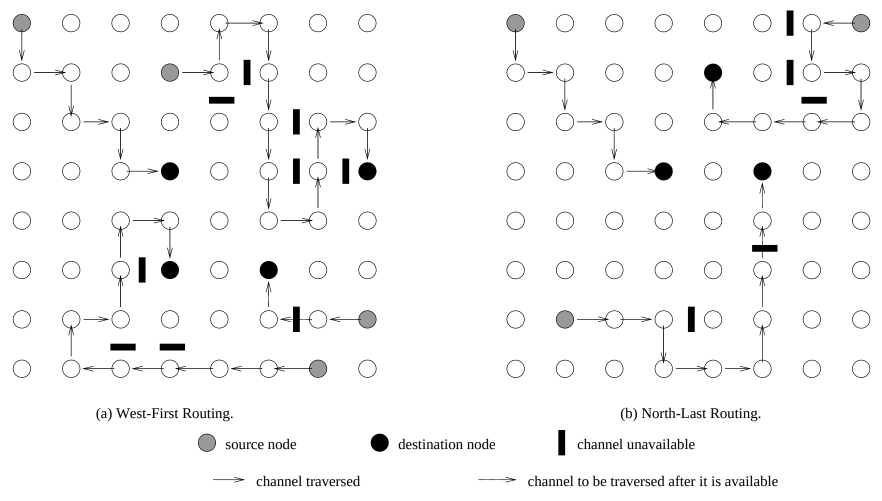


Figure 15: Examples of west-first and north-last algorithms [31].

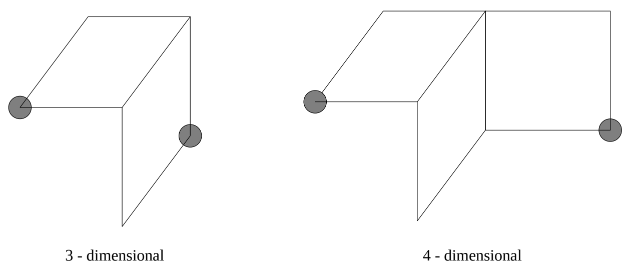


Figure 16: Adaptive routing planes.

6.3 Deadlock Recovery in Fully Adaptive Algorithms

Almost all the deadlock-prevention mechanisms discussed so far use additional resources or implementations to prevent the formation of cycles and thereby avoid deadlock. However, studies have shown that potential deadlock situations are rare in multicomputer systems [6, 44, 58, 59] and it may not be cost-effective to dedicate resources to handle rare events. Deadlock recovery is an alternative to deadlock avoidance or prevention. Using this concept, the messages can be routed fully adaptively, allowing the formation of cycles. A detection mechanism identifies potential deadlock configurations; once deadlock is detected, a recovery scheme breaks the deadlocked cycle. In this section, we review the deadlock recovery mechanisms proposed so far in the literature.

Reeves et al. have proposed an adaptive routing scheme for hypercube systems that uses an abort-and-retry mechanism for recovering from deadlock and reducing traffic congestion [59]. They model a protocol that aborts a message whenever it is blocked beyond a threshold number of cycles. The message is then reintroduced into the network after a random number of cycles. It is shown that the abort-and retry mechanism improves performance under a broad range of traffic conditions.

Kim, Liu, and Chien [44] report an adaptive routing framework called *compressionless routing* (CR) that supports adaptive and fault-tolerant routing for a wide variety of network topologies without using any virtual channels. A feature of wormhole routing that provides feedback in the form of flow control is exploited in the proposed mechanism. The basic idea of CR is to use fine-grain flow control and backpressure of wormhole routing to communicate routing status and error conditions to the network interfaces. The network interface uses the information to provide deadlock recovery and end-to-end fault tolerance. Thus if the message is long enough, the sender can determine if the message header has reached its destination; if the message is not long enough, the sender pads it to ensure that the header reaches the destination before the last flit is injected by the source. Figure 17 illustrates message routing in a CR network. While routing, if a message is blocked at a node for an interval larger than a preset time-out, it is aborted by the source and resent later.

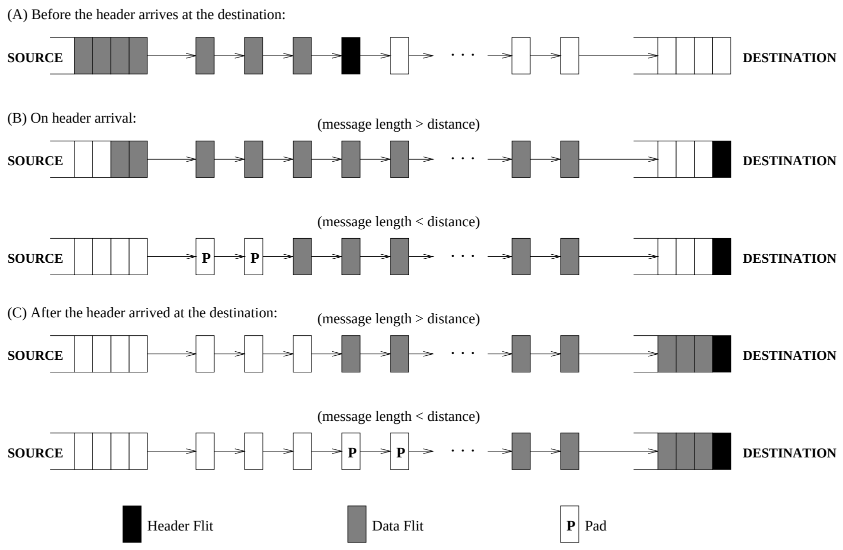

(A) Before the header arrives at the destination:
SOURCE
DESTINATION
(B) On header arrival:
(message length > distance)
(message length < distance)
P
(C) After the header arrived at the destination:
Header Flit
Data Flit
Pad

Figure 17: Message routing and padding in CR networks.

The performance of CR depends on the time-out interval. Simulation results also indicate that the CR torus networks (with a single channel) give comparable or better performance than dimension-ordered networks (with two virtual channels) under uniform traffic. CR with two virtual channels significantly outperforms dimension-ordered networks.

Anjan and Pinkston propose a deadlock-recovery strategy called *disha* that provides a framework for supporting deadlock-free fully adaptive wormhole routing [6]. The routing is done without any virtual channels or turn restrictions. However, virtual channels can be used to increase throughput and reduce deadlock frequency. Recovery from deadlock is achieved through a single additional flit buffer at each node. This “deadlock buffer” is a special input buffer central to each router used only in potential deadlock situations; it is a shared resource that can be accessed from all neighboring nodes. The deadlock buffers form a deadlock-free lane during recovery. When a deadlock cycle is formed, one of the messages in the cycle is switched to the deadlock-free lane and routed minimally along the path until it reaches its destination and is eventually consumed. Thus the cycle breaks and all the other messages proceed. Two versions of *disha* have been proposed. The first allows only sequential deadlock recovery [6]: access to deadlock buffers is controlled by a circulating token. The second version allows concurrent deadlock recovery [7]: it requires no token and is based on an extension of the theory of deadlock avoidance described in Section 4.2. Deadlocks can be detected by a time-dependent selection function or by a counter associated with each input channel [26]. A time-out interval determines the maximum time a message can be blocked at a router. After this interval, the message is considered "deadlocked." The selection of a proper time-out interval is important for optimum performance. The counter counts the number of cycles since the header arrived. When the header waits for longer than some threshold, it is assumed to be deadlocked. Simulation results indicate that *disha* provides better performance than the deadlock-avoidance schemes [6].

7 Fault-Tolerant Wormhole Routing

With the trend toward large-scale parallel systems, fault tolerance becomes important in routing algorithms. Ideally the algorithm should be able to route a message to its destination as long as the source and the destination nodes are connected. However, this is not always possible due to the routing constraints for avoiding deadlocks. This section summarizes some of the fault-tolerance algorithms presented in the literature.

The planar adaptive routing discussed in Section 6.2 can be modified to support fault tolerance [15]. As the algorithm does not allow backtracking, a message may get "trapped" in a concave faulty region. This problem is solved by marking some operational nodes as faulty in order to convert the concave faulty regions to convex. Then the planar adaptive routing can be used to route messages to all the nodes that are connected. The basic idea is to use the adaptivity to circumvent any faulty channels. Consider a plane A_i in dimensions d_i and d_{i+1}. The high-level routing remains the same as described in Section 6.2. The low-level steps are:

1. If not blocked by a fault, route as in the fault-free case.
2. If blocked by a fault in dimension d_{i+1}, route in d_i.
3. If blocked by a fault in d_i, route in d_{i+1}.
4. If blocked by a fault in d_i and the d_{i+1} distance has already been reduced to zero, then misroute. If we were routing in d_{i+1}, continue to route in the same direction. If we were routing in d_i, pick an arbitrary d_{i+1} direction and begin misrouting. At the first opportunity, route in d_i toward the destination. Continue to route in d_i only until it is possible to correct d_{i+1}. At that point, route in d_{i+1} to distance zero in this dimension, then revert to step 1.
5. We cannot be blocked in d_{i+1} and have reduced the d_i distance to zero. If this were the case, we would have proceeded to the next adaptive plane.

The CR scheme described in Section 6.2 above can be also extended to handle faults in wormhole routed networks [44]. The basic idea in fault-tolerant CR is to use the retransmission mechanism to tolerate transient faults and to use unrestricted routing flexibility to circumvent permanent network faults. When a transient fault is detected, the detecting router sends kill signals in both forward and backward directions along the message path. The source node retransmits the same message after receiving the kill signal. When a message encounters a permanent fault, it circumvents the faults using alternative paths. To ensure reliable message transfer, each message holds its path until the last data flit reaches the destination.

Glass and Ni have proposed an extension of the negative-first algorithm to make it fault-tolerant [34]. The negative-first algorithm provides full adaptivity to the message at all times except when it is routing in the negative edge of the mesh or in the last dimension. The fault-tolerant extension of this algorithm is targeted at removing these few cases of non-adaptiveness. Their approach is based on the following restrictions: (a) avoid routing a packet to the negative edge of the mesh as long as possible; (b) route a message around a faulty node on a negative edge of the mesh; (c) route a message farther negative if the destination node is in the negative direction of the source; (d) avoid routing a message in the positive direction from the destination as long as possible. With these restrictions, if a node ever finds it impossible to route a message further, it discards the packet and possibly returns an acknowledgment of the error to the sender. An example of a few paths allowed by this fault-tolerant routing algorithm is illustrated in Figure 18. The main advantage of this scheme is that it needs no virtual channels for fault tolerance: it is a simple modification of the negative-first algorithm that enables it to tolerate $n-1$ faults in an n-dimensional mesh.

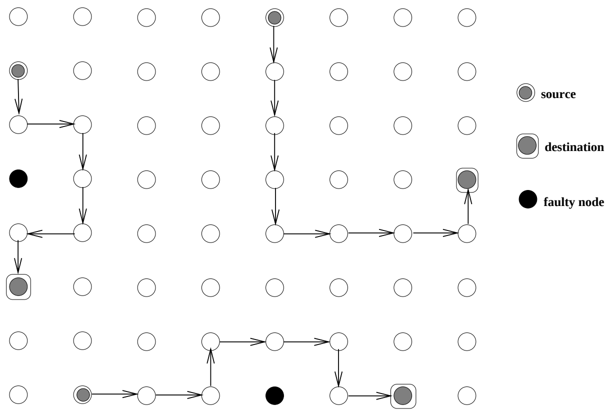


Figure 18: Sample paths allowed by the fault-tolerant routing in a two-dimensional mesh.

Hadas and Brant propose an origin-based fault-tolerant routing scheme for mesh-connected systems [48]. In origin-based routing, one of the nodes is considered the origin and the other nodes are represented with coordinates with respect to this origin. The coordinates for the source node s and the destination node t are (x_s, y_s) and (x_t, y_t), respectively. The channels of the network are partitioned into an IN subnetwork and OUT subnetwork: the IN subnetwork contains all channels directed toward the origin and the OUT subnetwork contains all channels directed away from the origin. The following definitions are helpful for explaining the origin-based routing [48]:

Definition 10: The *outbox* for a destination node t with coordinates (x_t, y_t), $(x_t, y_t \geq 0)$ is the set of all nodes n with coordinates (x_n, y_n) such that $0 \leq x_n \leq x_t$ and $0 \leq y_n \leq y_t$.
Definition 11: A node v is safe with respect to destination node t if

1. Node v is in the outbox for t.

2. For any pattern of faults in which v and t are non-faulty, there exists a fault-free path in the OUT subnetwork from v to t.

Definition 12: The diagonal band for destination node t is the set of all nodes v satisfying the following properties:

1. Node v is in the outbox for t.

2. If (x_t, y_t) and (x_v, y_v) are the coordinates of t and v, respectively, then $x_t - x_v = y_t - y_v + e$, where $e \in \{-1, 0, 1\}$.

The origin-based fault-tolerant routing scheme has three phases. In the first phase, a message is routed adaptively using the IN subnetwork while the header flit is not in the outbox for its destination node t. When the header flit enters the outbox, the second phase starts. While the header flit is not at a safe node, the distance to the nearest safe node in each direction is computed and compared with the distance to the nearest fault in that direction. If the safe node is closer than the fault, the message is routed to the safe node; otherwise, message routing continues in the IN subnetwork. The message enters the third phase of routing when it arrives at a safe node v. If v has a safe non-faulty neighbor, the header flit is forwarded to that node. Otherwise, since v must be on the edge of a faulty region, v advances along the edge of the faulty region towards t and turns towards the diagonal band when it reaches the corner of the faulty square. Thus, the message returns to a safe node as the nodes on the diagonal band for t are safe with respect to t. Origin-based fault-tolerant routing is deadlock-free as well as livelock-free [48]. Figure 19 shows an example of origin-based fault-tolerant routing algorithm.

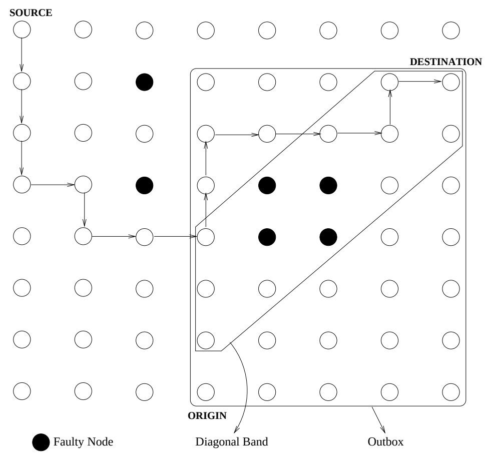


Figure 19: Sample origin-based fault-tolerant routing in a mesh.

Boppana and Chalasani have enhanced the e-cube algorithm for fault tolerant routing in mesh networks [9]. Their algorithm the f-cube algorithm, is based on the concept of fault rings and fault chains, which are formed using fault-free nodes and links around each faulty region. The fault-free nodes form either rings (f-rings) or chains (f-chains) in a faulty network, as shown in Figure 20. Messages are routed around the f-rings and f-chains following dimension-ordered routing. They have shown that if fault rings do not overlap, i.e., if the links in the fault rings are pair-wise disjoint, two virtual channels per physical channels are sufficient to make e-cube algorithm tolerant of any number of faulty blocks. For cases where fault rings overlap, three to four virtual channels are required. Boppana and Chalasani have further extended their algorithms for adaptive fault-tolerant routing using four additional virtual channels [9].

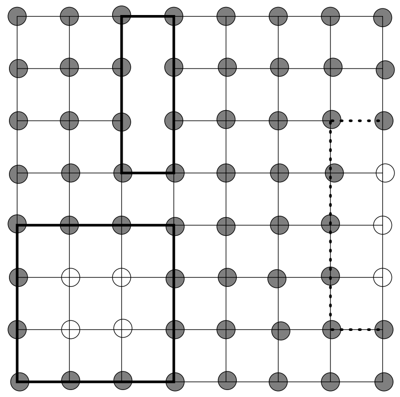


Figure 20: Examples of f-cube and f-rings in a mesh. Unshaded nodes are faulty nodes, solid bold lines are f-rings, and dotted bold lines represent a f-chain.

Gaughan and Yalamanchili have proposed a family of fault-tolerant routing protocols for direct multiprocessor networks [30]. Their scheme is based on pipelined circuit switching (PCS), which is a variant of the wormhole routing mechanism. In PCS, data flits do not immediately follow the header flits into the network. After the header reaches the destination, an acknowledgment is sent to the source and then the data flits are transmitted as in wormhole routing. By relaxing some of the routing constraints imposed by wormhole routing, PCS makes possible routing behavior that cannot otherwise be realized. For example, rather than blocking on busy channels, the header may "backtrack," release previously reserved channels on the path and attempt an alternative path to the destination. Thus, using PCS, the authors have developed fault-tolerant routing algorithms – misrouting backtracking with m misroutes (MB-m). Results indicate that this methodology provides performance approaching that of wormhole routing unless messages are very short, while realizing resilence to static faults that is difficult to achieve with wormhole routing.

A new flow-control mechanism called *scouting* has been proposed by Duato et al. for improving fault tolerance [29]. In Scouting, a probe is sent to reserve the path. The probe is allowed to backtrack, as in PCS. Instead of waiting for an acknowledgment however, data flits follow the header at a distance that can be dynamically adjusted from 0 (as in wormhole routing) in fault-free regions of the network to a value equal to the diameter of the network when the message is crossing a region with faults. A detailed performance analysis with respect to the dynamically adjustable distance is reported in [21].

Duato has also developed necessary and sufficient conditions for deadlock avoidance in fault-tolerant routing algorithms [28]. by analyzing the channel redundancy as well as the network redundancy and defining redundancy levels. Network redundancy level is the maximum number of channels that can fail in the worst case such that the resulting routing algorithm remains connected and deadlock-free. Using these analyses, Duato proposed necessary and sufficient conditions for a routing algorithm to achieve a given redundancy level [28]. The same work also studies the effect of faults in physical channels on virtual channels and node failures.

Varavithya et al. study combining virtual cut-through with wormhole routing to achieve fault tolerance [71] by extending the PFNF algorithm [68] to handle faults without using any additional virtual channels. Upon encountering a faulty node, a message uses alternative paths provided by the routing function unless it is in the last dimension. In the final dimension, when a message is blocked by faulty nodes, it is completely stored at one of the adjacent nodes from which it is subsequently retransmitted. It is shown that under normal circumstances, this approach results in much better performance, with the added advantage that it does not need any additional hardware specifically for fault tolerance. A similar scheme was also proposed independently by Suh et al. [67].

The Reliable Router (RR) reported by Dally et al. is an excellent router design that addresses a practical implementation of fault-tolerant routing algorithms [20]. It is designed to run at 100 MHz and reaches a useful link bandwidth of 3.2 Gbit/sec. RR uses adaptive routing coupled with link-level retransmission and a unique-token protocol to increase both performance and reliability. The RR can tolerate a single node or link failure anywhere in the network without interruption of service.

8 Wormhole Routing in Commercial Systems

In this section, we describe the implementation of wormhole routing schemes in commercial parallel computers. The algorithms, hardware requirements, flow control, and other implementation details of wormhole routing in nCUBE-2, CM-5, Cray T3D, Intel Paragon and IBM SP1/SP2 are reported next.

Wormhole Routing in nCUBE-2

The nCUBE-2 parallel computer series supports a hypercube configuration of up to 8192 nodes in 13 dimensions [53]. Interprocessor communication is handled through a network communication unit (NCU), which includes 14 DMA ports that support the hypercube interconnection scheme. The DMA ports include 13 bidirectional interprocessor communication ports (26 unidirectional channels) for communicating with processors that are part of the local hypercube and one bidirectional system interconnect I/O port, consisting of two unidirectional channels, for communicating with remote or other systems.

The NCU architecture has three layers: the interconnect layer, the routing layer, and the message layer. The interconnect layer provides the hardware to establish physical communication links: 28 independent serial DMA channels provide 14 full-duplex I/O ports, allowing systems to be designed with up to a 13-dimensional hypercube. The variable communication speed of I/O ports allows matching of the port speed to the signal propagation time of the interconnect. The routing layer provides the arbitration and switching logic for creating, maintaining, and removing communication paths between processors in the network. The wormhole switching technique routes messages between the nodes. The routing layer assumes that each processor has a unique processor ID and that the IDs of two processors connected to each other through port k vary only in the kth bit. The software establishes a communication path by sending an address packet over a channel. The hardware passes the address from node to node. Each processor compares the destination node address with its own ID and sends the address packet out through the port number corresponding to the bit position of the first difference, starting at bit $n+1$, where n is the number of port on which the message was received. Thus, while using the hardware default routing, messages are always sent out on a port with a number higher than the port on which the message was received. An established routing path blocks all messages that try to use the same channels until the hardware clears the path with an end-of-transmission (EOT) packet. The nCUBE uses the dimension-order e-cube routing algorithm to route messages from one node to another; as has been shown above, this algorithm enables deadlock-free message transfers.

The message layer provides for reliable and efficient point-to-point data transfer between the nodes. The NCU can buffer up to two packets on each incoming channel and requests another packet as soon as it has space for it, until an EOT packet arrives.

Wormhole Routing in CM-5

The basic topology of the CM-5 data network is a 4-ary fat tree in which each internal node is made up of several router chips [46]. The fat-tree topology provides adaptable bandwidth between the nodes. Each router chip is connected to four child chips and either two or four parent chips; each connection provides a link to another chip with a raw bandwidth of 20 megabytes/second in each direction. Flow control is provided on every link.

Message routing in the CM-5 data network uses an adaptive wormhole routing mechanism. The network design provides many alternative paths from a source to the destination. As a message goes up the tree, it may have several choices for taking a parent connection. A pseudorandom selection process selects a link that is not occupied by other messages. After the message attains the height of the least common ancestor of the source and destination processors, it takes the single available path of links from that chip to its destination. The random selection at each level balances network load and avoids congestion. On average, each processor can provide data into and out of the network at a rate in excess of four megabytes/second; higher bandwidths are achievable for localized communication patterns. Network latency ranges between three and seven microseconds, depending on the size of the machine.

The data router chip has an eight-bit-wide bidirectional link (four bits in each direction) to each of its four child chips lower in the fat tree, and four 8-bit-wide bidirectional links to its parent chips higher in the fat tree. The chip can be viewed as a crossbar connecting the eight input ports to the

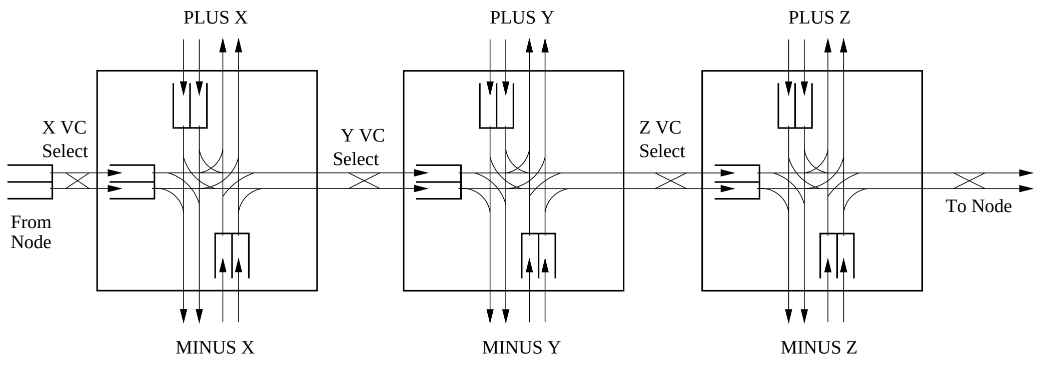


Figure 21: Architecture of the Cray T3D router.

eight output ports. When a message is blocked from its desired output port, it is buffered. Flow control information is passed in the reverse direction of message traffic to prevent buffer overflow. When multiple messages compete for the same output port, the arbitration is fair and prevents any link from being starved.

The data network in CM-5 has a contract with the processors that guarantees delivery of all messages. The contract promises to accept and deliver all messages injected into the network by the processors as long as the processors promise eventually to eject all messages from the network when they are delivered to them. The data network is acyclic from inputs to outputs, which precludes deadlock if the contract is obeyed. Each processor has two outgoing and two incoming FIFOs in its interface to the data network: a left port and a right port. The topology of the network is such that all links reachable from the left port are unreachable from the right port and vice versa. Thus, the data network is really two independent, interleaved networks. Requests can be sent on the left side of the network, and responses returned on the right side. If a processor cannot send a response on the right side and its constant-size buffer is full, it stops receiving on the left side. Since any processor requesting data has a place to put it, however, the processors can satisfy the contract on the right side and the responses will eventually clear out. A processor can eventually accept every request that arrives on the left side, and thus satisfy the contract on the left side. Consequently, deadlock cannot occur.

Wormhole Routing in Cray T3D

Cray T3D is a distributed shared-memory system in which the nodes are interconnected through a bidirectional, 3D torus network [41]. The network links are 24 bits wide (16 data, 8 control) and are clocked at 150 MHz. The latency per hop in the absence of contention is two clock cycles. The network ports have the same capacity as the internode links, and are shared by two PEs at each node. The T3D network uses a deterministic, dimension-order, e-cube wormhole routing scheme. The router is physically partitioned into three ECL gate arrays, one for each dimension (see Figure 21). Packets route in the X direction first, the Y direction next, then the Z direction. The flits are usually of 8 16-bit phits or less. Packet sizes range from 3 to 26 phits, and carry header information plus zero, one, or four 64-bit data words. Virtual channel buffers are one flit deep.

Each physical channel is associated with four virtual channels, two of which are used for request traffic and the other two for response traffic. For clarity, only two virtual channels are shown in Figure 21. Deadlocks are avoided by a combination of the following three factors: (a) separate virtual networks for requests and responses remove cyclic dependencies between these traffic classes, (b) dimension-order routing removes cyclic dependencies involving multiple dimensions, and (c) two virtual channels per traffic class are used to remove cyclic channel dependencies involving the wraparound connection within a given dimension. Routing in T3D employs end-to-end routing tables. Each node contains a table, stored in dedicated hardware, that provides a routing tag for every destination node in the machine specifying a direction, offset, and virtual channel for each of the three dimensions. The routing tables are loaded by software but are used directly by the hardware. They also allow alternate routes to be taken (i.e., the long way around a torus) to avoid faulty nodes and links. The routing tables let us individually specify the virtual channels used for every source-destination pair in the machine. Thus, two packets traveling along exactly the same segment on a ring (as identified by the source and destination nodes on that ring) could use different virtual channels, depending upon their original source and final destination.

The primary use of virtual channels in the T3D is to prevent deadlock. Since request and response packets are routed on separate virtual networks, the dimension-order routing breaks cycles between dimensions. Thus, we need to be concerned only about avoiding deadlock within a single ring of the torus. This can be done using two virtual channels, VC0 and VC1, and through logical *datelines*, which are imaginary lines cutting a ring that can only be crossed by traffic on the appropriate virtual channel. A packet stays on the same virtual channel while routing on a given ring. No traffic ever switches between virtual channels. Therefore, if datelines are enforced for both VC0 and VC1 in both positive and negative directions for each ring, there can be no cyclic dependencies among VC buffers, and thus no deadlock. The datelines are logically placed at carefully selected nodes so that the traffic through the two virtual networks is more or less balanced. Balanced traffic enforces load balancing among the two virtual channels, which results in improved performance over the unbalanced usage proposed in several adaptive routing schemes.

Wormhole Routing in Intel Paragon

Intel Paragon has a two-dimensional mesh topology and can be configured as a 16x4N mesh, where N is the number of cabinets. Each node in Paragon is connected to a mesh router chip (MRC) as shown in Figure 22. There are 10 unidirectional ports per MRC. A pair of ports are in each of the four directions – north, south, east, and west – for communicating with neighboring nodes; the fifth pair is used for communicating with the node associated with the router. The ports enable 16-bit parallel transfers plus parity, providing a total bandwidth of more than 200 MBytes/second per port. Data flow through the MRC takes about 10ns for a straight path and a little longer for a turn.

Messages in Paragon are routed deterministically using the wormhole switching technique. Messages are packetized prior to hardware transfer and the XY routing algorithm is used to send the packets from a source node to destination. Signed horizontal and vertical displacements are used for

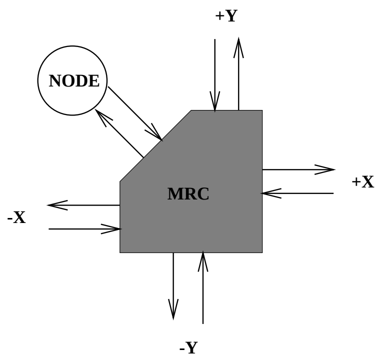


Figure 22: Mesh router chip of Intel Paragon.

routing the packets of a message. The horizontal and vertical displacements are computed at the source node. The packet is routed first in the horizontal dimension until the horizontal displacement is zero, and then in the vertical dimension until the vertical displacement becomes zero. The message is then absorbed at the destination node. All ports in the MRC can be active simultaneously, and thus up to five message packets can be routed simultaneously without contention.

The message routing scheme in Paragon uses the deterministic XY algorithm, so it is deadlock-free, as proved earlier.

Wormhole Routing in IBM SP1/SP2

The nodes of IBM SP1 and SP2 are interconnected by a multistage interconnection network (MIN) [1, 3], whose links contain two channels carrying packets in opposite directions between two network devices. The MIN comprises of 8x8 Vulcan switch chips [65] (Figure 23). The switch consists of eight receiver and eight transmitter modules, an unbuffered 8x8 crossbar, and a 1 KB large central queue. Each input and output port consists of eight data lines and two control lines, so that each port can process one flit (one byte) per cycle. A crossbar switch is implemented between the input and output ports for transferring packets that encounter no contention for their desired output port. If there is contention, the flits of a packet are stored in a central queue that has a 1 KB dynamically allocated shared buffer. Larger portions of the shared buffer are allocated to the busier input ports. This dynamic behavior improves the network performance. To match the maximum possible bandwidth from the input ports, it is necessary to write eight flits per cycle into the central queue. Thus, each input port queues a chunk of eight flits at the deserializer before requesting service from the central queue, and writes the entire chunk in one cycle when the request is granted. The serializer at the output ports converts an eight-flit central queue chunk into the flit-wide data stream sent from the output port. As long as the central queue is not full, each input port can continue to receive flits at full bandwidth.

Processor nodes of the IBM SP1/SP2 communicate by sending and receiving message packets. Packets are of variable length and up to 255 bytes in size. The method of packet transfer is similar to wormhole routing. The only difference is that when a packet is blocked the packet bytes are not buffered in place but are temporarily transferred to the central queue until the blocked output port

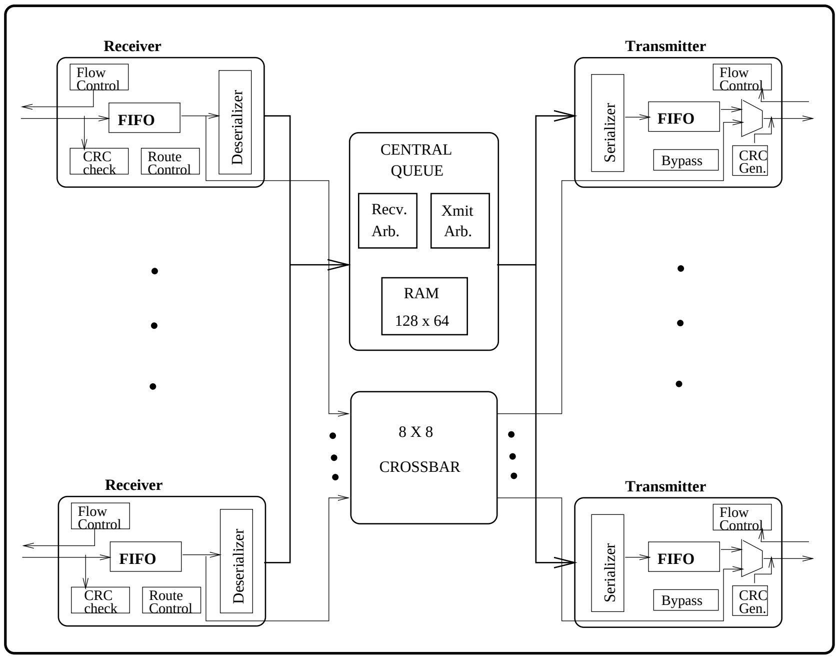


Figure 23: Architecture of the Vulcan switch in IBM SP1/SP2.

is cleared up. The first byte of each packet indicates the packet length, followed by a number of routing bytes, followed by data. The complete routing decision is made at the source node (source-initiated routing). At each switch, the first byte of the packet is examined and the output port is determined. The corresponding switch strips off the portion of routing information pertaining to itself before sending the packet forward. In the absence of output contention, packet bytes pass through a switch in five clock cycles. The switch operates at 40 MHz, resulting in a peak bandwidth of 40 MB/s per port. The corresponding input and output ports of the switches are paired to form a full duplex bidirectional channel. Thus the 4x4 bidirectional switch element can forward a packet to any of the eight output ports, including the output ports on the same side of the input ports. This implementation facilitates *turnaround routing*.

The routing algorithm used in SP1/SP2 selects a single shortest path between each pair of processor nodes and is deterministic in nature. A modified *breadth-first* search algorithm is implemented to build a breadth-first spanning tree rooted at each source node, and then the spanning tree paths are followed to find the shortest path from the source nodes to the rest of the processor nodes. A static load-balancing technique is used to ensure that links are included in the selected routes in a balanced manner. The routes are stored in a routing table in each processor's memory, which lets routing be done in a topology-independent fashion. This approach differentiates the SP1/SP2 routing algorithm from the other deterministic routing schemes, namely the XY and e-cube: they are topology dependent, XY for mesh and e-cube for hypercube.

9 Conclusions and Open Issues

Wormhole routing has emerged as the most widely used switching technique in massively parallel computer systems. This paper has given a comprehensive survey of various algorithms and techniques proposed to enhance its performance. A number of open issues, however, remain. Some of these issues are itemized as follows.

- Performance of wormhole-routed networks is not necessarily improved by increasing the adaptivity of the routing algorithm. Adaptivity is incorporated in the routing algorithm at the expense of additional hardware. Other factors, such as balanced traffic distribution, have a significant effect on network performance and may well be implemented at a lower cost. It is thus necessary to investigate cost-effective methods to improve network performance.

- The performance of wormhole routed algorithms has been evaluated through analysis or simulation with generalized workloads. The behavior of such algorithms should be studied with real traces obtained from parallel computers.

- The overhead associated with the decision making process at the router is generally ignored but might have significant impact on the overall network performance. Attempts in this direction have already been made by Chien [14] and others. However, all microoperations needs to be evaluated in detail for complete performance evaluation.

- Fault-tolerant routing algorithms have been evaluated assuming random faults at the nodes. It might be interesting to incorporate fault-injection mechanisms and evaluate the performance of the network under faulty links, nodes, routers, and interfaces.

- Collective operations that involve several processing nodes occur frequently in multicomputer systems, for instance in multicast, broadcast, gather, scatter, and barrier synchronization. A good survey of several algorithms for collective communication is given in [52]. Efficient hardware communication support can be implemented for these operations to reduce their latency further. Additional work in this area is needed.

- Research efforts on wormhole routing in switch-based networks should continue because of the renewed interests in such architectures [54].

- Finally, application-specific routing algorithms are needed for specialized systems, probably related to the routing of real-time traffic or multiple classes of traffic [60].

This report has concentrated on hardware routing algorithms. However, some functionality can be implemented in software to derive hybrid routing schemes for cost-effective high-performance networks.

Acknowledgements

The author is grateful to Jatin Upadhyay and Vara Varavithya for their help in preparing this paper, and the comments and suggestions by Professor Lionel Ni have improved its quality. An extensive and detailed review of an earlier draft of the paper by Professor Jose Duato is greatly appreciated. The author is thankful to Katrina Avery for her editorial assistance, which improved the readability of the paper. The author also thanks the referees for their valuable comments.

References

[1] B. Abali and C. Aykanat, "Routing Algorithms for IBM SP1," *Parallel Computer Routing and Communications Workshop,* pp. 161-175, May 1994.

[2] V. S. Adve and M. K. Vernon, "Performance Analysis of Mesh Intercomnnection Networks with Deterministic Routing," *IEEE Trans. on Parallel and Distributed Systems*, pp. 225-246, Mar. 1994.

[3] T. Agerwala, J. L. Martin, J. H. Mirza, D. C. Sadler, D. M. Dias, and M. Snir, "SP2 System Architecture," *IBM Systems Journal,* vol. 34, No. 2, pp. 152-184, 1995.

[4] A. Agrawal, "Limits on Interconnection Network Performance," *IEEE Trans. on Parallel and Distributed Systems*, vol. 2, no. 4, pp. 398-412, Oct. 1991.

[5] S. B. Akers and B. Krishnamurthy, "A Group-Theoretic Model for Symmetric Interconnection Networks," *IEEE Trans. on Computers,* pp. 555-566, April 1989.

[6] K. V. Anjan and T. M. Pinkston, "An efficient, fully adaptive deadlock recovery scheme: DISHA," *Int. Symp. on Computer Architecture*, pp. 201-210, June 1995.

[7] K. V. Anjan, T. M. Pinkston, and J. Duato, "Generalized Theory for Deadlock-Free Adaptive Routing and its Application to Disha Concurrent," *International Parallel Processing Symposium*, April, 1996.

[8] K. Aoyama and A. A. Chien , "The Cost of Adaptivity and Virtual Lanes in Wormhole Router,", To appear in the *Journal of VLSI Design.*

[9] R. V. Boppana and S. Chalasani, "Fault-tolerant wormhole routing for mesh networks," *IEEE Trans. on computers*, pp. 848-864, July, 1995.

[10] R. V. Boppana and S. Chalasani, "A Comparison of Adaptive Wormhole Routing Algorithms," *Intl. Symposium on Computer Architecture*, pp. 351-360, May 1993.

[11] R. V. Boppana and S. Chalasani, "A Framework for Designing Deadlock-Free Wormhole Routing Algorithms," *IEEE Trans. on Parallel and Distributed Systems,* pp. 169-183, Feb. 1996.

[12] Y. M. Boura and C. R. Das, "Efficient fully adaptive wormhole routing in n-dimensional meshes," *Intl. Conference on Distributed Computing Systems*, pp. 589-596, 1994.

[13] Y. M. Boura and C. R. Das, "A class of partially adaptive routing algorithms for n-dimensional meshes," *Proc. of the 23rd Intl. Conference on Parallel Processing*, vol. 3, pp. 175-182, August, 1993.

[14] A. A. Chien, "A Cost and Speed Model for k-ary n-cube Wormhole Routers," *Proc. of Hot Interconnects*, 1993.

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

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

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

[18] W. J. Dally and H. Aoki, "Deadlock-free adaptive routing in multicomputer networks using virtual channels," *IEEE Trans. on Parallel and Distributed Systems*, vol. 4, no. 4, pp. 466-475, April 1993.

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

[20] W. J. Dally, L. Dennison, D. Harris, K. Kan and T. Xanthopoulus, "The Reliable Router: A Reliable and High-Performance Communication Substrate for Parallel Computers," *Workshop on Parallel Computer Routing and Communications*, pp. 241-255, May 1994.

[21] B. V. Dao, J. Duato and S. Yalamanchili, "Configurable Flow Control Mechanisms for Fault-Tolerant Routing," *Int. Symposium on Computer Architecture,* June 1995.

[22] J. J. Dongarra, S. W. Otto, M. Snir, D. Walker, "An Introduction to the MPI Standard," *Communications of the ACM,* (to appear).

[23] J. Draper and J. Ghosh, "A Simple Analytical Model for Wormhole Routing in Multicomputer Systems," *Jou. of Parallel and Distributed Computing,* 20, pp. 202-214, 1994.

[24] J. Duato, "A necessary and sufficient condition for deadlock-free adaptive routing in wormhole networks," *Int. Conf. on Parallel Processing*, vol. I, pp. 142-149, 1994.

[25] J. Duato, "A new theory of deadlock-free adaptive routing in wormhole network," *IEEE Trans. on Parallel and Distributed systems*, vol. 4, no. 12, pp. 1320-1331, Dec. 1993.

[26] J. Duato, "Improving the efficiency of virtual channels with time-dependent selection functions," *Proc. of Parallel Architectures and Languages Europe*, 1992.

[27] J. Duato and P. Lopez, "Performance Evaluation of Adaptive Routing Algorithms for K-ary N-cubes," *Workshop on Parallel Computer Routing and Communication,* May 1994.

[28] J. Duato, "A Theory of Fault-Tolerant Routing in Wormhole Networks," *Int. Conf. on Parallel and Distributed Systems,* pp. 600-607, 1994.

[29] J. Duato, B. V. Dao, P. T. Gaughan and S. Yalamanchili, "Scouting: Fully Adaptive, Deadlock-Free Routing in Faulty Pipelined Networks," *Int. Conf. on Parallel and Distributed Systems,* Dec. 1994.

[30] P. T. Gaughan and S. Yalamanchili, "A Family of Fault Tolerant Routing Protocols for Direct Multiprocessor Networks," *IEEE Trans. on Parallel and Distributed Systems*, pp. 482-497, May 1995.

[31] C. J. Glass and L. M. Ni, "The Turn model for adaptive routing," *Jou. of the ACM*, pp. 874-902, vol. 41, Sept. 1994.

[32] C. J. Glass and L. M. Ni, "Maximally Fully Adaptive Routing in 2D Meshes," *Int. Conf. on Parallel Processing,* Aug. 1992.

[33] C. J. Glass and L. M. Ni, "Adaptive routing in mesh-connected networks," *Intl. Conference on Distributed Computing Systems*, pp. 12-19, 1992.

[34] C. J. Glass and L. M. Ni, "Fault-tolerant wormhole routing in meshes," *Intl. Symposium on Fault-Tolerant Computing*, pp. 240-249, 1993.

[35] I. S. Gopal, "Prevention of Store-and-Forward Deadlock in Computer Networks," *IEEE Trans. on Communications,* pp. 1258-1264, Dec. 1985.

[36] L. Gravano, G. D. Pifarre, P. E. Berman, and J. L. C. Sanz, "Adaptive deadlock- and livelock-free routing with all minimal paths in torus networks," *IEEE Trans. on Parallel and Distributed Systems* vol. 5, no. 12, pp. 1233-1251, Dec. 1994.

[37] *A Touchstone DELTA System Description*, Intel Corporation, Santa Clara, 1990.

[38] *Paragon XP/S Product Overview*, Intel Corporation, 1991.

[39] C. R. Jesshope, P. R. Miller, and J. T. Yanchev, "High Performance Communications in Processor Networks," *Int. Symp. on Computer Architecture,* pp. 150 - 157, May 1989.

[40] P. Kermani and L. Kleinrock, "Virtual Cut-through: A New Computer Communication Switch Technique," *Computer Network*, vol. 3, pp. 267-286, 1979.

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

[42] J. Kim and C. R. Das, "Modeling Wormhole Routing in a Hypercube," *IEEE Trans. on Computers,* pp. 1052-1060, Dac. 1991.

[43] J. H. Kim and A. A. Chien, "Network Performance under Bimodal Traffic Loads," *Jou. of Parallel and Disributed Computing,* 28, pp. 43-64, 1995.

[44] J. H. Kim, Z. Liu and A. A. Chien, "Compressionless Routing : A Framework for Adaptive and Fault- Routing", *Intl. symposium on Computer Architecture*, pp. 289-300, April 1994.

[45] S. Konstantinidou and L. Snyder, "Chaos Router: Architecture and Performance," *Int. Symp. on Computer Architecture,* pp. 212-221, May 1991.

[46] C. E. Leiserson, Z. S. Abuhamdeh, D. C. Douglas, C. R. Feynman, M. N. Ganmukhi, J. V. Hill, B. C. Kuszmaul, M. A. S. Pierre, D. S. Wells, M. C. Wong, S. W. Yang, and R. Zak, "The Network Architecture of the Connection Machine CM-5," *ACM Symposium on Parallel Algorithms and Architectures,* pp. 544-557, 1992.

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

[48] R. Libeskind-Hadas and E. Brandt, "Origin-Based Fault-Tolerant Routing in the Mesh," *Symp. on High Performance Computer Architecture*, pp.102-111, Jan. 1995.

[49] X. Lin, P. K. McKinley, and L. M. Ni, "The message flow model for routing in wormhole-routed networks," Int. Conf. on Parallel Processing, vol. I, pp. 294-297, 1993.

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

[51] P. Lopez and J. Duato, "Deadlock-free Adaptive Routing Algorithms for the 3D-Torus: limitations and solutions," *Proc. of Parallel Architectures and Languages Europe*, 1993.

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

[53] NCUBE Company, *NCUBE-2 Processor Manual*, 1990.

[54] L. M. Ni, Y. Gui, and S. Moore, "Performance Evaluation of Switch-Based Wormhole Networks," *Int. Conf. on Parallel Processing, 1995.*

[55] L. M. Ni and P. K. McKinley, "A survey of Wormhole Routing Techniques in direct networks", *IEEE Computers*, vol. 26, no. 2, pp. 62-76, Feb. 1993.

[56] M. Noakes, D. A. Wallach and W. J. Dally, "The J-Machine Multicomputer: An Architectural Evaluation," *Intl. Symposium on Computer Architecture*, pp. 224-235, 1993.

[57] H. Park and D. P. Agrawal, "Efficient Deadlock-Free Wormhole Routing in Shuffle Based Networks," *IEEE Symp. on Parallel and Distributed Processing,* pp. 92-99, 1995.

[58] T. M. Pinkston and S. Warnakulasurya, "On deadlocks in Interconnection Networks," International Symposium on Computer Architecture, pp. 38-49, June, 1997.

[59] D. S. Reeves, E. F. Gehringer, and A. Chandiramani, "Adaptive routing and deadlock recovery: A simulation study," 4th Conference on Hypercube Concurrent Computers and Applications, Mar. 1989.

[60] J. Rexford and K. G. Shin, "Support for Multiple Classes of Traffic in Multicomputer Routers," Parallel Computer Routing and Communication Workshop, Computer Science Lecture Notes, vol. 853, pp. 116-130, 1994.

[61] M. R. Samatham and D. K. Pradhan, "The de Bruijn Multiprocessor Network: A versatile parallel processing and sorting network for VLSI," *IEEE Trans. on Computers,* C-38, pp. 567-581, Apr. 1989.

[62] L. Schwiebert and D. N. Jayasimha, "Optimally Fully Adaptive Minimal Wormhole Routing for Meshes," *Jou. of Parallel and Distributed Computing,* pp. 56-70, v. 27, 1995.

[63] L. Schwiebert and D. N. Jayasimha, "A Necessary and Sufficient Condition for Deadlock-Free Wormhole Routing," *Jou. of Parallel and Distributed Computing,* 32, pp. 103-117, 1996.

[64] S. Scott and G. Thorson, "Optimized Routing in the Cray T3D,", *Intl. Workshop on Parallel Computer Routing and Communication*, pp. 281-294, 1994.

[65] C. B. Stunkel, D. G. Shea, B. Abali, M. M. Denneau, P. H. Hochschild, D. J. Joseph, B. J. Nathanson, M. Tsao, and P. R. Varker, "Architecture and Implementation of Vulcan," *Int. Parallel Processing Symposium*, pp. 268-274, April 1994.

[66] C. Su and K. G. Shin, "Adaptive deadlock-free routing in multicomputers using only one extra channel," *Intl Conference on Parallel Processing*, vol. 1, pp. 227-231, Aug. 1993.

[67] Y. J. Suh, B. V. Dao, J. Duato, and S. Yalamanchili, "Software based fault-tolerant oblivious routing in pipelined networks," *Int. Conf. on Parallel Processing*, Aug. 1995.

[68] J. Upadhyay, V. Varavithya, and P. Mohapatra, An efficient and balanced routing in two-dimensional meshes. In *Proc. of the First Intl. Symposium on High Performance Computer Architecture*, pp. 112-122, Jan. 1995.

[69] J. Upadhyay, V. Varavithya, and P. Mohapatra, "A Traffic-Balanced Adaptive Routing Scheme for Two-Dimensional Meshes," *IEEE Trans. on Computers*, pp. 190-197, Feb. 1997.

[70] J. Upadhyay, V. Varavithya, P. Mohapatra, "Routing Algorithms for Torus Networks," *Intl. Conference on High Performance Computing*, pp. 743-748, 1995.

[71] V. Varavithya, J. Upadhyay and P. Mohapatra, "An Efficient Fault-Tolerant Routing Scheme for Two-Dimensional Meshes," *Intl. Conference on High-Performance Computing*, pp. 773-778, 1995.