Fault-Tolerant Wormhole Routing in Tori*

Suresh Chalasani
Electrical & Comp. Engr. Dept.
Univ. of Wisconsin-Madison
Madison, WI 53706-1691

Rajendra V. Boppana
Div. of Math. and Computer Science
The Univ. of Texas at San Antonio
San Antonio, TX 78249-0664

Abstract. We present a method to enhance wormhole routing algorithms for deadlock-free fault-tolerant routing in tori. We consider arbitrarily-located faulty blocks and assume only local knowledge of faults. Messages are routed via shortest paths when there are no faults, and this constraint is only slightly relaxed to facilitate routing in the presence of faults. The key concept we use is that, for each fault region, a fault ring consisting of fault free nodes and physical channels can be formed around it. These fault rings can be used to route messages around fault regions. We prove that at most four additional virtual channels are sufficient to make any fully-adaptive algorithm tolerant to multiple faulty blocks in torus networks. As an example of this technique, we present simulation results for a fully-adaptive algorithm and show that good performance can be obtained with as many as 10% links faulty.

Keywords: adaptive routing, deadlocks, fault-tolerant routing, multicomputer networks, message routing, performance evaluation, torus networks, wormhole routing.

1 Introduction

Point-to-point torus and related networks are being used in many recent experimental and commercial multicomputers and multiprocessors [1, 22, 27, 6, 2]. A (k, n)-torus network has an n-dimensional grid structure with k nodes (processors) in each dimension such that every node is connected to two other nodes in each dimension by direct communication links.

The *wormhole* (WH) switching technique by Dally and Seitz [10] has been used in many recent multicomputers [22, 1, 20, 25]. In the WH technique, a packet is divided into a sequence of fixed-size units of data, called *flits*. If a communication channel transmits the first flit of a message, it must transmit all the remaining flits of the same message before transmitting flits of another message. To avoid deadlocks among messages, multiple virtual channels are simulated on each physical channel and a pre-defined order is enforced on the allocation of virtual channels to messages. Alternatives to the wormhole switching are the virtual-cut-through [16] and store-and-forward switching techniques, which require more storage at each routing node.

For fault-free networks, some of the most important issues in the design of a routing algorithm are high throughput, low-latency message delivery, avoidance of deadlocks, livelocks and starvation, and ability to work well under various traffic patterns [12]. Given a network with faults, our approach is to use the existing network rather than recreate the original network using spare nodes and links. Therefore, for networks with faults, a routing algorithm should exhibit the following additional features: graceful degradation of performance, and ability to handle faults with only a small increase in routing complexity and local knowledge of faults—each non-faulty processor knows only the status of its neighbors.

The well-known e-cube or dimension-order routing algorithm is an example of non-adaptive routing algorithms, since always a particular path is used in routing messages between a pair of nodes even when multiple shortest paths are available. With the e-cube, even a single fault disrupts communication between multiple pairs of nodes. With increase in adaptivity, a message is more likely to find a less congested path or fault-free path. Therefore, the issue of adaptivity—the extent of choice in selecting a path between a pair of nodes in routing a message—plays an important role in designing fault-tolerant routing algorithms.

Description of the problem and results. We present a technique to enhance minimal, fully-adaptive routing algorithms for fault-tolerant routing in tori. A minimal fully-adaptive algorithm routes messages along any of the shortest paths available. The other options are nonminimal fully-adaptive algorithms and partially-adaptive algorithms. We do not consider partially-adaptive algorithms in this paper, since, in general, they do not performance well even for 2- and 3-dimensional networks [4, 17].

Adaptive, nonminimal routing algorithms could cause livelocks. To avoid livelocks, a "backup" nonadaptive routing algorithm is often used to guarantee the delivery of messages. For example, the dimension reversal schemes of Dally and Aoki [9] have the e-cube algorithm as the backup algorithm. The Tera computer [2] (which uses deflection routing with store-and-forward switching) uses a Hamiltonian path of the network for delivering messages that do not reach their destinations within a certain amount of time. Furthermore, these backup algorithms cannot handle faults; additional routing techniques or resources must be used to solve this problem.

We consider routing methods that use only local knowledge of faults. We assume that faulty processors are confined to one or more rectangular blocks. With the current technology and anticipated advances in packaging, it is reasonable to expect that each node (CPU-memory-router com-

*Suresh Chalasani's research has been supported in part by a grant from the Graduate School of UW-Madison and the NSF grants CCR-9308966 & ECS-9216308. Rajendra Boppana's research has been supported by the NSF Grant CCR-9208784.

In Proceedings of 1994 ACM International Conference on Supercomputing, July 1994

bination) of a multicomputer could be implemented as a single chip or as a multichip-module, with several such nodes placed on a printed circuit board. The block-fault model used in this paper accurately models the chip-, multichip module-, and board-level faults.

For each fault region, there exist one or more paths that pass through fault-free nodes and links and encircle the fault. For a fault in a 2D torus, there is an undirected ring of fault-free nodes and links; we refer to this ring as *fault-ring*. The fault-ring around a block fault is rectangular in shape. In this paper, we show that fault rings can be used to route messages around the fault regions using only local knowledge of faults, and without introducing deadlocks and livelocks. Our techniques thus achieve fault-tolerance without using spare nodes and physical channels.

Our techniques are especially suitable for the high-radix ($k > 2$), low-dimensional ($n = 2, 3$) tori commonly used in the recent multicomputers [25]. We show, using simulations, that graceful degradation of performance is achieved even with 10% of the links faulty.

Related results. Adaptive, fault-tolerant routing algorithms for WH and virtual cut-through switching techniques has been the subject of extensive research in recent years [7, 11, 21, 9, 14, 24, 3]. Several results have been reported for fault-tolerant routing in hypercubes; see, for example, [19, 26] and the references therein. The results for hypercube exploit the rich interconnection structure of hypercubes and are not suitable for high-radix, low-dimensional tori.

Reddy and Freitas [23] use global knowledge of faults, spare nodes, and routing tables to investigate the performance limitations caused by faults. Gaughan and Yalamanchili [13] use a pipelined circuit-switching mechanism with backtracking for fault-tolerant routing. These two results are applicable to networks with arbitrarily-shaped faults. Our interest in this paper is to design fault-tolerant routing algorithms that can be applied with local knowledge of faults. One important criterion is that the fault-free performance should not be sacrificed for fault-tolerant routing.

Often, the results developed for meshes [9, 7, 5] can be extended to tori with suitable modifications, since meshes and tori are closely related. The wraparound links in tori lead to extra deadlock possibilities, however. Therefore, if the results developed for meshes are applied with few changes, then the number of virtual channels required to avoid deadlocks may be doubled [7, 5].

The planar adaptive routing (PAR) by Chien and Kim [7] is a partially-adaptive algorithm and provides fault-tolerant routing using 6 virtual channels. But the PAR does not yield good performance for 3 or higher dimensional networks [17].

In terms of adaptivity and performance comparisons, the results by Dally and Aoki [9] are the most relevant to ours. Dally and Aoki present fault-tolerant algorithms based on the concept of dimension reversal, which occurs whenever a message takes a hop in a dimension lower compared to that of the previous hop. A message can be routed adaptively if the number of dimension reversals it has taken is less than the number of highest virtual channel class (static algorithm) or if the message finds a free channel in other outgoing channels of the current host in a finite amount of time (dynamic algorithm). The main advantage of their algorithm is that arbitrarily shaped faults can be handled. Their work presented for meshes can be easily applied to tori by taking hops on wraparound links as dimension reversal hops.

With the dimension-reversal schemes of Dally and Aoki, a message may lose its adaptivity as described above. A message that has lost adaptivity is routed by the e-cube algorithm and is not guaranteed to be delivered to its destination if there are faults in the network. Thus the number of virtual channels needed and the number of faults tolerated is highly dependent on the location of faults and the misrouting logic—which determines when to change the dimension of travel—used.

In contrast, our algorithms can tolerate any number and combination of rectangular faulty blocks with simple logic, and require only four virtual channels[1] more than that required for the original adaptive algorithm. Furthermore, our algorithms guarantee that each and every message injected into the network is delivered. This result compares well with our earlier result that four extra virtual channels are sufficient for routing in meshes with faults [5].

Organization of the paper. Section 2 describes the fault-model and the concept of fault-rings. Section 3 presents our method to enhance fully-adaptive algorithms to tolerate multiple faulty blocks in two-dimensional tori. Section 4 gives the simulation results on the performance of a fully-adaptive algorithm, which is originally developed for fault-free networks and modified for fault-tolerant routing using our method. Section 5 extends the proposed technique to higher dimensional tori. Section 6 concludes the paper.

2 Preliminaries

A (k, n)-torus (also called k-ary n-cube) has n dimensions, numbered from 0 to $(n-1)$, and $N = k^n$ nodes. Each node is uniquely indexed by an n-tuple in radix k. Each node is connected via communication links to two other nodes in each dimension. The neighbors of the node $x = (x_{n-1}, \ldots, x_0)$ in dimension i are $(x_{n-1}, \ldots, x_{i+1}, x_i \pm 1, x_{i-1}, \ldots, x_0)$, where addition and subtraction are modulo k. A link is said to be a wraparound link if it connects two neighbors whose addresses differ by $k-1$ in dimension i, $0 \leq i < n$. A (k, n)-mesh is a (k, n)-torus with the wraparound connections missing.

We assume that each communication link actually represents two unidirectional physical communication channels. The link between nodes x and y is denoted by $< x, y >$.

In the remainder of this section, we describe the fault model considered in this paper and the concept of fault-rings, which are created by faults. To simplify presentation, we discuss these concepts for two-dimensional (2D) tori. We label the sides of a 2D torus as North, South, East and West.

2.1 Wormhole routing concepts for fault-free networks

Dally and Seitz [10] used the concept of channel dependency graphs and multiple virtual channels to show deadlock free wormhole routing on various networks. The channel dependency graph is formed as follows. The virtual channels of the network form the nodes of the channel dependency graph; there is a directed edge from node u to v if there is a message that has acquired the virtual channel u and is waiting or has acquired the virtual channel v. A channel dependency graph may be formed at any instant of the network operation for any routing algorithm. A routing algorithm for an interconnection network is deadlock free if there are no cycles in the channel dependency graph formed at any time.

[1] Throughput this paper, we indicate the number of virtual channels on per physical channel basis.

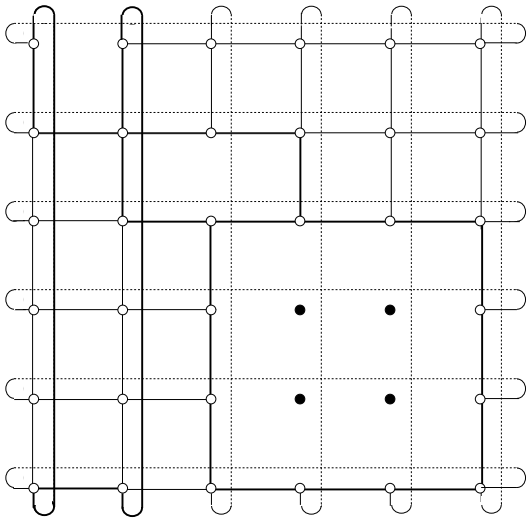

Figure 1: Three fault regions and their associated fault rings in a 6×6 torus.

In our proof methods, we consider maximal channel dependency graphs and show that they are acyclic. A maximal channel dependency graph for a network and an algorithm is obtained by placing edges from virtual channel u to virtual channel v if the routing algorithm allows the use of v after using u for any message. All channel dependency graphs, formed during routing, are subgraphs of the maximal dependency graph for the routing algorithm used.

We consider only minimal routing methods as per which a message always moves closer to its destination when not blocked by a fault.

2.2 The fault model

We consider both node and link faults. All the links incident on a faulty node are considered faulty. Status signals are continually sent on physical channels and monitored by processors. Missing or incorrect sequences of signals indicate malfunction of the link or the processor sending them. In either case, the processor that is connected at the other end stops using the link. We assume that all faults are non-malicious faults; that is, a failed component simply ceases to work. Therefore, only non-faulty processors generate messages. Furthermore, messages are destined only to fault-free processors. These assumptions are commonly made in fault analyses in literature.

We model multiple simultaneous faults, which could be connected or disjoint. We assume that the mean time to repair faults is quite large, a few hours to many days, and that the existing fault-free processors are still connected and thus should be used for computations in the mean time.

If global knowledge of faults is to be maintained in the system, too many status messages (as faulty processors are repaired and become functional and working processors become faulty) may have to be transmitted on the network. For massively parallel processors, faults could occur frequently and routing tables could be expensive. Also, routing algorithms that depend on global fault information should have alternate schemes to route messages during the transition period—the interval from the time a fault occurred to the earliest time it is known globally. Therefore, we develop fault-tolerant algorithms, for which it is sufficient if each non-faulty processor knows the status of the links incident on it.

A *fault set* is defined as the set F of faulty nodes and links. For example, the fault-set $F = \{(3,3),(3,4),(4,3),(4,4),< (0,0),(0,1) >,< (1,2),(2,2) >\}$ represents four node faults and two link faults in the two-dimensional network shown in Figure 1.

We assume that faults in a 2D torus have *rectangular* shapes. A set F of faulty nodes and links in a 2D torus is said to have a rectangular shape, if there is a rectangle in the torus such that (a) there are no faulty components on the boundary of the rectangle, (b) the interior of the rectangle[2] includes all faulty components in F, and (c) the interior of the rectangle contains no component that is not present in F.

For example, the set $F_1 = \{(3,3),(3,4),(4,3),(4,4)\}$ of faulty nodes shown in Figure 1 is rectangular, since the interior of the rectangle — with corners $(2,2)$, $(2,5)$, $(5,2)$, and $(5,5)$ — includes all faulty components in F and no non-faulty component (recall that a processor fault implies that all links incident on it are faulty). However, the set of faulty links $F_2 = \{< (1,1),(1,2) >,< (1,2),(2,2) >,< (2,2),(2,1) >,< (2,1),(1,1) >\}$ in a 6×6 torus is not rectangular, since any rectangle with nonfaulty elements on its boundary contains at least one element not in F. The faulty link $< (1,2),(2,2) >$ is an example of a rectangular fault region, since the interior of the rectangle with corners $(1,1)$, $(1,3)$, $(2,1)$, and $(2,3)$ contains only the faulty link. The faulty link $< (0,0),(0,1) >$ in Figure 1 is considered rectangular; the rectangle that covers the faulty link has processors $(1,0),(1,1),(5,1)$ and $(5,0)$ as its corners.

We call this model the *block-fault* model. Chien and Kim [7] call this model the convex fault model. An f-region is the fault region of the torus given by a block-fault. Under the block-fault model, the fault-set in a 2D torus can be written as a union of disjoint smaller fault sets, each of which denotes an f-region. For example, the fault set F in Figure 1 is in fact the union of three disjoint f-regions $\{(3,3),(3,4),(4,3),(4,4)\}$, $\{< (0,0),(0,1) >\}$ and $\{< (1,2),(2,2) >\}$. We also assume that faults do not disconnect the network, an assumption commonly made in the literature [7, 9]. If the network becomes disconnected, our results given in [5] can be applied on the resulting subnetworks, which are meshes.

There are many reasons to consider block faults. First, they model several common fault scenarios such as faults of isolated nodes and links and consecutive nodes in a row or column. Second, an arbitrarily-shaped fault can be modeled as a block-fault, albeit by labeling some non-faulty processors and/or links as faulty [7]. Finally, the block fault model accurately models faults at the chip, multichip module, and board levels.

2.3 Fault rings

Conceptually, fault regions may be considered as islands of faults in a sea of communication channels and nodes. In the same manner a ship is navigated around an island, it should be feasible to route a message around fault regions. For this purpose, we use the concept of *fault rings*, denoted f-rings.

For each f-region in a network with faults, it is feasible to connect the fault-free components around the region to form a ring or chain. This is the fault ring for that region and consists of the fault-free nodes and channels that are adjacent

[2] Interior of a rectangle is defined as the set of processors and links that are not on the boundary of the rectangle.

(row-wise, column-wise, or diagonally) to one or more components of the fault region. The f-ring of a block-fault has rectangular shape. For example, the f-ring of the node fault region $\{(3,3),(3,4),(4,3),(4,4)\}$ in Figure 1 passes through the fault-free nodes

$$(2,2),(2,3),(2,4),(2,5),(3,5),(4,5),$$
$$(5,5),(5,4),(5,3),(5,2),(4,2),(3,2)$$

as shown in Figure 1. The f-ring associated with the link fault region $\{< (1,2),(2,2) >\}$ has nodes $\{(i,j) \mid 1 \leq i \leq 2, 1 \leq j \leq 3\}$ in its perimeter. The f-ring for the faulty link $< (0,0),(0,1) >$ has nodes (1,0), (0,0), (5,0), (5,1), (0,1), and (1, 1) on its perimeter.

A fault-free node is in the f-ring only if it is at most two hops away from a faulty node or is adjacent to a node with a faulty-link incident on it. There can be several fault rings, one for each f-region, in a faulty network with multiple faults. Up to two f-rings in a 2D torus may have a common link, and up to four f-rings may have a common node. For example, nodes $(2,2),(2,3)$ and the link between them are common to two f-rings in Figure 1.

A set of fault rings are said to overlap if they share one or more links. For example, the two f-rings with (1,1) and (2,2) as the northwest corner nodes overlap in Figure 1, since they share a link.

An f-ring represents a two-lane path to a message that needs to go through the f-region contained by the f-ring. Thus, an f-ring simulates four paths to route messages in two dimensions. Depending on the size of the f-region, physical channels in an f-ring may need to handle a large amount of traffic compared to the other fault-free physical channels. Further, routing messages around one or more fault-rings creates additional possibilities for deadlocks. Hence, wormhole routing algorithms must be designed to handle additional congestion and deadlocks caused by faults.

When a fault occurs, the f-ring around it can be formed in a distributed manner using a two-step process. In the first step, each processor that detected a faulty link sends this message to its neighbors in other dimensions. Based on the set of messages received, each node that is to be on the f-ring determines its position in the f-ring. There are eight possible positions for a processor to be in an f-ring: North West corner, North, North East corner, East, South East corner, South, South West corner, and West. For more details, the reader is referred to [5].

3 Fault-tolerant routing algorithms for 2D tori

In this section, we present techniques using which any fully-adaptive routing algorithm can tolerate multiple rectangular fault regions in a 2D torus. There are two types of fully-adaptive algorithms: *strongly* and *weakly* fully-adaptive algorithms. In a *strongly* fully-adaptive algorithm, a blocked message retains its full-adaptivity. In contrast, with a *weakly* fully-adaptive algorithm, a blocked message may lose some or all of its adaptivity; so, it is more difficult to modify weakly fully-adaptive algorithms for fault-tolerant routing. Our results are applicable only to *strongly* fully-adaptive algorithms. Henceforth, a reference to a fully-adaptive algorithm is actually a reference to a strongly fully-adaptive algorithm. For the most part of the section, we assume that the faults in a torus are such that the resulting f-rings are nonoverlapping. However, at the end of the section, we indicate how fault-tolerant routing can be achieved with overlapping f-rings.

A message is said to be blocked by faults at node x if there is no fault-free link $< x, y >$ such that the hop from x to y is along the shortest path from x to d.

The following lemma forms the basis for the result presented in this section.

Lemma 1 *Consider a 2D torus with multiple rectangular fault blocks. Suppose that a message with destination d is being routed in the torus using a fully-adaptive routing algorithm. If the message is blocked at a node, say x, then the addresses of x and d differ in exactly one dimension.*

Proof: We prove this by contradiction. Assume that the message is blocked at node x and that x and d differ in both dimensions. It can be easily verified that under the block-fault model, a nonfaulty node can have faulty links incident in at most one dimension. If x has no faulty links incident on it, then a hop in either dimension will take the message closer to its destination. If x has one or both links in a dimension faulty, then one of the links in the other dimension takes the message closer to its destination. In all cases, the message can move closer to its destination and cannot be blocked. This contradicts the assumption that the message is blocked at x. ∎

To distinguish blocked messages from others, we use the concept of message status, which could be *unaffected* or *affected*. A message is injected into the network with the *unaffected* status. When an unaffected message is blocked by a fault, its status is changed to *affected*, and it retains this status for the remainder of its journey. In our method, when a message becomes *affected*, it starts using a special class of virtual channels. The class of virtual channels used by an *affected* message is based on the dimension and direction it needs to travel to reach its destination.

Consider a message with destination $d = (d_1, d_0)$. Let it become affected at node $x = (x_1, x_0)$. From Lemma 1, it is clear that the message needs to travel in only one dimension; that is, either $x_1 = d_1$ or $x_0 = d_0$. In each dimension, there are two possible directions. Thus, there can be four different types of affected messages: 0^+, 0^-, 1^+, and 1^-. The message is termed a 0^+-message if $d_1 = x_1$ and $d_0 > x_0$. Furthermore, it is a 0^+M message if it will not use a wraparound link in dimension 0; otherwise it is a 0^+W-message. There are eight types of affected messages for a 2D torus and are given in Table 1.

When a message becomes affected, its type is determined and assigned. This type is used to determine the virtual channel class to be used for remainder of the message's journey, and the orientation (direction of travel) to be used when routed on an f-ring. Table 1 gives this information for each of the eight possible message types.

Routing *affected* messages. The fault-tolerant version of a generic fully-adaptive algorithm $\mathcal{F}$, denoted $\mathcal{F}_f$, uses four virtual channels — c_0, c_1, c_2 and c_3 — in addition to those used by $\mathcal{F}$. Channel c_0 is used exclusively by 0^+ affected messages (both 0^+M and 0^+W); similarly, virtual channel classes c_1, c_2, c_3, respectively, are used exclusively by $0^-, 1^+, 1^-$ messages, respectively. Rules to route various messages are specified in procedure Fully-Adaptive-2D (Figure 2).

Example. In Figure 4, three faulty nodes — (1, 0), (4, 1) and (5, 4) — are present. There are three f-rings corresponding to these three faulty nodes. Several messages in

Table 1: Virtual Channels and F-Ring Orientations Used by Affected Messages

Message Type	Conditions Satisfied	Virtual Channel	F-Ring Orientation
0^+M	$d_1 = x_1$ & $d_0 > x_0$ & $(d_0 - x_0) \leq \lfloor k/2 \rfloor$	c_0	Clockwise
0^+W	$d_1 = x_1$ & $d_0 > x_0$ & $(d_0 - x_0) > \lfloor k/2 \rfloor$	c_0	Counter-Clockwise
0^-M	$d_1 = x_1$ & $x_0 > d_0$ & $(x_0 - d_0) \leq \lfloor k/2 \rfloor$	c_1	Clockwise
0^-W	$d_1 = x_1$ & $x_0 > d_0$ & $(x_0 - d_0) > \lfloor k/2 \rfloor$	c_1	Counter-Clockwise
1^+M	$(d_1 - x_1) \leq \lfloor k/2 \rfloor$ & $d_1 > x_1$ & $d_0 = x_0$	c_2	Clockwise
1^+W	$(d_1 - x_1) > \lfloor k/2 \rfloor$ & $d_1 > x_1$ & $d_0 = x_0$	c_2	Counter-Clockwise
1^-M	$(x_1 - d_1) \leq \lfloor k/2 \rfloor$ & $d_1 < x_1$ & $d_0 = x_0$	c_3	Clockwise
1^-W	$(x_1 - d_1) > \lfloor k/2 \rfloor$ & $d_1 < x_1$ & $d_0 = x_0$	c_3	Counter-Clockwise

($x = (x_1, x_0)$ is the node at which the message is affected, and $d = (d_1, d_0)$ is its destination)

PROCEDURE FULLY-ADAPTIVE-2D(M) /* Uses a generic fully-adaptive algorithm $\mathcal{F}$ */
/* Uses four additional virtual channels c_0, c_1, c_2, c_3 */

RULE 1: If M is unaffected, reserve virtual channels and links according to $\mathcal{F}$.

RULE 2: If M is a 0^+-message, route it using virtual channel c_0 for the rest of its journey. Virtual channel c_1 is used exclusively for 0^--messages, c_2 for 1^+-messages and c_3 for 1^--messages.

Let d be the dimension in which the message was blocked when it was affected. Let d' be the other dimension. Let a *legal* hop be defined as a hop in d that takes the message closer to its destination.

CASE 1: If the current host and destination match in d', and the legal hop is available, it is taken (see Figure 3).

CASE 2: If M is a message on an f-ring, it routed using the virtual channel and orientation shown in Table 1 until it reaches the other parallel side of the f-ring such that current host and destination match in d' (see Figure 3).

Figure 2: Fault-tolerant routing logic for 2D tori.

this figure and their routes are indicated in Table 2. For example, the message from $(5,2)$ to $(3,1)$ is first routed from $(5,2)$ to $(5,1)$. It becomes affected by fault $(4,1)$ at $(5,1)$. Since it needs to travel from $(5,1)$ to $(3,1)$, it is labeled as a 1^-M message. From the rules for 1^-M messages (see Table 1), it is routed in the clockwise orientation using virtual channel c_3.

Theorem 1 *Assume that the fully-adaptive routing algorithm $\mathcal{F}$ correctly routes messages and is deadlock-free. The fault-tolerant fully-adaptive routing algorithm $\mathcal{F}_f$ described by* RULES 1–2 *(in procedure Fully-Adaptive-2D) is deadlock-free in the presence of multiple rectangular fault blocks and delivers messages correctly between any pair of nonfaulty nodes.*

Proof. Algorithm $\mathcal{F}_f$ correctly routes unaffected messages between any pair of nonfaulty nodes, since $\mathcal{F}$ correctly routes messages. Since the virtual channels used for affected messages are different from those used for unaffected messages, the statement is true for all unaffected messages. To complete the proof, we need to show that affected messages are also routed correctly without deadlocks and livelocks.

To see that the procedure Fully-Adaptive-2D correctly delivers messages, observe that (i) an affected message is misrouted only around an f-ring, (ii) a message, once it leaves an f-ring will never revisit it, (iii) an affected message takes only a finite number of hops on each f-ring, and (iv) there are a finite number of f-rings in the torus. These four observations show that a message is delivered to its destination in a finite number of hops and that there are no livelocks in the system.

We next prove the deadlock-freedom of the procedure Fully-Adaptive-2D. Unaffected messages cannot be involved in a deadlock, since $\mathcal{F}$ is deadlock-free and since they do not require or wait for the virtual channels c_0, c_1, c_2 and c_3. Among affected messages, 0^+-messages use only virtual channel c_0; similarly, a distinct virtual channel is used for messages of each type. Thus, it is enough if we show that there are no deadlocks among 0^+-messages.

There are two types of 0^+ messages: 0^+M and 0^+W. The part of the network used by 0^+M messages consists of row channels in the East direction and column channels in the North direction in the West columns and South channesl in the East columns of f-rings. Its underlying graph is acyclic. Similary, 0^+W uses an acyclic network of c_0 channels. Therefore, a deadlock among 0^+-messages involves both 0^+M 0^+W messages. But 0^+M and 0^+W messages use disjoint sets of physical channels. This is obvious for the physical channels that are not part of the f-ring, since 0^+M messages travel from West to East, and 0^+W messages from East to West when not misrouted. From Table 1, it is clear that 0^+M and 0^+W messages reserve virtual channels on physical channels in clockwise and counter-clockwise directions, respectively, on an f-ring. Therefore, there is no dependency among the 0^+ messages. Similarly, we can prove the deadlock freedom for other types of messages. ∎

Fault-tolerant routing with overlapping f-rings. Thus far, we have assumed that faults are such that the f-rings

Table 2: A Few Messages in the Faulty Network of Figure 4

Src	Dest	Affected at	Affected by Faulty Node	Message Type	Virtual Channel	Orientation	Path Indicated by
(0, 0)	(2, 0)	(0, 0)	(1, 0)	1^+M	c_2	Clockwise	Solid thick lines
(1, 1)	(1, 5)	(1, 1)	(1, 0)	0^+W	c_0	Counter-Clockwise	Dashed thick lines
(0, 4)	(4, 4)	(0, 4)	(5, 4)	1^+W	c_2	Counter-Clockwise	Solid thick lines
(5, 3)	(5, 5)	(5, 3)	(5, 4)	0^+M	c_0	Clockwise	Dashed thick lines
(5, 2)	(3, 1)	(5, 1)	(4, 1)	1^-M	c_3	Clockwise	Dashed thick lines
(4, 2)	(4, 0)	(4, 2)	(4, 1)	0^-M	c_1	Clockwise	Solid thick lines

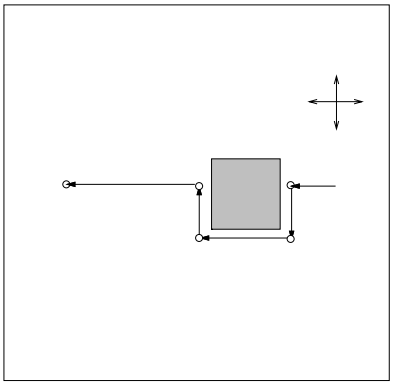

Figure 3: Routing of affected messages in a 2D torus using procedure Fully-Adaptive-2D. The message destined to (D_1, D_0) becomes affected at (X_1, X_0). Here $X_1 = Y_1 = D_1$. Routing of the message from (X_1, X_0) to (Y_1, Y_0) is done using CASE 2 of RULE 2 of procedure Fully-Adaptive-2D, and routing from (Y_1, Y_0) to (D_1, D_0) is done using CASE 1.

do not overlap. However, this is not a serious restriction. If f-rings overlap, then deadlock-free fault-tolerant routing may be provided using either one of the following methods.

- When two f-rings overlap, deadlocks may occur when, for example, 0^+M and 0^+W messages use the same physical channels in the column shared by the overlapping f-rings. This can be avoided by using two virtual channels (instead of one) for 0^+ messages. Similarly, two virtual channels for each of $0^-, 1^+, 1^-$ message types are needed. Therefore, this technique requires eight extra virtual channels.

- An alternative is to route an affected message such that it does not use wraparound links in the only dimension it needs to travel at the time it is affected. For example, a 0^+M message is routed as before. But a 0^+W message is also routed as a 0^+M message; this means that a 0^+W message is not routed along its shortest path. This ensures that the physical channels used by 0^+ messages form an acyclic directed graph. Since each message type uses a distinct class of virtual channels, the routing is deadlock free.

4 Simulation results

To study the performance issues, we have developed a cycle-by-cycle simulator. This simulator can be used for wormhole routing in k-ary n-cubes with and without faults. In this section, we present simulation results on the performance of the negative-hop (NHop) algorithm [4]. The NHop provides

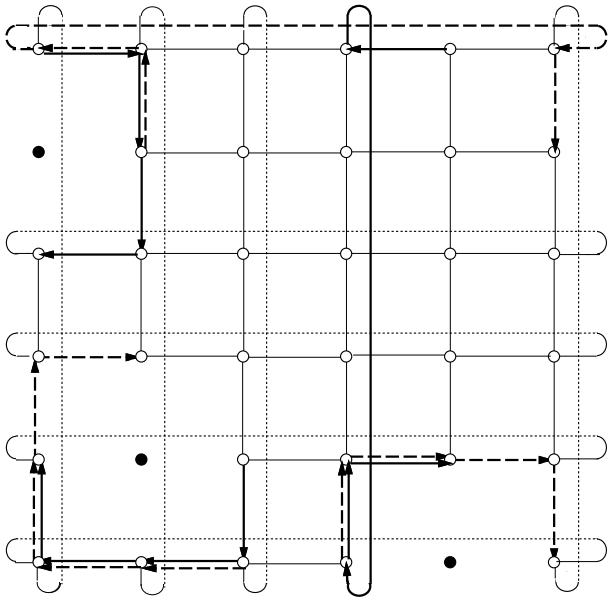

Figure 4: Routing of three messages using the fault-tolerant version of a fully-adaptive routing algorithm. There are three f-rings corresponding to the three faulty nodes $(1, 0)$, $(4, 1)$ and $(5, 4)$. Various messages in this network and how they are routed is indicated in Table 2.

minimal, fully-adaptive, and deadlock-free routing in fault-free tori using $\lfloor n\lceil k/2 \rceil / 2 \rfloor + 1$ virtual channels.

The negative hop wormhole algorithm is based on the store-and-forward algorithm by Gopal [15]. To use the negative hop algorithm, the network is colored, and each node is given a label corresponding to its color. A hop by a message is a negative hop if it moves from a node with higher label to a node with lower label. Any other hop is a nonnegative hop. Messages when injected have 0 negative hops and are routed minimally when there are no faults. If a message has taken $i \geq 0$ negative hops, then it uses virtual channels of class i for its next hop.

In our simulations, we have used the NHop algorithm, developed originally for fault-free networks, and fortified it with four additional channels and the fault-tolerant logic described in Section 3.

We have simulated a 16×16 torus for the uniform traffic pattern and 20-flit messages. The virtual channels on a physical channel are demand time-multiplexed, and it takes one cycle to transfer a flit on a physical channel. The message interarrival times are geometrically distributed. We use the uniform traffic model.

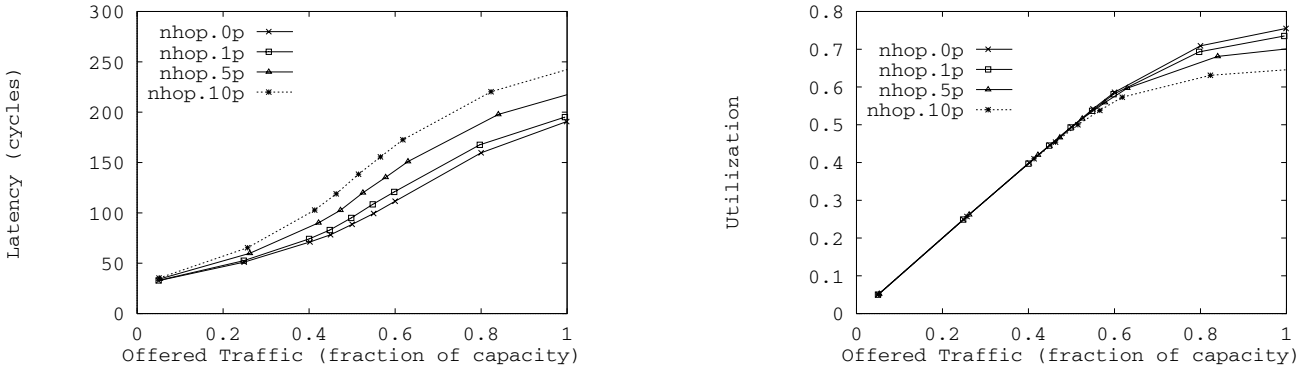

Figure 5: Performance of the NHop algorithm for uniform traffic in a 16×16 torus with various faults. The extension dp indicates the results for d% faults. 16 virtual lanes per physical channel are used: twelve lanes are distributed among the nine fault-free virtual channels, and one lane is provided for each of the four extra virtual channels required for fault-tolerant routing.

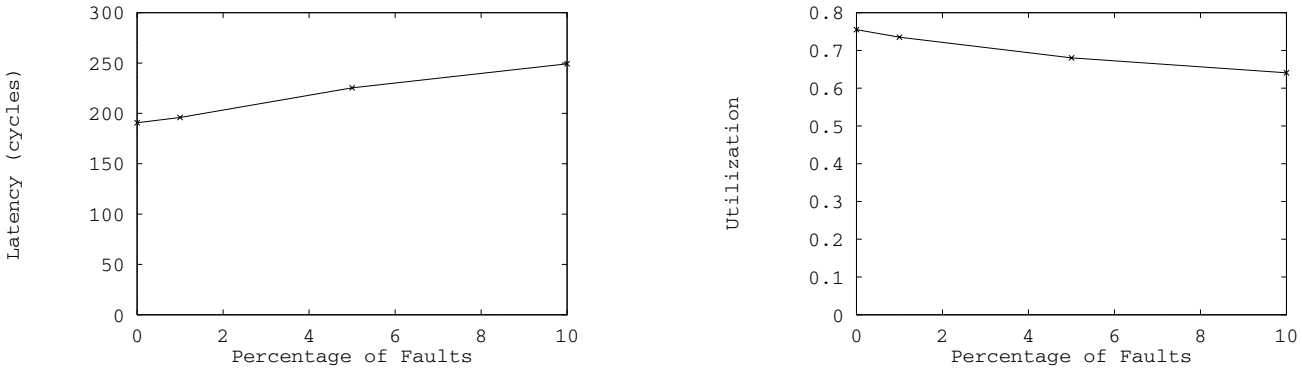

Figure 6: Performance of the NHop algorithm for uniform traffic in a 16×16 torus for different number of faults. One lane per virtual channel and 20 virtual channels are used for this algorithm. For each fault case with a certain number of faults, 10 different fault sets are randomly generated and the performance metrics are averaged.

We use bisection utilization and average message latency as the performance metrics. The bisection utilization (ρ_b) is defined as

$$(\#\text{ messages across the bisection/cycle}) * \frac{\text{Message length}}{\text{Bisection BW}}.$$

The bisection bandwidth (BW) is defined as the maximum number of flits that can be transferred across the bisection in a cycle, and is proportional to the number of nonfaulty links in the bisection of the network. The maximum value of ρ_b is 1.0. For fault-free networks with uniform traffic, the bisection utilization and channel utilization are the same. For networks with faults, they differ. But bisection bandwidth is more easily tractable and provides a consistent measure of performance. The half-width of the 95% confidence interval for each point shown in the graphs is within 4% of the value reported.

Dally [8] has shown that providing multiple lanes improves the performance of wormhole algorithms considerably. In view of this fact, we have used 16 virtual lanes per physical channel. The fault-tolerant NHop has 13 virtual channel classes (9 for fault-free routing and 4 extra for fault-tolerant routing); we distributed 12 lanes among the 9 normal, fault-free classes and allocated 1 lane to each of the four special, fault-tolerant classes.

To facilitate simulations at and beyond the normal saturation points for the routing algorithm, we have limited the injection by each node. This injection limit is independent of the message interarrival time; the motivation for injection control is due to Lam [18]. After some experimentation, we have chosen an injection limit of 4 for the NHop; that is, a node is not allowed to inject a new message if four or more messages generated by it are still in the node. Too high an injection limit leads to uncontrolled latencies at saturation; too low an injection limit reduces throughputs around the saturation slightly. For the value we have selected, there is little effect on the latency and throughput achieved by the algorithm prior to the saturation of network.

4.1 Performance for various fault cases

We have simulated a 16×16 torus with 1%, 5%, and 10% of the total network links faulty. Specificly, for the 1% case, we have set, randomly, a node and link faulty; since 4 links are incident on a node, 5 of the 512 links in the network are faulty. For the 5% fault case, we have set 4 nodes and 10

links faulty; for the 10% fault case, we have set 9 nodes and 15 links faulty. In each case, we have randomly generated the required number of faulty nodes and links. To see the performance degradation with faults, we have simulated the routing algorithm on a fault free torus also.

Since we have simulated only isolated faults, a slightly more flexible version of fault-tolerant logic can be used without creating deadlocks. This flexible version uses any of the orientations—clockwise or counter-clockwise—to route any affected message. The simulation results reported in this section are for the NHop fortified with this flexible fault-tolerant logic. We have incorporated two more improvements that are specific to the NHop algorithm: (1) a message that has taken i negative hops can use virtual channels in any of classes $0, \ldots, i$; (2) an affected message is allowed to use channels in normal classes even for misrouting until it takes more negative hops than the number of normal virtual channel classes, at which point one of the four special channels is used for the remainder of its journey. These changes do not introduce any deadlocks among messages routed by the NHop algorithm.

The results for various fault cases are given in Figure 5. For the fault-free network, the NHop has a peak utilization of 0.755 at a latency of 191 cycles. The NHop shows a graceful degradation of performance in the presence of faults. The message latencies with faults are higher; the utilization ranges from 0.648 to 0.735.

4.2 Peak performance

Comparative performance across different fault cases in Figure 5 is specific to the fault sets used. Therefore, we have further simulated the NHop for 1, 5, and 10 percent faults. For each case, we have simulated 10 different fault sets for 100% traffic load. (The injection control helps us here; otherwise, we would have to perform the tedious task of determining the saturation point for each fault set and for each fault case.) The values obtained from the ten different fault sets are averaged and shown in Figure 6.

As the number of faults is increased, the latency increases steadily and the utilization drops steadily. Comparing the fault-free case and 10%-faults case, we note that NHop has 31% increase in latency and 15% decrease in throughput.

Our previous results [5] indicate that the fault-tolerant version of NHop exhibits a similar graceful degradation in performance in meshes with faults. In particular, the NHop for mesh exhibits a 20% drop in throughput from the fault-free case to the case with 10% faults. Dally and Aoki [9] indicate that the dynamic dimension-reversal algorithm exhibits a similar graceful degradation of throughput for meshes.

5 Fault-tolerant routing in multidimensional tori

In this section, we extend the results of Section 3 to multidimensional tori using the results for 2D and 3D tori as the base cases.

The block-fault model for n-dimensional tori is defined as follows. An n-dimensional box has a base node $x = (x_{n-1}, \ldots, x_0)$ and apex node $y = (y_{n-1}, \ldots, y_0)$ and the set of nodes of the form $t = (t_{n-1}, \ldots, t_0)$ such that $x_i \leq t_i \leq y_i$, for $0 \leq i < n$. If a fault set is contained in an n-dimensional box such that the interior of the box has only the faulty components and none on its exterior, then the fault-set represents a block-fault.

A set F of faulty nodes and links in a (k, n)-torus is said to be a *block-fault* if F can be written as the union of disjoint subsets F_1, F_2, ..., F_r such that each F_i is a faulty-block by itself.

It is noteworthy that the n-dimensional block fault still appears as a block fault in each $k \times k$ 2D plane it touches. Therefore, the results for 2D tori can be applied with suitable modifications.

Our main result in this section is that any fully-adaptive routing algorithm for an n-dimensional tori can be made fault-tolerant by using four additional virtual channels per physical channel.

As in the two-dimensional case, a message is said to be blocked by faults if all of its shortest paths go through one or more fault regions. A message that is blocked for the first time becomes an affected message and remains so for the rest of its journey.

Lemma 2 *A message destined to d is blocked at a node, say x, only if and the addresses of x and d differ in exactly one dimension.*

The proof is similar to that Lemma 1 and is omitted.

From Lemma 2, it is clear that when a message becomes affected by a fault, it needs to travel in only one dimension; however, its journey along this dimension would have been blocked by the faulty rectangular region. Let us consider a message M with destination d which becomes affected at x. M will be referred to as an i-message if it only needs to travel in dimension i when it becomes affected. Further, we say that an affected message is an i^+-message (respectively, i^--message) if $x_i < d_i$ (respectively, $x_i > d_i$). As before, an i^+ message is actually an i^+M or i^+W message; an i^+M message uses the clockwise orientation and i^+W message the counter-clockwise orientation when routed on an f-ring.

Before we consider routing in general n-dimensional tori, we design fault-tolerant fully-adaptive routing algorithms for 3D tori that use four additional virtual channels.

In a 3D torus, there are six types of affected messages ($0^+, 0^-, 1^+, 1^-, 2^+, 2^-$). The planes and virtual channels used to correct in the final dimension and are shown in Table 3. The enhanced fully-adaptive routing algorithm is shown in Figure 7.

Lemma 3 *Assume that the original fully-adaptive routing algorithm F is correct and deadlock- and livelock-free. The procedure Fully-Adaptive-3D correctly routes messages in 3D tori with faulty blocks and does not cause deadlocks or livelocks.*

The proof is similar to that of Theorem 1.

We use the Fully-Adaptive-2D and Fully-Adaptive-3D algorithms to provide fault-tolerant routing in n-dimensional tori. The routing logic is given in Figure 8.

The correctness, deadlock-freedom and livelock-freedom of Fully-Adaptive-nD procedure follow from the corresponding proofs for Fully-Adaptive-2D and Fully-Adaptive-3D procedures.

6 Concluding remarks

We have presented techniques to enhance fully-adaptive wormhole routing algorithms for fault-tolerant routing in tori. In particular, we have shown that four extra virtual channels per physical channel are enough to convert a fully-adaptive wormhole algorithm for fault-tolerant routing.

We have used the block-fault model in which faulty processors and links are in the form of multiple rectangular regions of the network. The concept of fault-rings is used

PROCEDURE FULLY-ADAPTIVE-3D(M) /* Uses a generic fully-adaptive algorithm $\mathcal{F}$ */
/* Uses four additional virtual channels c_0, c_1, c_2, c_3 */

1 Route M using algorithm $\mathcal{F}$ until M either reaches its destination or is affected by faults.

2 If M reached its destination, stop.

3 Determine the i for which M is an i^+-message or an i^--message. /* $i \in \{0, 1, 2\}$ */

4 Depending on the value of i, route M in the plane specified in Table 3 and using the virtual channels indicated.

Figure 7: Fully-adaptive routing in a 3D tori using four additional virtual channels.

Table 3: Virtual Channels Used by Messages in Algorithm f-cube3D

Message Type	Plane Used	Virtual Channel Used
0^+-message	$(0,1)$-plane	c_0 in both dimensions 0 and 1
0^--message	$(0,1)$-plane	c_1 in both dimensions 0 and 1
1^+-message	$(1,2)$-plane	c_2 in both dimensions 1 and 2
1^--message	$(1,2)$-plane	c_3 in both dimensions 1 and 2
2^+-message	$(2,0)$-plane	c_2 in dimension 0 and c_0 in dimension 2
2^--message	$(2,0)$-plane	c_3 in dimension 0 and c_1 in dimension 2

to route around the fault-regions. Each nonfaulty node can determine its position in an f-ring using a distributed algorithm based on exchanging messages with its neighbors. Our algorithms are deadlock- and livelock-free and correctly deliver messages between any pair of nonfaulty nodes in a connected component of the network even in the presence of multiple faulty blocks.

The increase in routing-complexity to achieve fault tolerant wormhole routing is moderate. The status of a message and its type (to indicate its virtual channel class and its direction on f-rings) can be maintained using a few bits in its header. The other overhead is changing the status and setting the type of a message blocked for the first time. This is done just once for each misrouted message. The type of a misrouted message is determined by comparing the addresses of the current host and destination of the message.

To study the performance issues, we have taken a fully-adaptive wormhole algorithm, NHOP, developed originally for routing in fault-free networks, and fortified it with extra virtual channels and the fault-tolerant logic described in this paper. Our simulation results indicate that the NHOP exhibits graceful degradation in performance as the number of faulty components in the network increases.

There are no previous specific results for fault-tolerant routing in tori. Often, the results developed for meshes can be extended to tori with suitable modifications. For example, the dimension reversal algorithm by Dally and Aoki [9], described in the context of meshes, can be applied to tori by considering the hops on wraparound links as dimension reversal hops. It is not clear whether the results by Glass and Ni [14] and Chien and Kim [7], developed specifically for meshes, can be applied to tori in a similar manner. Our experience is that the wraparound links in a torus create more possibilities of deadlocks. If the results developed for meshes are to be applied with few changes, then the number of channels required to avoid deadlocks may have to be doubled. In this paper, we have shown that four extra virtual channels are sufficient for tori for deadlock-free fault-tolerant routing. This result compares well with our earlier result that four extra virtual channels are sufficient for deadlock-free fault-tolerant routing in meshes [5].

The concept of fault rings can be extended to faults of arbitrary shape. In such cases, the fault rings do not have a regular shape. Using the concept of fault rings, one arbitrary shaped fault of any size can be tolerated in a 2D torus. The results presented here appear to be applicable to the case where there are multiple arbitrarily-shaped connected faults and each fault region can be inscribed in a rectangle on the torus (n-dimensional box on an n-dimensional torus) without including faults from other fault regions. We are currently pursuing this topic.

References

[1] A. Agarwal, *et. al.* The MIT Alewife machine: A large-scale distributed multiprocessor. In *Proc. of Workshop on Scalable Shared Memory Multiprocessors.* Kluwer Academic Publishers, 1991.

[2] R. Alverson, D. Callahan, D. Cummings, B. Koblenz, A. Porterfield, and B. Smith. The Tera computer system. In *Proc. 1990 Int. Conf. on Supercomputing.*

[3] K. Bolding and L. Snyder. Overview of fault handling for the chaos router. In *Proceedings of the 1991 IEEE International Workshop on Defect and Fault Tolerance in VLSI Systems*, pages 124–127, 1991.

[4] R. V. Boppana and S. Chalasani. A comparison of adaptive wormhole routing algorithms. In *Proc. 20th Ann. Int. Symp. on Comput. Arch.*, pages 351–360, May 1993.

[5] R. V. Boppana and S. Chalasani. Fault-tolerant wormhole routing algorithms for mesh networks. Submitted for publication, Dec. 1993.

[6] S. Borkar *et al.* iWarp: An integrated solution to high-speed parallel computing. In *Proc. Supercomputing '88*, pages 330–339.

PROCEDURE FULLY-ADAPTIVE-nD(M) /* Uses a generic fully-adaptive algorithm $\mathcal{F}$ */
/* Uses four additional virtual channels c_0, c_1, c_2, c_3 */

1 Route M using algorithm $\mathcal{F}$ until M either reaches its destination or is affected by faults.

2 If M reached its destination,
Stop.

3 Determine the i for which M is an i^+-message or an i^--message.

4 If (n is odd and $i \in \{0, 1, 2\}$),
Route M in the 3D torus formed by dimensions 0, 1, and 2 using Fully-Adaptive-3D algorithm (described above).

Else if ((i is even and n is even) or (i is odd and n is odd)),
Route M in the 2D tori formed by dimensions i and $i+1$ applying the logic of Fully-Adaptive-2D algorithm (Figure 2) with i as 0 and $i+1$ as 1.

Else if ((i is odd and n is even) or (i is even and n is odd)),
Route M in the 2D tori formed by dimensions i and $i-1$ applying the logic of Fully-Adaptive-2D algorithm (Figure 2) with $i-1$ as 0 and i as 1.

Figure 8: Fully-adaptive routing in an nD tori using four additional virtual channels.

[7] A. A. Chien and J. H. Kim. Planar-adaptive routing: Low-cost adaptive networks for multiprocessors. In *Proc. 19th Ann. Int. Symp. on Comput. Arch.*, pages 268–277, 1992.

[8] W. J. Dally. Virtual-channel flow control. *IEEE Trans. on Parallel and Distributed Systems*, 3(2):194–205, Mar. 1992.

[9] W. J. Dally and H. Aoki. Deadlock-free adaptive routing in multicomputer networks using virtual channels. *IEEE Trans. on Parallel and Distributed Systems*, 4(4):466–475, April 1993.

[10] W. J. Dally and C. L. Seitz. Deadlock-free message routing in multiprocessor interconnection networks. *IEEE Trans. on Computers*, C-36(5):547–553, 1987.

[11] J. Duato. A new theory of deadlock-free adaptive routing in wormhole networks. *IEEE Trans. on Parallel and Distributed Systems*, 4(12):1320–1331, Dec. 1993.

[12] S. A. Felperin, L. Gravano, G. D. Pifarré, and J. L. Sanz. Routing techniques for massively parallel communication. *Proceedings of the IEEE*, 79(4):488–503, 1991.

[13] P. T. Gaughan and S. Yalamanchili. Pipelined circuit-switching: A fault-tolerant variant of wormhole routing. In *Proc. Fourth IEEE Symp. on Parallel and Distributed Processing*, pages 148–155, 1992.

[14] C. J. Glass and L. M. Ni. Fault-tolerant wormhole routing in meshes. In *Twenty-Third Annual Int. Symp. on Fault-Tolerant Computing*, pages 240–249, 1993.

[15] I. S. Gopal. Prevention of store-and-forward deadlock in computer networks. *IEEE Trans. on Communications*, COM-33(12):1258–1264, Dec. 1985.

[16] P. Kermani and L. Kleinrock. Virtual Cut-Through: A New Computer Communication Switching Technique. *Computer Networks*, 3:267–286, 1979.

[17] J. H. Kim and A. A. Chien. An evaluation of planar-adaptive routing (PAR). In *Proc. Fourth IEEE Symp. on Parallel and Distributed Processing*, 1992.

[18] S. S. Lam and M. Reiser. Congestion control of store-and-forward networks by input buffer limits—an analysis. *IEEE Trans. on Communications*, com-27(1):127–133, Jan. 1979.

[19] T. Lee and J. Hayes. A fault-tolerant communication scheme for hypercube computers. *IEEE Trans. on Computers*, 41(10):1242–1256, Oct. 1992.

[20] S. L. Lillevik. The Touchstone 30 Gigaflop DELTA prototype. In *Sixth Distributed Memory Computing Conference*, pages 671–677, 1991.

[21] D. H. Linder and J. C. Harden. An adaptive and fault tolerant wormhole routing strategy for k-ary n-cubes. *IEEE Trans. on Computers*, 40(1):2–12, 1991.

[22] M.D. Noakes et al. The J-machine multicomputer: An architectural evaluation. In *Proc. 20th Ann. Int. Symp. on Comput. Arch.*, pages 224–235, May 1993.

[23] A. L. Narasimha Reddy and R. Freitas. Fault tolerance of adaptive routing algorithms in multicomputers. In *Proc. Fourth IEEE Symp. on Parallel and Distributed Processing*, pages 156–161, 1992.

[24] J. Y. Ngai and C. L. Seitz. A framework for adaptive routing in multicomputer networks. In *Proc. First Symp. on Parallel Algorithms and Architectures*, 1989.

[25] W. Oed. The cray research massively parallel processor system, CRAY T3D. Technical report, Cray Research Inc., Nov. 1993.

[26] C. S. Raghavendra, P.-J. Yang, and S.-B. Tien. Free dimensions – an effective approach to achieving fault tolerance in hypercubes. In *Twenty-Second Annual Int. Symp. on Fault-Tolerant Computing*, pages 170–177, 1992.

[27] C. Seitz. Concurrent architectures. In R. Suaya and G. Birtwislte, editors, *VLSI and Parallel Computation*, chapter 1, pages 1–84. Morgan-Kaufman Publishers, Inc., San Mateo, California, 1990.