Adaptive wormhole routing in tori with faults

S. Chalasani
R.V. Boppana

Indexing terms: Adaptive routing, Deadlocks, Fault-tolerant routing, Multicomputer networks, Message routing, Performance evaluation, Torus networks, Wormhole routing

Abstract: The authors present a method to enhance wormhole routing algorithms for deadlock-free fault-tolerant routing in tori. They 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 used 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. Simulation results are presented for a fully-adaptive algorithm showing that good performance can be obtained with as many as 10% links faulty.

1 Introduction

Point-to-point torus and related networks are being used in many recent experimental and commercial multicomputers and multiprocessors [1–3]. 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 [8] has been used in many recent multicomputers [1–4]. 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 predefined order is enforced on the allocation of virtual channels to messages.

For fault-free networks, important issues in the design of a routing algoritlim are high throughput, low-latency message delivery, avoidance of deadlocks, livelocks and starvation and ability to work well under various traffic patterns. 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.

The well-known e-cube or dimension-order routing algorithm is an example of nonadaptive 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.

1.1 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. We consider routing methods that use only local knowledge of faults. We assume that faulty processors are confined to one or more rectangular blocks.

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*. 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.

1.2 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 [5–10]. Reddy and Freitas [11] use global knowledge of faults, spare nodes, and routing tables to investigate the performance limitations caused by faults. Gaughan and Yalamanchili [12] 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 wormhole 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.

IEE Proceedings online no. 19952079

Paper first received 5th January 1995 and in revised form 10th May 1995

S. Chalasani is with the Electrical & Computer Engineering Department, University of Wisconsin-Madison, Madison, WI 53706-1691, USA

R.V. Boppana is with the Computer Science Division, The University of Texas at San Antonio, San Antonio, TX 78249-0664, USA

There are no previous results speciftcally on fault-tolerant wormhole routing in tori. Often, the results developed for meshes [5,8,13] can be extended to tori with suitable modiftcations, 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, the number of virtual channels required to avoid deadlocks may be doubled [5]. Furthermore, meshes have edges, for example, the top row in a 2D mesh, and faults on edges are complicated to handle [13]. But this case never arises in tori, since they are node symmetric. Hence, extending efficient mesh routing techniques to tori in a straight forward manner may not necessarily yield efficient routing algorithms for the latter networks.

In terms of adaptivity and performance comparisons, the results by Dally and Aoki [8] are the most relevant to ours. With the dimension-reversal schemes of Dally and Aoki, a message may lose its adaptivity, if its number of dimension reversals equals the number of virtual channel classes. 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 number of virtual channels and the location of faults. In contrast, our algorithms can tolerate any number and combination of rectangular faulty blocks with simple logic, and require only four virtual channels more than that required for the original adaptive algorithm. (Throughout this paper we indicate the number of virtual channels on per physical channel basis). This result compares well with our earlier result that four extra virtual channels are sufficient for routing in meshes with faults [13].

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 neighbours of the node $x = (x_{n-1}, ..., x_0)$ in dimension i are $(x_{n-1}, ..., x_{i+1}, x_i \pm 1, x_{i-1} ..., x_0)$, where addition and subtraction are modulo k. A link is said to be a wraparound link if it connects two neighbours 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 represents two unidirectional physical communication channels. The link between nodes x and y is denoted by $<x, y>$. To simplify presentation, we discuss the concept of fault-rings for two dimensional (2D) tori. We label the sides of a 2D torus as North, South, East and West.

2.1 Fault model

We consider both node and link faults. All the links incident on a faulty node are considered faulty. We assume that faults are permanent and nonmalicious faults. Therefore, messages are generated by and for nonfaulty processors only. We develop fault-tolerant algorithms, for which it is sufficient if each nonfaulty 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 Fig. 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[†] includes all faulty components in F and (*c*) the interior of the rectangle contains no component that is not present in F.

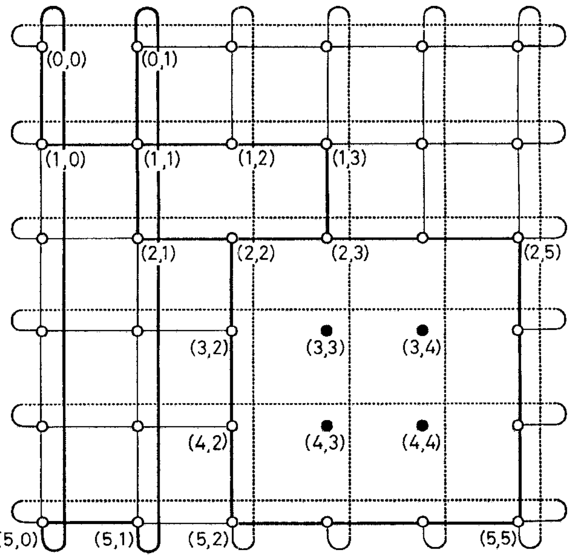

Fig. 1 *Three fault regions and their associated fault rings in a 6 × 6 torus*

For example, the set $\{(3,3),(3,4),(4,3),(4,4)\}$ of faulty nodes shown in Fig. 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 $\{<(1,1),(1,2)>, <(1,2),(2,2)>, <(2,2),(2,1)>, <(2,1),(1,1)>\}$ 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 Fig. 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.

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 Fig. 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 assume that faults do not disconnect the network.

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 modelled as a block-fault, albeit by labelling some nonfaulty processors and/or links as

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

faulty [5]. Finally, the block fault model accurately models faults at the chip, multichip module, and board levels.

2.2 Fault rings

Conceptually, fault regions may be considered as islands of faults in a sea of communication channels and nodes. In the same manner as 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 (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 Fig. 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 Fig. 1. The *f*-ring associated with the link fault region {<(1,2),(2,2)>} has nodes $\{(i,j) \mid 1 \le i \le 2, 1 \le j \le 3\}$ 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 Fig. 1. A set of fault rings are said to overlap if they share one or more links.

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 neighbours in other dimensions. Based on the set of messages received, each node that is to be on the *f*-ring determines its neighbours in the *f*-ring. For more details, the reader is referred to [13].

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. Our approach is to use a known adaptive algorithm as much as possible to route messages. When a message is blocked[‡] by a fault and the adaptive algorithm cannot handle it, the additional routing logic described here will be used to route the message around the fault.

The fully-adaptive algorithms we consider have the following property: if a message, upon arriving at an intermediate node, cannot find an idle channel for its next hop, then it can choose any one of the channels allowed by the routing logic and wait for that channel indefinitely without causing deadlocks. The adaptive algorithms based on Duato's theory [6] do not satisfy this, since they require the message in the above example to rechoose a channel, to avoid deadlocks, after waiting for a finite amount of time.

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

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

[‡] 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.

Table 1: Virtual channels and *F*-ring orientations used by affected messages

message type	conditions satisfied	virtual channel	*F*-ring orientation
0^+M	$d_1 = x_1$ and $d_0 > x_0$ and $(d_0 - x_0) \le \lfloor k/2 \rfloor$	c_0	clockwise
0^+W	$d_1 = x_1$ and $d_0 > x_0$ and $(d_0 - x_0) > \lfloor k/2 \rfloor$	c_0	counter-clockwise
0^-M	$d_1 = x_1$ and $x_0 > d_0$ and $(x_0 - d_0) \le \lfloor k/2 \rfloor$	c_1	clockwise
0^-W	$d_1 = x_1$ and $x_0 > d_0$ and $(x_0 - d_0) > \lfloor k/2 \rfloor$	c_1	counter-clockwise
1^+M	$(d_1 - x_1) \le \lfloor k/2 \rfloor$ and $d_1 > x_1$ and $d_0 = x_0$	c_2	clockwise
1^+W	$(d_1 - x_1) > \lfloor k/2 \rfloor$ and $d_1 > x_1$ and $d_0 = x_0$	c_2	counter-clockwise
1^-M	$(x_1 - d_1) \le \lfloor k/2 \rfloor$ and $d_1 < x_1$ and $d_0 = x_0$	c_3	clockwise
1^-W	$(x_1 - d_1) > \lfloor k/2 \rfloor$ and $d_1 < x_1$ and $d_0 = x_0$	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 vitual chanels 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 dimension *d* that takes the message closer to its destination.

Case 1: If the current host and destination match in dimension *d′*, and the legal hop is available, it is taken (see Fig. 3).

Case 2: If *M* is a message on an *f*-ring, it is 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 Fig. 3).

Fig. 2 *Fault-tolerant routing logic for 2D tori*

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 they 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 the 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.

3.1 Routing affected messages

The fault-tolerant version of a generic fully-adaptive algorithm *F*, denoted F_f, uses four virtual channels, – c_0, c_1, c_2 and c_3 – in addition to those used by *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 are used exclusively by 0^-, 1^+, 1^- messages, respectively. Rules to route various messages are specified in procedure 'Fully-Adaptive-2D' (Fig. 2).

Example: In Fig. 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 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 labelled 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.

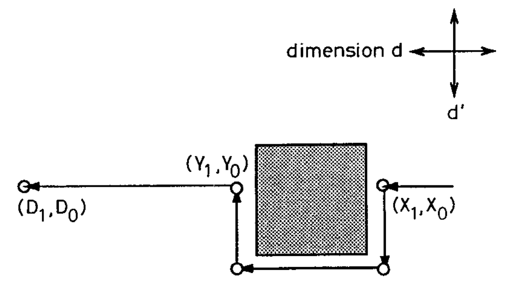

Fig. 3 *Routing of the message from (X_1, X_0) to (Y_1, Y_0) is done using Case 2 of Rule 2 of procedure 'Fully-Adaptive-2D'; Case 1 is used thereafter*

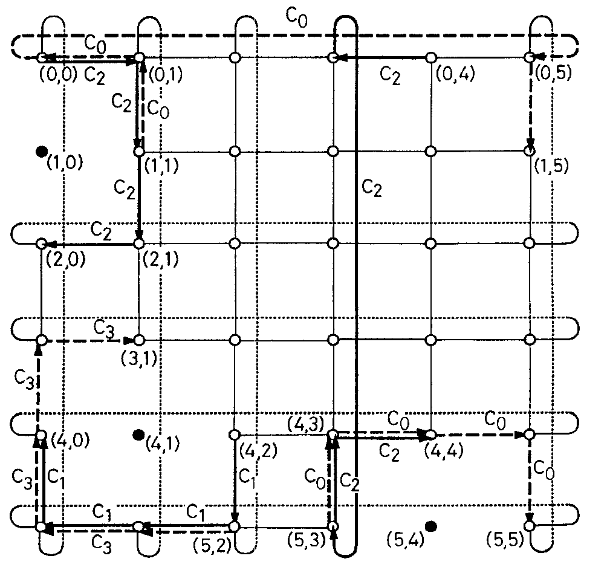

Fig. 4 *There are three f-rings corresponding to the three faulty nodes (1,0), (4,1) and (5,4). Various messages in this network and how they are routed is indicated in Table 2*

Table 2: A few messages in the faulty network of Fig. 4

source	destination	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

Theorem 1: Assume that the fully-adaptive routing algorithm F correctly routes messages and is deadlock free. The fault-tolerant fully-adaptive routing algorithm F_f described by Rules 1 and 2 (in Fig. 2) is deadlock-free in the presence of multiple rectangular fault blocks and delivers messages correctly between any pair of nonfaulty nodes.

Proof. Algorithm F_f correctly routes unaffected messages between any pair of nonfaulty nodes, since 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 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 virtiial 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 channels in the East columns of f-rings. Its underlying graph is acyclic. Similarly, 0^+W uses an acyclic network of c_0 channels. Therefore, a deadlock among 0^+-messages involves both 0^+M and 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.

3.2 *Fault-tolerant routing with overlapping f-rings*

Thus far, we have assumed that faults are such that the f-rings 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.

(a) 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. Except for the use of additional virtual channels, the routing logic remains the same as in the case of nonoverlapping f-rings.

(b) 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 labelled as a 0^+M message when it is blocked for the first time and routed as a 0^+M message until it reaches its destination. That is, 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 tvpe uses a distinct class of virtual channels, the routing is deadlock free.

4 Simulation results

To study the performance issues we have developed a flit-level 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 [14]. The NHop provides minimal, fully-adaptive routing in fault-free tori using $\lfloor n\lceil k/2 \rceil/2\rfloor + 1$ virtual channels.

The negative hop wormhole algorithm is discussed in [14]. To use the negative hop algorithm, the network is coloured, and each node is given a label corresponding to its colour. 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. We use bisection utilisation and average message latency as the performance metrics. The bisection utilisation (ρ_b) is defined as

$$(\text{no. of messages across the bisection/cycle}) \times \frac{\text{message length}}{\text{bisection BW}}$$

The bisection bandwidth (BW) is deftned 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 utilisation and channel utilisation 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.

The fault-tolerant Nhop requires 13 virtual channel classes: nine for fault-free routing and four for fault-tolerant routing. We have used 16 virtual lanes per physical channel. we distributed 12 lanes among the nine normal, fault-free classes and allocated one lane to each of the four special, fault-tolerant classes. The three additional lanes improve the performance of the NHop.

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 inter-arrival time; the motivation for injection control is due to Lam [15]. After some experimentation, we have chosen an injection limit of four 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 injectlon limit reduces throughputs around the saturation slightly. For the value 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. Specifically, for the 1% case, we have set, randomly, a node and link faulty; since four links are incident on a node, five of the 512 links in the network are faulty. For the 5% fault case, we have set four 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 also simulated the routing algorithm on a fault free torus.

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 counterclockwise, 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: (*a*) a message that has taken *i* negative hops can use virtual channels in any of classes 0, ..., *i*; (*b*) 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 Fig. 5. For the fault-free network, the NHop has a peak utilisation 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 utilisation ranges from 0.648 to 0.735.

4.2 Peak performance

Comparative performance across different fault cases in Fig. 5 is specific to the fault sets used. Therefore, we have further simulated the NHop for 1, 5 and 10% 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 Figs. 6 and 7.

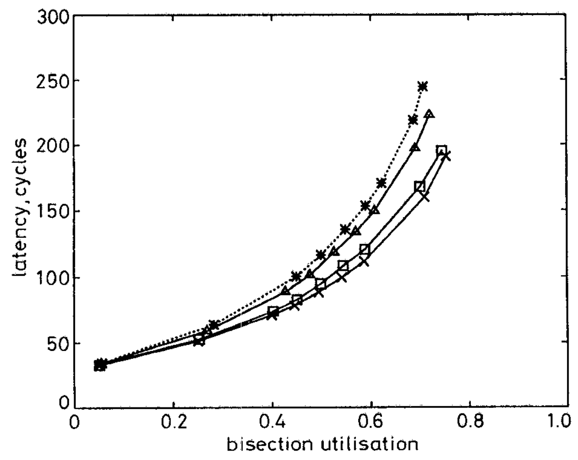

Fig. 5 *Performance of the NHop algorithm for uniform traffic in a 16 × 16 torus with various faults. The label dp indicates the results for d% faults*
-×- 0p
-□- 1p
-Δ- 5p
··×·· 10p

As the number of faults is increased, the latency increases steadily and the utilisation 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.

The fault-tolerant version of NHop exhibits a similar graceful degradation in performance in meshes with faults. We performed additional simulations in which the NHop for mesh exhibited a 20% drop in throughput from the fault-free case to the case with 10% faults. Dally and Aoki [8] indicate that the dynamic dimension reversal algorithm exhibits a similar graceful degradation of throughput for meshes.

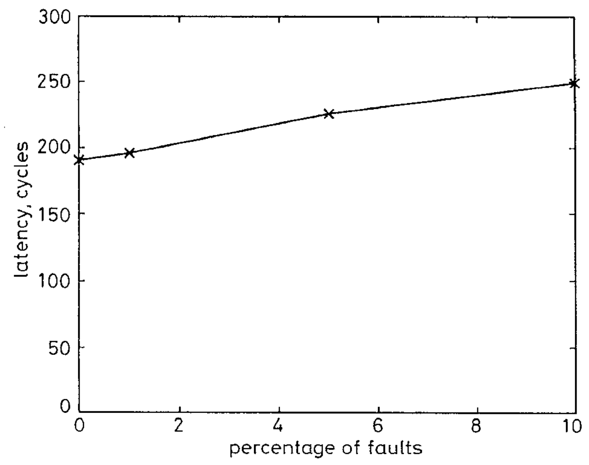

Fig. 6 *Latencies for different fault cases*

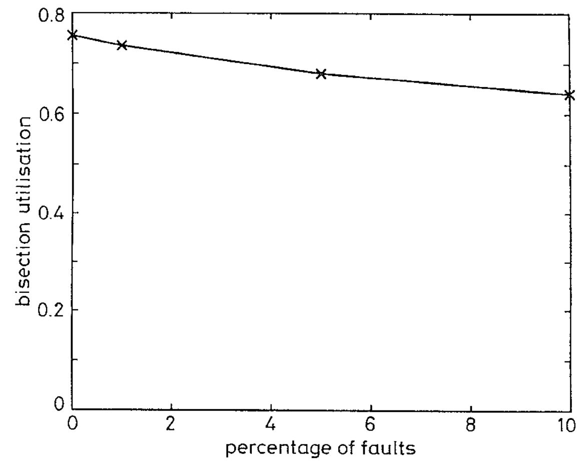

Fig. 7 *Maximum utilisation for different fault cases*

5 Fault-tolerant routing in multidimensional tori

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

5.1 Fault model

To define the fault model precisely, we consider n-tuples of k symbols $\{0,..., k - 1\}$; we refer to this set as Q and these correspond to nodes in a k-ary n-dimensional torus.

A node $x = (x_{n-1},...,x_0)$ is said to be a *base-node* with respect to another node $y = (y_{n-1},...,y_0)$ if and only if $x_i \le y_i$ for all i; if x is a base node for y, y is said to be an *apex-node* for x. A block B_{xy} with base-node x and apex-node y contains all the nodes in the set

$$N_{xy} = \{(z_{n-1}, \dots, z_0) \mid x_i \le z_i \le y_i,\ 0 \le i \le (n-1)\}$$

and every link that connects any two nodes in N_{xy}. A node $z = (z_{n-1},..., z_0)$ is said to be a *boundary node* of block B_{xy} if $z_i = x_i$ or $z_i = y_i$ for some i. A link $<z,z'>$ is a *boundary link* iff both z and z' are boundary nodes. The interior of B_{xy} contains all nodes and links that are not on the boundary of B_{xy}.

A translation T_x of (k, n)-torus with respect to node x is defined as relabeling every node $y = (y_{n-1},...,y_0)$ in (k,n)-torus with $y = ((k+y_{n-1}-x_{n-1}) \bmod k,...,(k+y_i-x_i) \bmod k,...,(k+ y_0-x_0) \bmod k)$. A set F of faulty nodes and links in a (k, n)-torus is said to be a *faulty-block* iff there exist a node z, base-node $x \in T_z(Q)$ and an apex-node $y \in T_z(Q)$ such that:

(i) The interior of $B_{xy} \in T_z(Q)$ contains all and only the components of F.

(ii) No boundary node or link of $B_{xy} \in T_z(Q)$ is faulty.

(iii) The faults in each $k \times k$ subtorus of $T_z(Q)$ satisfy the 2D block fault model described in Section 2.

A set F of faulty nodes and links in a k-ary n-dimensional 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. The following observation forms the basis for our fault-tolerant routing algorithms for n-dimensional tori.

Observation 1: Let F be a block fault in (k, n)-torus and let $F_1,...,F_r$ be the corresponding faulty blocks. If we consider any $k \times k$ subtori H of the (k, n) torus, the faulty nodes of F_i, $1 \le i \le r$ form a rectangular fault region (defined in Section 2.1) in H.

Observation 1 allows us to route in a k-ary n-dimensional torus under block-faults by routing a message in 2D tori each with faulty blocks of rectangular shapes. We use this idea to develop the algorithms presented in this Section.

5.2 Design of fault-tolerant fully-adaptive routing algorithms

Our main result in this Section is that any tully-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.

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

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.

Proof: First, we note that the message will be blocked at a node x only if x is on the fault ring associated with the fault block. Further, due to the block-fault model, x can have faulty links in at most one dimension. Assume, to the contrary, that the message becomes blocked at x and that x and d differ in more than one dimension. Since the outgoing links of x are faulty in only one dimension, the message can take a hop in a dimension other than that of the faulty link(s). This contradicts our assumption that the message is blocked at x. This proves the lemma. ■

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

5.2.1 Fault-tolerant fully-adaptive routing in 3D tori:

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 are shown in Table 3. The enhanced fully-adaptive routing algorithm is shown in Fig. 8.

Lemma 3: Assume that the originalfully-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.

Proof: From observation 1, it is clear that in any 2D subtori of a 3D tori with faulty blocks, the f-regions are of rectangular shape. Further, the fault model assumes that no complete row or column of faults can be faulty in any 2D subtori of the network. Thus, $0^{\pm}$ messages use the appropriate links in dimension 0 and the (0,1)-plane around f-rings to reach their destinations. Similarly, $1^{\pm}$ messages travel in dimension 1 using (1,2) plane to get around faults; $2^{\pm}$ messages travel in dimension 2 using (2,0) plane to get around faults. Since our fault model satisfies the constraint that each 2D plane of the 3D torus has only block-faults, affected messages are correctly delivered to their destinations.

The routing in each 2D plane is livelock-free, since a message visits each f-ring at most once and its journey along an f-ring is bounded by a finite number of hops (see proof of theorem 1).

To prove deadlock freedom, first observe that routing in each plane is deadlock-free, since the i^+ and i^- messages use distinct virtual channels in each plane.

```
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.
```

Fig. 8 *Fully-adaptive routing in 3D tori using four additional virtual channels*

```
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 tori formed by dimensions i and i + 1 applying the logic of Fully-Adaptive-2D
    algorithm (Fig. 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 tori formed by dimensions i and i – 1 applying the logic of Fully-Adaptive-2D
    algorithm (Fig. 2) with i – 1 as 0 and i as 1.
```

Fig. 9 *Fully-adaptive routing in nD tori using four additional virtual channels*

Further, the virtual channels used by 0-messages (c_0 and c_1) and 1-messages (c_2 and c_3) are disjoint. Virtual channels used by 0-messages and 2-messages are also disjoint, since 0-messages use c_0 and c_1 in dimension 0 and 2-messages use c_2 and c_3 in dimension 0 (from Table 3). A similar statement holds for 1-messages and 2-messages. Hence the theorem. ■

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 Fig. 9. 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. Though the techniques developed for meshes may be extended with suitable modifications to tori, such extensions usually double the number of channels used for mesh networks. By separating fault-tolerant routing from normal adaptive routing, which is the approach used in this paper, this effect can be limited to only the resources used 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 to route messages around the fault-regions. 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. Our fault-tolerant technique facilitates modular implementation. The switching among the extra virtual channels can be implemented by a 4×4 crossbar independent of the crossbar used for the original adaptive routing.

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.

The proposed techniques seem to be applicable to a wider class of adaptive algorithms, more complex fault shapes, and different network topologies. More work on networks other than tori is needed, however. Since the NHop algorithm is applicable to any network topology the fault-tolerant routing based on NHop and fault-rings may yield a simple and efficient routing method for many networks of interest.

7 Acknowledgments

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

8 References

1 AGARWAL,A.: 'The MIT Alewife machine: a large-scale distributed multiprocessor', Proceedings of workshop on *Scalable shared memory multiprocessors* (Kluwer Academic, 1991)

2 NOAKES,M.D.: 'The J-machine multicomputer: an architectural evaluation', Proceedings 20th annual international symposium on *Computer architecture*, May 1993, pp. 224–235

3 CRAY RESEARCH INC.: 'Cray T3D Aarchitectural summary', Oct. 1993

4 LILLEVIK,S.L.: 'The touchstone 30 gigaflop DELTA prototype', 6th Distributed memory computing conference, 1991, pp. 671–677

5 CHIEN,A.A., and KIM,J.H.: 'Planar-adaptive routing: low-cost adaptive networks for multiprocessors', Proceedings of the 19th annual international symposium on *Computer architecture*, 1992, pp. 268–272

6 DUATO,J.: 'New theory of deadlock-free adaptive routing in wormhole networks', *IEEE Trans. Parallel & Distrib. Syst.* Dec. 1993, **4**, (12), pp. 1320–1331

7 LINDER,D.H., and HARDEN,J.C.: 'An adaptive and fault tolerant wormhole routing strategy for -ary n-cubes', *IEEE Trans. Comput.*, 1991, **40**, (1), pp. 2–12

8 DALLY,W.J., and AOKI,H.: 'Deadlock-free adaptive routing in multicomputer networks using virtual channels', *IEEE Trans. Parallel & Distrib. Syst.* Apr. 1993, **4**, (4), pp. 466–475

9 GLASS,C.J. and NI,L.M.: 'Fault-tolerant wormhole routing in meshes', 23rd annual international symposium on *Fault-tolerant computing*, 1993, pp. 240–249

10 BOLDING,K., and SNYDER,L.: 'Overview of fault handling for the chaos router', Proceedings of the 1991 IEEE International workshop on *Defect and fault tolerance in VLSI systems*, 1991, pp. 124–127

11 NARASIMHA-REDDY,A.L., and FREITAS,R.: 'Fault tolerance of adaptive routing algorithms in multicomputers', Proceedings of the 4th IEEE symposium on *Parallel and distributed processing*, 1992, pp. 156–161

12 GAUGHAN,P.T., and YALAMANCHILI,S.: 'Pipelined circuit-switching: a fault-tolerant variant of wormhole routing', Proceedings of the 4th IEEE symposium on *Parallel and distributed processing*, 1992, pp. 148–155

13 BOPPANA,R.V., and CHALASANI,S.: 'Fault-tolerant wormhole routing algorithms for mesh networks', *IEEE Trans. Comput.*, 1995, **44**, (7), pp. 848–864

14 BOPPANA,R.V., and CHALASANI,S.: 'A comparison of adaptive wormhole routing algorithms', Proceedings of the 20th annual international symposium on *Computer architecture*, May 1993, pp. 351–360

15 LAM,S.S., and REISER,M.: 'Congestion control of store-and-forward networks by input buffer limits—an analysis', *IEEE Trans. Commun.* Jan. 1979, **27**, (1), pp. 127–133