Fault-Tolerant Multicast Communication in Multicomputers[*]

Rajendra V. Boppana
Div. of Computer Science
The Univ. of Texas at San Antonio
San Antonio, TX 78249-0664
boppana@runner.utsa.edu

Suresh Chalasani
ECE Department
University of Wisconsin
Madison, WI 53706-1691
suresh@ece.wisc.edu

Abstract. *We describe fault-tolerant routing of multicast messages in mesh-based wormhole-switched multicomputers. With the proposed techniques, multiple convex faults can be tolerated. The fault information is kept locally—each fault-free processor needs to know the status of the links incident on it only. Furthermore, the proposed techniques are deadlock- and livelock-free and guarantee delivery of messages. In particular, we show that the previously proposed column-path and Hamilton-path based algorithms can be made tolerant to multiple faults using two or three virtual channels per physical channel.*

Keywords: block faults, fault-tolerant routing, Hamilton path routing, multicast routing, wormhole routing.

1 Introduction

Many commercially available parallel computers use mesh or grid based networks for interprocessor communication with a processor and router module at each node [5, 10]. The interprocessor communication functions in a multicomputer are usually handled by a router which receives data from incoming channels and transmits data to outgoing channels using a suitable channel assignment protocol. The channel assignment is specified as a routing algorithm and is implemented in distributed manner such that each router routes messages from its input channels to its output channels based on its local information only.

Wormhole switching [7], a form of pipelined communication, is the most commonly used switching technique in multicomputers; store-and-forward and virtual cut-through are alternatives to wormhole switching.

Several methods are available for unicast routing, where each message is between a pair of nodes. An excellent survey of wormhole routing methods can be found in [12]. The issue of routing becomes complicated when there are faults in the parallel computer or when multicast communication should be supported. In multicast communication, a node sends a message to multiple destinations. Multicast communication is complicated, since a multicast message, in general, requires more resources than a unicast message.

In this paper, we study the problem of fault-tolerant multicast communication for wormhole switched multicomputers. This is an important problem, since multicast communication is a natural communication primitive to handle synchronizations, invalidations and updates of cache lines in distributed shared memory computers, and since parallel computers must be used efficiently even in the presence of faults.

There are some recent results on fault-tolerant wormhole routing [6, 4, 9, 8, 1, 2] and multicast wormhole routing [13, 11, 3], but very few results exist on fault-tolerant multicast routing (for a result on hypercubes with limited number of faults see [14]).

In this paper, we show how to provide fault-tolerant communication using two recently proposed multicast routing algorithms for mesh based multicomputers. We consider the convex or block fault model used in literature [4, 1] with no global knowledge of fault information. If the network is connected, our techniques provide deadlock- and livelock-free delivery of messages to all of their destinations.

Section 2 describes the key multicast algorithms used in this paper and the fault model. Section 3 describes fault-tolerant routing with the column path algorithm. Section 4 describes fault-tolerant routing with Hamilton path based routing algorithm. Section 5 concludes the paper.

2 Fault model and routing algorithms

We consider k-radix, 2-dimensional meshes. But all the results and discussions can be applied to multidimensional tori and meshes with suitable modifications.

The two dimensions of the mesh are denoted as DIM_1 and DIM_0. The rows of a 2D mesh are numbered from top to bottom $0, 1, \ldots, k-1$, and the columns are numbered from left to right $0, 1, \ldots, k-1$. Node x, $0 \le x < k^2$, in a 2D mesh may be represented by a two-tuple (x_1, x_0), where x_1 is the node's row number and x_0 the node's column number in the 2D grid. The hops taken by a message in a row correspond to hops through processors in DIM_0 and hops in a column correspond to hops in DIM_1. In addition, a hop may be a '+' or a '−' hop depending on the indices of the current node and the next node in the dimension of travel. For example, DIM_{1+} hops correspond to column hops from top to bottom. A communication channel from node x to y is denoted by $< x, y >$. Each node is a combination of processor, local memory, and router. The router handles all the communication and is connected to its processor through injection and consumption (delivery) channels, and connected to other nodes (routers)

[*]Rajendra Boppana's research has been supported by NSF Grant CCR-9208784. 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.

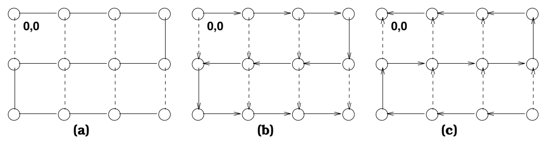

Figure 1: Example of (a) an undirected Hamilton path and the corresponding (b) H_u and (c) H_l directed networks of a mesh. The solid lines indicates the Hamilton path and dashed lines indicate the links that could be used to reduce path lengths in message routing.

through communication links. Each communication link is a full-duplex channel implemented using two unidirectional physical channels.

2.1 Multicast routing algorithms

First we describe a few recent multicast algorithms: the Hamilton path based algorithms [11] and the e-cube based *column path* algorithm [3].

2.1.1 Multicast routing based on Hamilton paths

First an undirected Hamilton path, which goes through each node exactly once, is constructed. An example of an undirected Hamilton path, with node $(0,0)$ as an end node, is given in Figure 1. From this two directed Hamilton paths can be constructed: one starts at $(0,0)$, the H_u network, and another ends at $(0,0)$, the H_l network. The links that are not part of the Hamilton path may be used with appropriate direction to reduce path length.

Dual-path algorithm. Due to the construction of the Hamilton paths H_u and H_l, the paths from any node to any other node are acyclic. In particular, some nodes are reached from a given node via H_u network only and the rest via H_l network only.

In the dual-path algorithm, multicast messages from a node are transmitted on appropriate parts of the H_u and H_l networks. Figure 2.1.1(a) illustrates the portions of H_u and H_l networks used by node $(3,2)$ to send its multicast messages. Hence, the destinations of a multicast message are placed into two groups. One group has all the destinations that can be reached from the source node using the H_u network, and the other has the remaining destinations, which can be reached using the H_l network.

Thus each source of a multicast message, depending on its location and the locations of the destinations, transmits either one or two copies of the message. For example, if $(3,2)$ needs to send a message to destinations $(0,5)$ and $(5,0)$, it will send two copies in opposite directions—one to $(0,5)$ and another to $(5,0)$. However, a multicast message from $(3,2)$ to destinations $(5,5)$ and $(5,0)$ will be sent as a single message. For shorter paths, vertical channels that are not part of the Hamilton path may be used appropriately. The routing of a multicast message from $(3,2)$ to (0,5), (1,4), (5,0), and (5,5) is indicated in Figure 2.1.1(a).

Multipath algorithm. The dual-path algorithm uses at most two copies of the message for multicast communication. This may increase the latency for some multicast messages. The multipath algorithm attempts to reduce long latencies by using up to four copies ($2n$ for n-dimensional meshes) of the original multicast message. As per the multipath routing algorithm, all the destinations of the multicast message are grouped into four disjoint subsets. Each subset of destinations are serviced by one copy of the multicast message [11]. Figure 2.1.1(b) indicates the routing of a multicast message from (3,2) to (0,5), (1,4), (5,0), and (5,5) using three copies.

The dual-path and multipath schemes provide deadlock-free routing of multicast messages. Further, they also provide minimal routing of unicast messages, since vertical links are used for shortcuts. Therefore, either scheme can be used to route unicast and multicast messages simultaneously in a common framework.

2.1.2 Column-path routing algorithm

The dual-path and multipath schemes are not compatible with the well-known e-cube routing algorithm. Since the e-cube is the most commonly used routing method, it is of interest to develop multicast techniques that can take advantage of implementation techniques and methods developed for e-cube. One such example is the column-path algorithm given in [3]. The e-cube routing is specified for unicast messages as follows: each message is routed in DIM_0 until it exhausts all its row hops, at which point it is in the same column as its destination; the message is then routed in DIM_1 until the destination is reached.

The column-path algorithm partitions the set of destinations of a multicast message into at most $2k$ subsets such that there are at most 2 messages directed to each column. Only one message is sent to a column if all destinations in that column are either below or above the source node; otherwise, two messages are sent to that column. In the example of Figure 2.1.1(c), the destinations for the multicast message with source $(2,2)$ are $(1,4)$, $(3,3)$, $(3,4)$, $(4,4)$, $(1,5)$ and $(0,5)$. In all four copies of the message are sent; one copy to $(3,3)$, one to $(1,4)$, another to $(3,4)$ and $(4,4)$, and yet another to $(1,5)$ and $(0,5)$. Each of these messages is routed using the e-cube (or, row-column) routing algorithm. Hence, the column-path routing is compatible with the unicast routing method used in the current parallel computers. A similar but more general method has been independently developed by Panda *et al.* [13].

2.2 The fault model

We consider both node and link faults. All the links incident on a faulty node are considered faulty. We assume that failed components simply cease to work and that messages are generated among nonfaulty processors only.

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. We assume that each non-faulty processor knows only the status of its neighbors.

A set F of faulty nodes and links indicates a (rectangular) fault block, or f-region, if there is a rectangle connecting various nodes of the mesh such that (a) the boundary of the rectangle has only fault-free nodes and links and (b) the interior of the rectangle contains all and only the components given by F. A fault set that includes a component from one of the four boundaries—top and

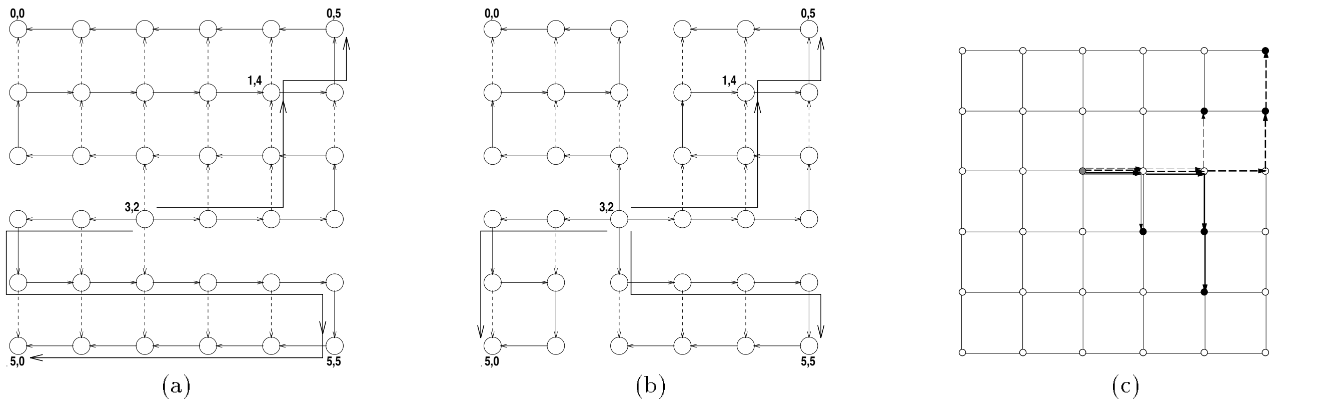

Figure 2: Examples of routing using (a) dual-path (b) multipath (c) column-path algorithms.

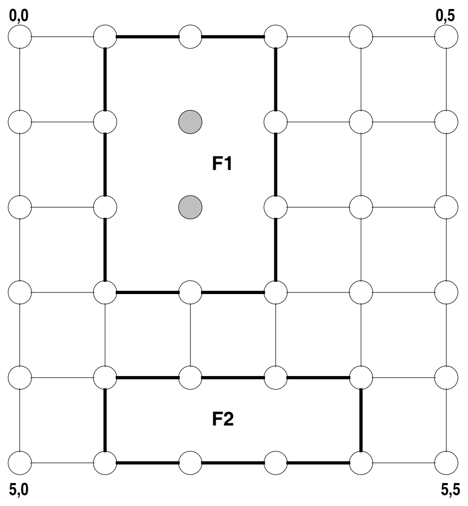

Figure 3: Block faults in a 2-D mesh. Faulty nodes are shown as filled circles, and faulty links are not shown.

bottom rows, left most and right most columns—of a 2D mesh denotes a rectangular fault block, if the above definition is satisfied when the mesh is extended with nonfaulty *virtual* rows and columns on all four sides. However, fault blocks abutting on network boundary are not convex [4]. Therefore, we do not consider faults on network edges. Figure 3 indicates two fault blocks: $F_1 = \{(1,2),(2,2)\}$ and $F_2 = \{<(4,2),(5,2)>,<(4,3),(5,3)>\}$.

We use the *block-fault* model, in which each fault belongs to exactly one fault block. Under the block-fault model, the complete set of faults in a 2D mesh is the union of multiple fault blocks (e.g., $F_1 \cup F_2$ in Figure 3). If faults disconnect the network, our results can still be applied to each subnetwork.

For each fault region, there is a ring of fault-free nodes and links such that it encloses the fault-free region. Such a ring with minimal number of links is called the fault-ring (f-ring) for that fault region. The f-ring of a block fault is rectangular. The f-rings associated with the two fault blocks in Figure 3 are indicated by thick lines. The four sides of an f-ring are classified into left and right columns and and top and bottom rows. If two f-rings have common physical channels, then they are said to overlap.

When a fault occurs, the corresponding f-ring 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. Using the set of messages received, each node determines its position and neighbors on the f-ring. There are eight possible positions for a processor to be in an f-ring: four corner positions, two row positions and two column positions. For more details on forming f-rings, see [2].

An f-ring represents a two-lane path to a message that needs to go through the f-region contained by the f-ring. Routing messages on fault-rings creates new dependencies among resources acquired by messages and, hence, additional possibilities for deadlocks.

3 Fault-tolerant column-path routing

Because of its simple routing logic, the column-path algorithm can be easily enhanced to handle faults. When there are no faults, the original column-path algorithm is used. Even when there are faults in the network, each message is routed using the original column-path algorithm as much as possible. When a message arrives at a node, the next hop for that message is specified by the column-path algorithm. If that hop is on a faulty link, then the message is blocked by the fault. The routing logic is enhanced to handle such situations so that the message is routed around faults. Once the message is routed around faults, the column-path algorithm is used to route the message until it reaches all of its destinations or is blocked again. It is noteworthy that the modifications to routing are used only when a message is blocked by a fault. First, we consider nonoverlapping f-rings.

Modifications to the routing logic. The path of a message in column-path algorithm consists of two parts: the first part is on row (DIM_0) channels and the second part is on column (DIM_1) channels. Therefore, a message may be blocked by a fault while traveling in a row or in a column. A message blocked from taking its DIM_0 hop travels on two sides of the f-ring; a message blocked from taking its DIM_1 hop travels on three sides of the f-ring.

If a message is blocked from taking a DIM_{0+} hop, then it touches the corresponding f-ring on the left column of the f-ring. This message travels on the the f-ring in clockwise orientation (up and right) if its first destination is in a row below its current row; otherwise it travels the f-ring in counter clockwise orientation (down and right). Similarly, if a message is blocked from taking its DIM$_0$− hop, then it travels on the f-ring in clockwise orientation if its first destination is in a row above the current row or counter clockwise orientation otherwise (see Figure 4). A DIM$_0$ message that is in the same column as its first destination becomes a DIM$_1$ message.

If a message is blocked from taking a DIM$_1$+ hop, then it is blocked at a node in the top row of the f-ring. This message travels on three sides of the f-ring in clockwise orientation such that it reaches the same column at the bottom row of the f-ring. Finally, if a message is blocked from taking a DIM$_1$− hop, then it travels on the counter clockwise orientation on the f-ring starting from the bottom row of the f-ring to the top row of the f-ring such that it is in the same column as it was before being blocked by the fault. See Figure 4 for an illustration.

Because of this misrouting of messages when blocked by faults, some physical channels around f-rings are used by multiple messages. This creates cyclic dependencies. To break these new dependencies, we use a general technique given in [7] and simulate multiple virtual channels on each physical channel. The bandwidth of a physical channel is demand time-multiplexed among the virtual channels. When faults are such that only nonoverlapping f-rings occur, just two virtual channels per physical channel are sufficient to provide deadlock free routing even when there are multiple faults in the network.

We use two classes of virtual channels: c_0 and c_1. On each physical channel, a virtual channel of each class is simulated. The channel allocation is such that any new dependencies among the four classes of messages caused by sharing of the physical channels on f-rings are broken. For DIM$_0$ messages, virtual channels of class c_0 are used and for DIM$_1$ messages c_1 virtual channels are used.

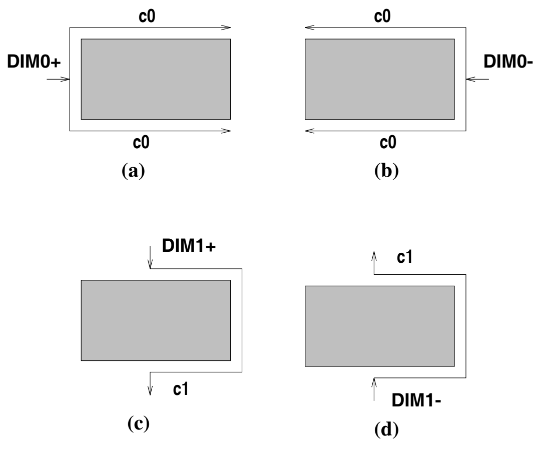

Figure 4: Routing of four different message types around a fault. The shaded area represents a faulty block, and directed lines indicate the paths of messages on the f-ring.

The complete routing logic and channel allocation are given in Figure 5. An example of fault-tolerant routing with the proposed method is shown in Figure 6.

Table 1: Orientations and virtual channels used by the Fault-Tolerant-Column-Path-Routing algorithm.

Message Type	Position of Next Destination	F-Ring Orientation (Virtual Channel)
DIM$_0$+	In a row above its row of travel	Counter Clockwise (c_0)
DIM$_0$+	In a row below its row of travel	Clockwise (c_0)
DIM$_0$−	In a row above its row of travel	Clockwise (c_0)
DIM$_0$−	In a row below its row of travel	Counter Clockwise (c_0)
DIM$_1$+	(don't care)	Clockwise (c_1)
DIM$_1$−	(don't care)	Counter Clockwise (c_1)

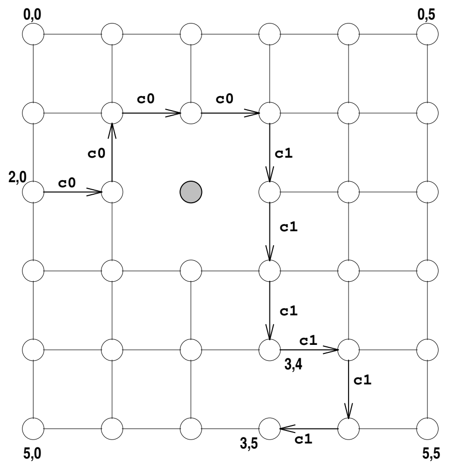

Figure 6: Example of fault-tolerant routing with the column-path algorithm. There is one faulty (shaded) node and one faulty link. The path of the multicast message from (2,0) to (3,4) and (3,5) is shown by directed lines.

Proof of deadlock free routing. In multicast wormhole routing, deadlocks can arise from dependencies on communication channels between nodes and consumption channels between a router and its processor in a node [3]. First let us consider deadlocks on communication channels. The proof technique is similar to the one we have given in [1] for fault-tolerant *e*-cube routing, since the column-path algorithm is similar to the *e*-cube routing. Therefore, a sketch of the proof is given below.

For the deadlock to occur, there has to be a cyclic dependency on the virtual channels acquired by the messages involved in the deadlock. For the purpose of the following discussion, we define row messages as messages that need to take DIM$_0$ hops when normal and column messages are messages that have completed all their row hops and need to take DIM$_1$ hops only when normal.

Row messages may turn into column messages after a few hops, but column messages never turn into row messages. Since row messages use only c_0 virtual channels and column messages use only c_1 virtual channels, there cannot be a deadlock cycle involving both row and column

```
Procedure Fault-Tolerant-Column-Path-Routing(Message $M$)
/* $M$ has a status field to indicate if it is currently misrouted or normal. Initially $M$'s status is normal. */
  1. As long as $M$ is not currently misrouted and the hop specified by the original column-path algorithm is not
     blocked, route $M$ accordingly.
  2. If $M$ is currently not misrouted but the hop specified by the column-path is blocked by a fault, then
       (a) set the status of $M$ to misrouted, and
       (b) route it using the orientation and virtual channels specified in Table 1.
     The misrouting of $M$ is completed and $M$'s status is set to normal if one of the following occurs:
       (a) $M$ is a DIM$_0$ message and reached a corner node of the f-ring it is traversing.
       (b) $M$ is a DIM$_1$ message and it is in the same column as its destination and is on the other side of the faulty
           block that caused the misrouting.
```

Figure 5: Modifications to the column-path routing to handle faults.

messages. Conceptually, the network may be considered as a union of two planes, plane 0 with virtual channels of c_0, and plane 1 with virtual channels of c_1. A message may move from plane 0 to plane 1 but never in the opposite direction. Therefore, if there is a deadlock, then it is among the channels of c_0 or c_1 only.

Class 0 channels are used by two types of row messages: DIM$_0+$ and DIM$_0-$ messages. The DIM$_0+$ messages use virtual channels of c_0 only on DIM$_0+$ physical channels, and virtual channels of c_0 on left columns of the f-rings in the network. The DIM$_0-$ messages use virtual channels of c_0 only on DIM$_0-$ physical channels, and virtual channels of c_0 on right columns of the f-rings in the network. The sets of physical channels and, hence, the set of virtual channels used by DIM$_0+$ and DIM$_0-$ are disjoint. Therefore, there cannot be deadlocks among row messages. A similar argument can be used to show that DIM$_1+$ and DIM$_1$- messages use disjoint sets of physical channels.

3.1 Handling overlapping fault rings

The above routing method can be easily extended to handle overlapping f-rings, when the sets of physical channels of a pair of f-rings are not disjoint. The routing logic remains the same. Because of increased sharing of physical channels more classes of virtual channels are needed to ensure deadlock-free routing.

When two f-rings overlap along a column (respectively, row), some physical channels in that column (respectively, row) belong to the left column (respectively, top row) of one f-ring and to the right (respectively, bottom row) of another f-ring. Take DIM$_0$ messages: our arguments for DIM$_0$ messages for the nonoverlapping case are based on the fact that they use disjoint sets of physical channels. This is no longer true when f-rings overlap in a column. Therefore, DIM$_0+$ and DIM$_0-$ messages should use disjoint sets of virtual channels to break the new dependencies and preserve deadlock freedom. Similarly, when two f-rings overlap in a row, DIM$_1+$ and DIM$_1-$ messages share the physical channels of the row. Once again, new dependencies, this time among DIM$_1$ messages, occur. To break these dependencies we require three classes of virtual channels: c_0, c_1, c_2. Virtual channels used by different types of messages are given Table 2. Extending the above

Table 2: Use of virtual channels for misrouting messages by the column-path algorithm for overlapping fault rings.

Message type	Channel	Used for
DIM$_0-$	c_0	all hops
DIM$_1+$	c_1	all hops
DIM$_1-$	c_2	all hops
DIM$_0+$	c_0	DIM$_0$ hops
	c_1	DIM$_1-$ hops
	c_2	DIM$_1+$hops

arguments, it can be shown the resulting routing is still deadlock free.

3.2 Deadlocks on consumption channels

Another source of deadlocks unique to multicast wormhole routing is the dependency among messages waiting for consumption channels. A unicast message upon reaching its destination does not compete for any additional communication resources. In multicast routing, however, a message may hold consumption channel (from router to processor) in one node and wait for other resources (communication/consumption channels) elsewhere [3].

Figure 7 illustrates the routing of two multicast messages in a row of a two-dimensional mesh. The first message, m_1, originates at node a and has destinations b, c; the second message, m_2, originates at node d and has destinations b, c. Furthermore, nodes a, b, c are left neighbors to b, c, d, respectively. The following scenario is shown in Figure 7. The message m_1 obtains the communication channel from a to b, denoted $[a,b]$, consumption channel in b, denoted $Cons_b$, and communication channel $[b,c]$; m_2 obtains the communication channel $[d,c]$, consumption channel in c, $Cons_c$, and communication channel $[c,b]$. The reservation of the consumption channel is shown by labeling the (flit) buffer associated to it with the name of the message that has reserved it.

Due to flit-level flow control in wormhole routing, node b can accept only the header flit (and may be a few data flits if more than one flit is sent between nodes at a time) of $m1$. Though the consumption channel in b is free, $m1$ cannot be consumed at b until it acquires the consumption

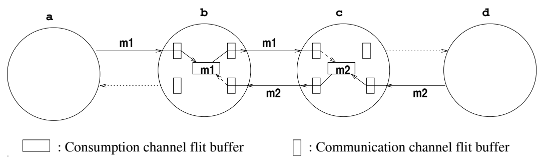
a
b
c
d
m1
m1
m1
m2
m2
m2
: Consumption channel flit buffer
: Communication channel flit buffer

Figure 7: Example of deadlocks on consumption channels.

channel in c also. A similar condition occurs with the consumption channel in c and message $m2$. This causes a circular wait on consumption channels between $m1$ and $m2$, which leads to deadlock.

One solution is to provide multiple classes of consumption channels and allocate them to messages using specific rules. For the fault-free column-path algorithm, two classes of consumption channels are sufficient [3]. One class of consumption channels are used by messages that travel on DIM_{1-} channels and the other class by messages that travel on DIM_{1+} channels. Messages that do not need to take any DIM_1 hops can be treated as DIM_{1+} or DIM_{1-} messages.

Since we use two virtual channels on each physical channel for fault-tolerant routing, more messages of each type compete for consumption channels in routers. Fortunately, the dependencies are still acyclic and therefore do not cause deadlocks if two classes of consumption channels are used as in the original algorithm. To see this, consider DIM_{1+} messages. Let $c+$ be the class of consumption channels they use. Also let the rank of the $c+$ consumption channel in node (i, j), $0 \leq i, j < k - 1$ be j. A DIM_{1+} message needs to deliver data to nodes in a single column only, and never takes DIM_{1-} hops, even when misrouted. So, DIM_{1+} messages do not cause cyclic dependencies on the consumption channels of its class. A similar argument can be used for DIM_{1-} messages. Therefore, two consumption channels per router are sufficient to eliminate deadlocks on consumption channels.

4 Fault-tolerant Hamilton path routing

In this section, we show that the Hamilton path based dual-path and multi-path algorithms can be made fault-tolerant, using two virtual channels per physical channel. We address the dual-path algorithm specifically, since all the dependencies that occur in multipath routing also occur in the dual path algorithm. In this section, Hamilton path and dual-path are used synonymously.

As before, our approach is to leave the original routing logic as it is and add new logic to help messages get around faults when they are blocked. There is one worst case scenario for the dual-path algorithm. This is shown in Figure 8. The source of a multicast message is (0,0) and its destinations are (1,4),(2,1),(3,4),(3,1), in the order to be visited. Because of fault block $\{(1, 2), (2, 2), (3, 2)\}$, however, the message travels on the f-ring many times, to preserve the order in which destinations are visited if there are no faults. This example shows that many classes of virtual channels are required just to keep this message from deadlocking itself. This can be avoided only when the message travels on the f-ring only a few times (typically, once or twice). Thus, faulty blocks spanning multiple rows cause severe problems for Hamilton path algorithms. Therefore, for Hamilton path algorithms, we restrict the fault model to faults with f-rings of height (the number of links in a column of the f-ring) of two or less. Examples of fault-blocks with f-ring height two are less include (i) a block of node-faults in a row, (ii) isolated DIM_0 link faults, and (iii) all possible block faults involving DIM_1 links. Of these, DIM_1 faults do not affect Hamilton path routing, since the routing logic depends on fault-free DIM_1 links only on network boundaries, which, in our fault model, are fault-free.

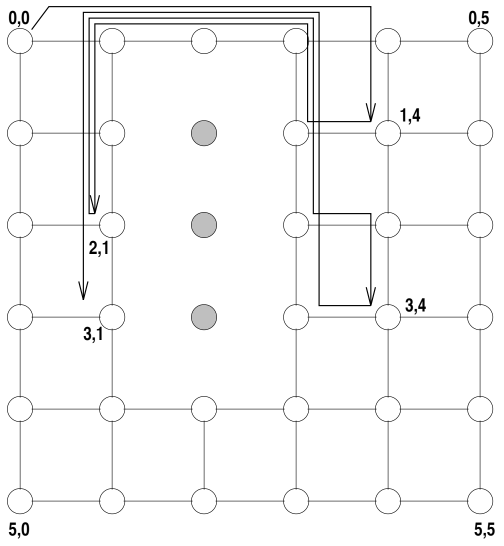

Figure 8: Worst case block fault for Hamilton path algorithms. Directed lines indicate the path of a multicast message from node (0,0) to nodes $\{(1, 4), (2, 1), (3, 4), (3, 1)\}$, visited in the order given.

Now, we present our fault-tolerance technique to handle block faults with f-rings of height two or less. We directly present our method for overlapping f-rings case.

Let M_l (respectively, M_u) be the set of messages that use the Hamilton path H_l (respectively, H_u) in the fault-free network. We use two virtual channels for fault-tolerant routing: c_0 is used exclusively by messages in class M_l and c_1 is used by messages in class M_u. The fault-tolerant routing algorithm is shown in Figure 9.

An M_l message, if blocked by a faulty component in a row, takes the clockwise orientation if the faulty component is to the left of the message, and the counter-clockwise orientation otherwise. An M_u message, if blocked by a faulty component in a row, takes the counter clockwise orientation if the faulty component is to the left of the message, and the clockwise orientation otherwise. A message can be blocked by a fault, while taking a column-hop, only if it is trying to take a short-cut, since no component on the network boundary is faulty. Hence, a message, if blocked on a column-hop, simply continues to the next node in the same row.

Theorem 1 *Procedure Fault-Tolerant-Dual-Path-Routing tolerates multiple block faults of height two or less.*

Proof. First, we observe that messages in class M_l use c_0, while those in M_u use c_1. Hence, deadlocks, if occur, can be only among either M_l messages or among M_u messages. Let us consider M_l messages. Let H_{lp} (respectively,

```
Procedure Fault-Tolerant-Dual-Path-Routing(Message $M$)
/* Messages in class $M_l$ use $c_0$ channels for all hops and those in class $M_u$ use $c_1$ for all hops. */
If the next hop of $M$ is not blocked by a fault
    route it as per the fault-free dual-path algorithm.

If the next hop of $M$ is a row-hop which is blocked by a fault
    1 Let $x$ be the node at which $M$ is blocked and let $y$ be the node in the same row as $x$ at the other end
      of the f-ring. $M$ is misrouted on the f-ring from $x$ to $y$ using the orientation given in Table 3.

If the next hop of $M$ is a column-hop which is blocked by a fault
    1 The column-hop is being used by $M$ to take a short-cut.
    2 $M$ foregoes the opportunity to take the short-cut and takes a row-hop.
```

Figure 9: Modifications to the dual-path routing to handle faults.

Table 3: Orientations and virtual channels used Fault-Tolerant-Dual-Path-Routing algorithm.

Message Type	Current Direction of Travel	F-Ring Orientation (Virtual Channel)
M_l	DIM0+	Counter Clockwise (c_0)
	DIM0−	Clockwise (c_0)
M_u	DIM0+	Clockwise (c_1)
	DIM0−	Counter Clockwise (c_1)

H_{up}) be the H_l (respectively, H_u) network consisting only of c_p virtual channels, for p = 0, 1. For fault-tolerant routing, M_l messages use network H_{l0} (which is also used for the fault-free case) and a part of the network H_{u0}. Only those channels in H_{u0} around the f-rings are used for fault-tolerant routing of messages in M_l. Figure 10 illustrates this case for three overlapping f-rings in a 6×6 mesh. In this figure, messages in M_l use channels in H_{l0} (shown using thin lines with arrows) and a few channels in H_{u0} around the f-rings (shown using dashed lines with arrows).

If an M_l message is blocked at node x, then it is misrouted on the corresponding f-ring to node y, which is in the same row as x and at the other side of the faulty block. Thus, in Figure 10, a message blocked at x must use links marked A, B, C and D to reach y. As per our fault-tolerant logic, a message, once it uses link A, must use links B, C and D to reach y. Hence, the path from X to Y can be replaced with a single link from X to Y as far as dependencies are concerned. The resulting dependency graph is acyclic, and hence there cannot be deadlocks among messages in class M_l. A similar argument holds for deadlock-freedom of messages in class M_u. ∎

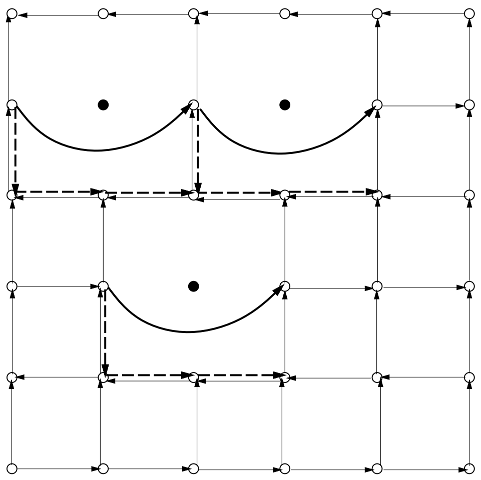

Figure 10: Links used by messages in M_l for fault-tolerant routing. Dashed lines with arrows indicate c_0 channels in the network H_{u0}, and thin lines with arrows indicate c_0 channels in H_{l0}. Thick (curved) lines indicate the net effect of misrouting.

Example. Consider a message from $(5,2)$ with destinations in $\{(4,2),(3,3),(2,0),(1,2),(0,1)\}$. This message takes the path indicated by thick lines with arrows in Figure 11. This message takes a short-cut from $(5,2)$ to $(4,2)$. It tries to take a short-cut from $(4,2)$ to $(3,2)$. However, since $(3,2)$ is faulty, it is routed to $(4,1)$, where it takes a short-cut to $(3,1)$. At $(3,1)$, it is blocked by $(3,2)$, and hence is misrouted to $(3,3)$ as shown in Figure 11.

This example illustrates several possibilities for optimizing the paths taken by messages. For example, $(4,2)$ can route the message to $(4,3)$ based on its knowledge that node $(3,2)$ is faulty; this allows the message to skip the path from $(4,2)$ to $(3,1)$ and then back to $(4,2)$. Using a similar argument, the journey from $(2,0)$ to $(1,0)$ and then back to $(2,0)$ can also be avoided. Such optimizations do not introduce deadlocks, and can be used to improve the latency of messages.

Consumption channel requirements of fault tolerant dual-path algorithm. It has been shown before that, for dual-path and multi-path algorithms, two consumption channels per node are sufficient to avoid deadlocks on consumption channels [3]. One consumption channel is used by messages in class M_l (traveling along H_l) and the other is used by messages in class M_u. Even for the fault-tolerant dual-path routing algorithm, two consumption channels are sufficient to avoid deadlocks on con-

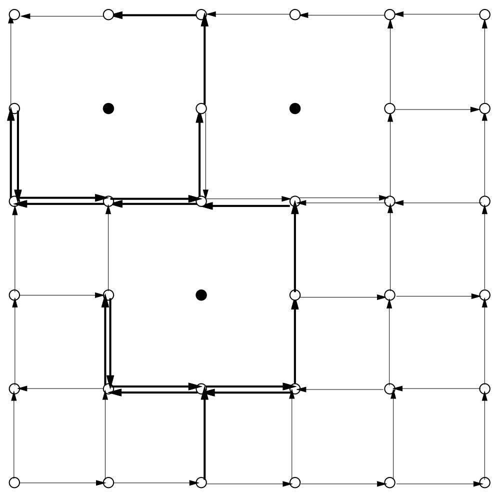

Figure 11: Example of fault-tolerant dual-path routing. Thick directed lines indicate the routing of a message from node $(5,2)$ to nodes $(4,2),(3,3),(2,0),(1,2),(0,1)$.

sumption channels: one consumption channel is used by M_l messages and the other by M_u messages, just as in the fault-free case.

The proof of deadlock-freedom is based on the fact that a message, while it is being misrouted, is not delivered to a destination. In other words, no message waits for consumption channels while it is being misrouted. To illustrate this, let us consider the message M in Figure 11. This message is misrouted from $(1,0)$ to $(1,2)$ via $(2,0)$. Though $(2,0)$ is a destination for this message, M would have acquired the consumption in $(2,0)$ during its journey from $(2,2)$ to $(1,0)$. Hence, during misrouting M does not need to wait for consumption channels. Since no message waits for consumption channels while being misrouted, the dependencies on consumption channels are the same as those in the fault-free Hamilton path algorithms. Thus, two channels are sufficient to avoid consumption channel deadlocks for fault-tolerant Hamilton-path algorithms.

5 Summary and concluding remarks

In this paper, we have addressed the issue of reliable multicast communication in wormhole-switched multicomputers. Our techniques handle multiple convex faults for meshes. When a fault occurs, only the fault-free nodes around the faulty components need to know the information. The concept central to our approach is fault rings, which are formed around each fault and are of rectangular shape for 2-D meshes.

We have specifically considered two recently proposed multicast algorithms: Hamilton path and column-path. With two virtual channels per physical channel, multiple convex faults with nonoverlapping f-rings can be handled by the column-path algorithm. Overlapping f-rings can be handled by the column-path with three virtual channels.

For Hamilton path based algorithms, providing fault-tolerance is more complicated, since a message uses longer paths and visits destinations in a predetermined sequence. However, when convex faults are such that the height of fault rings is two, two virtual channels per physical channel are sufficient to provide fault-tolerant routing.

The column-path and Hamilton path algorithms have different strengths. The column-path is compatible with e-cube and can be easily implemented in the next generation e-cube routers. The column-path is especially attractive when the number of destination is small [3]. The Hamilton path algorithm is a specialized algorithm and attempts to minimize congestion at sources of multicasts by limiting the number of copies of a message to 2 or 4, independent of the number of destinations. For large networks, a combination of these two algorithms may provide more efficient communication with less source congestion. In future, we plan to simulate the fault-tolerant versions of the column-path and Hamilton path algorithms and estimate performance degradation due to faults.

References

[1] R. V. Boppana and S. Chalasani, "Fault-tolerant routing with non-adaptive wormhole algorithms in mesh networks," in *Proc. Supercomputing '94*, Nov. 1994.

[2] R. V. Boppana and S. Chalasani, "Fault-tolerant wormhole routing algorithms for mesh networks," Tech. Rep. CS-94-2, Div. of Math and CS, U. Texas at San Antonio.

[3] R. V. Boppana, S. Chalasani, and C. S. Raghavendra, "On multicast wormhole routing in multicomputers," in *Proc. 1994 IEEE Symp. on Par. and Distr. Processing.*

[4] 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.*, pp. 268–277, 1992.

[5] Cray Research Inc., *Cray T3D Architectural Summary*, Oct. 1993.

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

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

[8] J. Duato, "A theory to increase the effective redundancy in wormhole networks," *Parallel Processing Letters.* To appear.

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

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

[11] X. Lin, P. K. McKinley, and L. M. Ni, "Deadlock-free multicast wormhole routing in 2-d mesh multicomputers," *IEEE Trans. on Parallel and Distributed Systems*, vol. 5, pp. 793–804, Aug. 1994.

[12] L. M. Ni and P. K. McKinley, "A survey of wormhole routing techniques in direct networks," *IEEE Computer*, vol. 26, pp. 62–76, Feb. 1993.

[13] D. K. Panda, S. Singhal, and P. Prabhakaran, "Multidestination message passing mechanism conforming to base wormhole routing scheme," in *Proc. of Parallel Routing and Communication Workshop*, May 1994.

[14] Y.-C. Tseng and D. K. Panda, "A trip-based multicasting model for wormhole-routed networks with virtual channels," in *Proc. 1993 Int. Parallel Processing Symp.*