Fault-Tolerant Routing in Two-Dimensional Mesh Networks with Less-Restricted Fault Patterns*

Sheng-De Wang and Po-Hwa Sui

Department of Electrical Engineering, EE building, Rm. 441

National Taiwan University, Taipei 106, Taiwan

sdwang@cc.ee.ntu.edu.tw

Fax number: 886-2-2367-1909

Abstract

Wormhole routing in networks is prone to deadlocks. Several techniques have been provided to solve the problem, including virtual channels and restriction on the fault patterns. In this paper, we will relax the fault patterns to be one that does not contain the *column-surrounded* fault pattern. In our routing scheme, the concept of off-node is proposed to help messages leave the visited *f*-ring at an appropriate node such that no message encounter the same *f*-ring more than once and, therefore, never get trapped in faulty blocks. Virtual channels are simulated on physical channels to avoid cyclic dependence on channels.

Index Term--- mesh networks, wormhole routing, fault-tolerant routing, virtual channels, deadlocks.

I. Introduction

Direct networks have been a popular means for interconnecting components in massively parallel computers. In a direct network, each node (computer) has a direct connection to only a few nodes, its neighbors, according to the topology of the direct network. Nodes communicate with each other by passing messages through the direct network. Neighboring nodes can pass messages to one another directly, while nodes that are not connected by a direct connection must rely on other nodes in the network to relay messages. The *n*-dimensional (*n*D) mesh is currently the most popular topology for direct network. Low dimensional mesh networks, due to its low node degree, are more popular than high dimensional mesh networks. The 2D mesh topology has been adopted by Symult 2010 [1], Intel Touchstone DELTA [2] and Intel paragon; the MIT J-machine uses the 3D mesh topology.

Network latency is one of the major factors that

affect the performance of a multicomputer system. In order to minimize network latency, the wormhole [3] switching technique has been widely used in the direct networks. In the wormhole technique, a message is divided into packets and a packet is composed of flow control digits or flits. The header flit governs the route. As the header advances along a specific route, the remaining flits follow in a pipeline fashion. A survey of wormhole routing for direct networks can be found in [4].

A multicomputer system can hardly avoid having faulty components in the real world. The well-known *e*-cube algorithm uses a fixed path to route messages between each pair of nodes in a direct network. That is, even one fault in the direct network can prevent the *e*-cube algorithm from successful message passing between many pairs of nodes. A routing algorithm for networks with faults should have the abilities to handle faults with a moderate increasing in routing complexity. Fault-tolerant routing in direct networks has been the subject of extensive research in recent years [5-19]. Chien and Kim [10] present a partial adaptive routing algorithm for *k*-ary *n*-cubes and multi-dimensional meshes. Every fault need to be augmented, by disabling functional nodes, to form *rectangular* faults, or *convex* faults in [10], to assure the correctness of their method. Boppana and Chalasani [12,13,14] propose a method to enhance current routing algorithms for fault-tolerant routing. The concepts of fault ring (*f*-ring) and fault chain (*f*-chain) are introduced and are used for routing messages around *rectangular* faults. Sui and Wang [15] improve the routing algorithm proposed in [14] by using less number of virtual channels. Su and Shin [17] propose an adaptive routing algorithm for meshes and hypercubes, where a given interconnection network is decomposed into two virtual interconnection networks, VIN_1 and VIN_2 . VIN_1 supports deterministic deadlock-free routing and VIN_2 supports fully adaptive routing. In *n*D meshes, their algorithm tolerates

* This work is partially supported by National Science Council, Taiwan, under contract NSC89-2218-E-002-030.

disconnected rectangular faults. In [18], Clalasan and Boppana introduce the *solid fault model*, which can reduce the number of disabled nodes. The solid fault model includes all convex faults and many non-convex faults, such as faults in the shape of L and T. Messages blocked by faults are routed along the fault ring in the direction of either clockwise or counter-clockwise. Overlapped fault rings are not allowed in [18]. Kim and Han [19] also propose a fault-tolerant routing algorithm, which is an extension of the algorithm proposed in [18]. The algorithm in [19] can tolerate any number of overlapped fault rings and fault chains based on the solid fault model.

Since faults occur in a multicomputer system may not belong to the solid fault model, a practical fault-tolerant routing scheme should tolerate a large class of fault patterns. In this paper, a fault-tolerant routing algorithm, which can tolerate fault patterns without column-surrounded fault blocks, for mesh networks is proposed. The rest of this paper is organized as follows. Section II describes terms used in this paper. In section III, the fault-tolerant routing algorithm is proposed. An example of routing is also given in section III. The conclusion is given in Section IV.

II. Preliminary

An n -dimensional mesh has $k_{n-1}k_{n-2}\cdots k_0$ nodes, k_i nodes along dimension i , $0 \leq i \leq n-1$, and $k_i \geq 2$. Each node x is uniquely indexed by an n -tuple $(x_{n-1}, x_{n-2}, \dots, x_0)$, where $0 \leq x_i \leq k_i - 1$. Two nodes $x = (x_{n-1}, x_{n-2}, \dots, x_0)$ and $y = (y_{n-1}, y_{n-2}, \dots, y_0)$ are neighbors if and only if $x_i = y_i$ for all i except one, j , where $x_j = y_j \pm 1$.

Neighboring nodes are connected by direct link implemented by two unidirectional physical channels with opposite directions. The link between nodes x and y is denoted $\langle x, y \rangle$. In this paper, we consider 2D meshes with faults. The four sides of a 2D mesh, hereinafter, labeled as North, East, South and West.

A faulty block is a set of connected faulty nodes and/or links. Channels incident to faulty nodes are considered faulty. In this paper, we assume that no fault touches mesh boundaries. Therefore, a fault ring (f -ring) [12,13,14] consists of fault-free nodes and links can always be formed around each faulty block. By exchanging link status to non-faulty neighbors, each node can know easily its position on an f -ring.

Definition 1: A node on an f -ring is an NE, NW, SE or SW node, if its links to the South and West, to the South and East, to the North and West or to the North and East, respectively, are on the f -ring.

Definition 2: Two faulty blocks are connected if their f -rings share some common channels; otherwise, they are disconnected.

Definition 3: A faulty block F1 is column-surrounded by another faulty block F2 if there exists a node (x_1, y_1) on the f -ring of F1 and two nodes (x_2, y_2) and (x_3, y_3) on the f -ring of F2 such that $x_2 < x_1 < x_3$ and $y_1 = y_2 = y_3$.

Definition 4: A faulty block F1 is row-surrounded by another faulty block F2 if there exists a node (x_1, y_1) on the f -ring of F1 and two nodes (x_2, y_2) and (x_3, y_3) on the f -ring of F2 such that $y_2 < y_1 < y_3$ and $x_1 = x_2 = x_3$.

Definition 5: Faulty blocks that are neither column-surrounded nor row-surrounded are un-surrounded.

Definition 6: A node (x_1, y_1) is column-surrounded by a faulty block F if there exist two nodes (x_2, y_2) and (x_3, y_3) on the f -ring of F such that $x_2 < x_1 < x_3$ and $y_1 = y_2 = y_3$.

Definition 7: A node (x_1, y_1) is row-surrounded by a faulty block F if there exist two nodes (x_2, y_2) and (x_3, y_3) on the f -ring of F such that $y_2 < y_1 < y_3$ and $x_1 = x_2 = x_3$.

From above definitions, a faulty block is a solid fault if there are no non-faulty nodes are column-surrounded or row-surrounded by the fault block. That is, if there are two faulty links in one faulty block that are on same row or column, then all nodes between these two links are also faulty. In this paper, we consider faults that are not column-surrounded. In other words, solid and row-surrounded faults can be tolerated. To the best of our knowledge, all fault-tolerant routing schemes in the literature consider un-surrounded faults with some further constraints.

When faults occur, a pre-work consists of two phases is performed for setting node status. In phase one, information of node position on f -rings is collected. Each NE node on f -rings sends a message along the f -ring in counter-clockwise (C.C.W.). The message contains the coordinates of the NE node and four fields, D0max, D0min, D1max and D1min, which are used to record the maximal and minimal index values of dimensions 0 and 1 of all nodes on the f -ring. Nodes of types NE, NW, SE and SW intercept and process the message. The message is then forward on the f -ring in C.C.W.

Phase one: /* assume the message is generated at node (x_1, y_1) , node (x_2, y_2) is processing the message. D0max and D0min are initialized to y_1 , D1max and D1min are initialized to x_1 */

1. If $(y_2 > y_1)$ or $(y_2 = y_1, x_2 > x_1)$, abort the message.
2. If $x_2 > D1max$, $D1max = x_2$.
3. If $x_2 < D1min$, $D1min = x_2$.
4. If $y_2 > D0max$, $D0max = y_2$.
5. If $y_2 < D0min$, $D0min = y_2$.

Step 1 in phase one can assure that only the message generated by the East-most North-most node survives, hence, can reduce the traffic. Since each node of types of NE, NW, SE and SW processes the message, the maximal and minimal index values of all nodes on

the f -ring on dimensions 0 and 1 are collected when the message return to the East-most North-most node. The collected information is used for setting node position status in phase two. The East-most North-most node sends the collected information along the f -ring in C.C.W. Each node on the f -ring compares its index values to the collected information and set its node position status, if needed. We assume that each node has six flags, N, W, S, E, B1 and B2, to indicate its position on f -ring. Flags B1 or B2 are used to indicate that a node is the East-most North-most or East-most South-most node, respectively, on an f -ring. These two nodes play important roles in our routing scheme proposed in the following section. The source node in phase two sets directly its B1 flag, for it is the East-most North-most node of the f -ring.

Phase two: /* assume the current node is (x_1, y_1) */

1. If $x_1 = D1max$, then sets N.
2. If $y_1 = D0min$, then sets W.
3. If $x_1 = D1min$, then sets S.
4. If $y_1 = D0max$, then sets E.
5. If $(y_1 = D0max, D1max \neq 0)$, then sets B2 and $D1max = 0$.

Step 5 is used to set the B2 flag of the East-most South-most node, which is the first node with its E flag set. Since the value of $D1max$ can't be 0, it is used as a condition to set B2 flag. The value of $D1max$ is then set to be 0 to prevent any other node sets its B2 flag.

Definition 8: On an f -ring, the set of nodes with E or W flag set forms the E_{max} or E_{min} , respectively, of the f -ring, and the set of nodes with N or S flag set forms the N_{max} or N_{min} , respectively, of the f -ring.

Definition 9: At any node, the first hop of the path, which is specified by the e -cube algorithm for a message, is called the e -hop for the message at that node.

III. Fault-Tolerant Routing

3.1. The Fault-Tolerant Routing

The e -cube algorithm is used as a basis in our fault-tolerant routing scheme. In our scheme, messages are classified into one of the following four types: WE (West-to-East), EW (East-to-West), NS (North-to-South), and SN (South-to-North). WE messages and EW messages are collectively called row messages, NS messages and SN messages are column messages. Each message is injected into the network as either a WE or an EW message, depending on the relative position of its source and destination node. Once a row message completes its routing on dimension 0, it becomes a column message. Thus, row messages can become column messages, but column messages never become row messages. While a message being transmitted, it is in direction of null,

clockwise (C.W.) or counter-clockwise (C.C.W.) in the network. When a message is newly generated, it is assigned to have null direction. Procedure Set-Type (in Fig. 1) defines the conditions for a row message becomes a NS or SN column message.

Procedure Set-type (M)

/* Assume that message M is currently at node (c_1, c_0) and destination node is (d_1, d_0) */

1. If M is a row message and $c_0 = d_0$ then
 set its direction to be null and change its type to
 NS, if $c_1 > d_1$, or SN, if $c_1 < d_1$.

Fig. 1 Procedure Set-Type

In fault-free regions, messages have null direction and are routed along their e -hops. When the message is blocked by a fault, depending on the message type and its location on the corresponding f -ring, direction of clockwise, counter-clockwise or null is assigned. The message then routes on the f -ring along its specified direction. After bypassing the fault encountered or the message changes its type, it is assigned to have null direction again. Table I. shows the directions of which messages are assigned to route around faults.

Table I. Routing direction for messages

Message Type	Direction
WE	C.W.
EW	C.C.W.
NS	C.C.W.
SN	C.W.

When a WE (EW) message enters a fault block the first time at a node located at E_{min} , N_{max} or N_{min} (E_{max} , N_{max} or N_{min}) of the corresponding f -ring on the first time, its direction remains null. The E_{min} , N_{max} and N_{min} (E_{max} , N_{max} or N_{min}) of an f -ring is called the *normal set* for WE (EW) messages. When a WE (EW) row message encounters a fault the first time at a node not in its normal set of the corresponding f -ring, its direction is set to be clockwise (counter-clockwise). The WE (EW) message then routes on the f -ring along the specified direction until either its routing on dimension 0 is completed or the E_{max} (E_{min}) of the current f -ring is reached. If the E_{max} (E_{min}) has been reached before the row message completes its routing on dimension 0, the row message leaves the f -ring at E_{max} (E_{min}); otherwise, it changes into a column message. In both cases, the row message is reset to have null direction. Messages of WE and EW use virtual channels of class 0 (VC_0) and class 1 (VC_1), respectively, on their journey. WE (EW) row messages, if have not yet changed into column messages, leave at the respective East-most (West-most) column of the f -rings encountered and route along their e -cubes in fault-free area, no row message encounter a same f -ring more than once.

When a NS (SN) message encounter a fault the first time at a node in its normal set, N_{min} , E_{max} and E_{min} (N_{max} , E_{max} and E_{min}), of the corresponding f -ring, its direction remains null. If the node, at which a NS (SN) column message encounters a fault the first time, is not in its normal set of the corresponding f -ring, the direction of the message is set to be counter-clockwise (clockwise). The NS (SN) column message is then routed on the f -ring along direction of counter-clockwise (clockwise) until its *off-node* (to be defined below) on the current f -ring or its destination node is reached. When a row or a column message reaches a node located at the overlapped portion of two f -rings the first time, it can choose either of the two f -rings to route on, provided that the message is not already on one of the two f -rings.

Every column message leaves each f -ring at its off node. We assume that there is a field in the message header that holds the index of the candidate off-node and this field can be updated while the message is tracing on an f -ring. The candidate off-node is initialized to be the node at which the column message encounters the f -ring the first time. The update policy is as follows. Each time the NS (SN) message gets to the destination column, it examines the index value of the current node. If the index value in dimension 1 of the current node is smaller (larger) than that of the candidate off-node but is greater (smaller) than or equal to that of the destination node, then current node is set to be the candidate off-node for the NS (SN) message. If the index value of the current node in dimension 1 is smaller (greater) than that of the destination node, the candidate off-node is unchanged. The candidate off-node becomes the off-node for the NS (SN) message in the following two cases: a) if the candidate off-node is in the N_{min} (N_{max}); b) the candidate off-node been reached again after visiting all the nodes on the f -ring. It is noticed that the off-node for a column message is on its destination column. The index value of the off-node for a NS (SN) message in dimension 1 is larger (smaller) than or equal to that of the destination node, but is smaller (greater) than that of nodes, which are above (below) the destination node, on the destination column. Therefore, on the destination column, there is no node on the f -ring locates between the off-node and the destination node of a NS (SN) message.

In order to avoid the situation in which column messages, while identifying their off nodes on a same f -ring, block each other, we define two *breaking nodes* on each f -ring. Column messages switch to a different set of virtual channels each time they pass through the breaking node. The East-most North-most node, whose B1 flag is set, on each f -ring is the breaking node for NS messages, and the East-most South-most node, whose B2 flag is set, is the breaking node for SN

messages. NS (SN) message uses VC_2 (VC_5) before passing through the breaking node the first time. Depending on the position where a column message encounters an f -ring, the column message may pass through the breaking node one or two times before identifying its off node. NS (SN) messages use VC_3 (VC_6) and VC_4 (VC_7) after passing through the breaking node the first and the second time, respectively. The usage of virtual channels is depicted in Fig. 2.

Fig.2 Usage of virtual channels by WE, EW, NS and SN messages

Fig. 3 A possible deadlock in fault-free area surrounded

If the off-node for a NS (SN) message is not in the South-most (North-most) row of the corresponding f -ring, the destination node is in the fault-free area surrounded by the f -ring. A 1-bit flag, called FLAG, is used to record this status when detected. That is, if the index value in dimension 1 of a visited node on the destination column is smaller (greater) than that of the destination node while a NS (SN) message is identifying its off node, the message sets FLAG to 1. When a message gets to its off-node, the value of its FLAG is used to guide the selection of virtual

channels, such that the column message will not be blocked by other column messages routing in the surrounded fault-free area. Fig. 3 shows a possible deadlock example among NS messages if the same discipline on selecting virtual channels is used within fault-free area surrounded by f -ring or not. In Fig. 3, if message M_1 , from S_1 to D_1 , use VC_2 in the surrounded fault-free area, M_1 and M_2 , which is from S_2 to D_2 , will block each other by requesting VC_2 .

In order to eliminate the situation shown in Fig.3, column messages, whose FLAG are set, must not use the same class of virtual channels as in the ordinary fault-free area. In our scheme, NS (SN) messages use VC_3 (VC_6) in the fault-free area surrounded, if their FLAG are set. Table II shows that how virtual channels are used by column messages before they passing through the breaking node the first time, after passing through the breaking node the first and the second time and routing in the fault-free area surrounded.

Table II. Usage of virtual channels by column messages

Message type	Before passing through the breaking node the first time	After passing through breaking node the first time	After passing through breaking node the second time	In surrounded fault-free area (FLAG set)
NS	VC_2	VC_3	VC_4	VC_3
SN	VC_5	VC_6	VC_7	VC_6

Steps 1, 2, 3 and 4 in Procedure Set-Direction (Fig. 4) are used for resetting a message to have null direction again. In step 5, depending on the message type (WE, EW, NS or SN), and the location of current node (c_1, c_0), either C.W. or C.C.W. is assigned. Our routing algorithm FT-Route is given in Fig. 5.

Procedure Set-Direction (M) /* Assume that message M is currently at (c_1, c_0) */

1. If M is a WE and the direction of M is not null then
 - if (c_1, c_0) is in the Emax then
 - set its direction to null
 - else return
2. If M is an EW and the direction of M is not null then
 - if (c_1, c_0) is in the Emin then
 - set its direction to null
 - else return
3. If M is a NS and the direction of M is not null then
 - if (c_1, c_0) is the off-node then
 - set its direction to null
 - else return
4. If M is an SN and the direction of M is not null then
 - if (c_1, c_0) is the off-node then
 - set its direction to null
 - else return
5. If the direction of M is null and (c_1, c_0) is on an f -ring but not in its normal set, then set its direction to be C.C.W., if M is an EW or NS message, or

C.W., if M is a WE or SN message.

Fig. 4 Procedure Set-Direction

FT-Route (M)

1. Set-Type (M).
2. Set-Direction (M).
3. If the direction of M is null then
 - route M along its e -hop
- else
 - route M along the specified direction.

Fig. 5 Fault-tolerant routing algorithm

3.2. Example

Let's consider the example of routing a message M from node (7,0) to node (3,10) in Fig.6. M begins as a WE message and is routed along its e -hop. After three hops, M gets to node (7,3), where its e -hop is faulty. It then travels along the f -ring of F1 in clockwise orientation from node (7,3) to node (13,8). Since node (13,8) is in the Emax of the f -ring, the direction of M is reset to be null at the node. M then routes along its e -hop to node (13,10), where M become a NS message. From nodes (7,0) to (13,10), M uses VC_0 . At node (13,10), M routes along its e -hop to node (12,10). At node (12,10), due to its faulty e -hop, M begins to travel along the f -ring of F2 in counter-clockwise orientation and node (12,10) becomes its candidate off-node. Node (10,10) becomes the candidate off-node when M gets to it, for its index value in dimension 1 is smaller than that of the previous candidate off-node (12,10) and is greater than that of the destination node (3,10). When M gets to node (2,10), the FLAG is set and the candidate off-node is still node (10,10), for the index value of node (2,10) in dimension 1 is smaller than that of the destination node. Node (5,10) becomes the candidate off-node when M gets to it. M then travels afterward along the f -ring of F2 with its off-node unchanged. From nodes (13,10) to (6,15), M uses VC_2 . Since node (6,15) is the breaking node for NS messages, M begins to use VC_3 at node (6,15). When M travels to node (5,10) again, node (5,10) becomes the off-node for M . Direction of M is then reset to be null and M routes along its e -hop to its destination node (3,10) using VC_3 , for its FLAG is set.

3.3. Proof of Deadlock-Freedom

1. Each type of the messages uses disjoint set of virtual channels. 2. Row messages (EW and WE) can become column messages (NS and SN), but not vice-versa. 3. The two types of row messages (EW and WE) cannot change into each other and the two types of column messages (NS and SN) cannot change into each other. From the above facts, to prove that routing Algorithm FT-Route is deadlock-free, only proofs of deadlock-freedom for each of the four types of messages are needed. Since the routing disciplines are the same for WE and EW messages and so are the routing disciplines for NS and SN messages, only

proofs of deadlock-freedom for WE and NS messages are given below.

Lemma 1: No deadlock occurs among WE messages.

Proof: An integer channel number is assigned to each virtual channel from West to East in VC_0 . In an $X \times Y$ mesh, virtual channels from node (i, j) to node $(i, j+1)$ in VC_0 has channel number j , where $0 \leq i \leq X-1, 0 \leq j \leq Y-2$. When faults occur, each virtual channel of VC_0 in C.W. on f -rings is assigned a non-integer channel number. When the top-most node in the Emax of an f -ring is (x, y) , a non-integer channel number between $y-1$ and y , $0 \leq y \leq Y-2$, is assigned. It is assigned increasingly from the virtual channel leaving the top-most node in the Emax to the virtual channel entering the top-most node in the Emax in C.W. orientation. A channel number assignment example is given in Fig. 7. It is noticed that virtual channels of VC_0 from West to East on f -rings in C.C.W. still have integer channel number.

1. WE messages routed in fault-free region travel virtual channels of VC_0 in an increasing order on channel number.

2. The channel number of any West-to-East virtual channel of VC_0 entering an f -ring is less than that of every virtual channel of VC_0 in C.W. on the f -ring.

3. WE messages routed on f -rings travel virtual channels of VC_0 in an increasing order on channel number.

4. The channel number of the virtual channels of VC_0 leaving an f -ring at the Emax is larger than that of every virtual channel of VC_0 on the f -ring.

From above facts, each WE message travels virtual channels in an increasing order on channel number on its journey. Hence, no deadlock occurs among WE messages. $\square$

Lemma 2: No deadlock occurs among EW messages.

Proof: Since the routing disciplines are the same for messages WE and EW, proof for lemma 2 is similar to the proof given for lemma 1 and is omitted.

Lemma 3: No deadlock occurs among NS messages.

Proof: Lemma 3 is proved by showing that no deadlock can occur among NS messages in the following three cases: a) on one f -ring, b) on one f -ring and in the fault-free area surrounded by the f -ring and c) on multiple f -rings.

NS messages travel virtual channels on f -rings in counter-clockwise orientation. Each time they pass through the breaking node, they use a different set of virtual channels. Depending upon the number of times NS messages pass through the breaking node, VC_2 , VC_3 and VC_4 are used sequentially by NS messages. Since no NS message can pass through the breaking node more than two times, three sets of virtual channels are enough to guarantee that no deadlock can occur among NS messages routing on one f -ring.

When the destination node of a NS message is

located in the fault-free area surrounded by an f -ring, VC_3 , instead of VC_2 , is used by the NS message in the fault-free area. The NS messages, which just traveled on the f -ring, and the NS messages, which are in the fault-free area surrounded but have not yet encounter the f -ring, use virtual channels of VC_3 and VC_2 , respectively, in the fault-free area surrounded. Therefore, there is no channel dependence among the NS messages, which just traveled on the f -ring, and the NS messages, which are in the fault-free area surrounded but have not yet encounter the f -ring, on an f -ring and in the fault-free area surrounded.

1. The off-node of NS message is surely below the node where the NS message enters the f -ring; and NS messages route from North to South in the fault-free area; and no fault block is column-surrounded, hence, NS messages travel f -rings acyclically. 2. NS messages travel the overlapped portions of connected f -rings in opposite direction. Therefore, no deadlock can occur among NS messages routed on multiple f -rings.

From above discussion, we assure that no deadlock can occur among NS messages. $\square$

Lemma 4: No deadlock occurs among SN messages.

Proof: Since the routing disciplines are the same for NS and SN messages, proof for lemma 4 is similar to the proof given for lemma 3 and is omitted.

Theorem 1: Algorithm FT-Route is deadlock-free.

Proof: From lemmas 1, 2, 3, and 4, it is clear that no deadlock can occur among each of the four types of messages. Thus, FT-Route is a deadlock-free routing algorithm. $\square$

IV. Conclusion

Most of the fault-tolerant routing schemes for mesh networks in the literature assume rectangular or solid faults, although in practice it is likely that faults of irregular shape may occur. In this paper, we proposed a fault-tolerant routing algorithm for 2D meshes that can tolerate more relaxed fault patterns. We consider a kind of relaxed fault patterns where there is no column-surrounded fault. Misrouted messages are routed along f -rings until getting to an appropriate node such that messages never encounter the same f -ring more than once. Messages therefore never get trapped in fault blocks. Virtual channels are simulated on physical channels to avoid deadlock among messages. If we consider the case where fault blocks are disconnected, half of the virtual channels used in this paper are enough for our algorithm to be deadlock-free since nodes on f -rings corresponding to this case are disjoint.

References:

- [1] C.L. Seitz, W.C. Athas, C.M. Flaig, A.J. Martin, J.

- Seizovic, C.S. Steele, and W.K. Su, "The architecture and programming of the Ametek Series 200 multicomputer," *Proc. 3rd Conf. Hypercube Concurrent Computers and Applications*, Jan. 1988, vol. I, pp. 33-36, Association for Computing Machinery.
- [2] Intel Corp., *A Touchstone DELTA System Description*, 1991.
- [3] W.J. Dally and C.L. Seitz, "Deadlock-free message routing in multiprocessor interconnection networks," *IEEE Trans. Computers*, vol. C-36, no. 5, May 1987, pp. 547-553.
- [4] L.M. Ni, and P.K. McKinley, "A survey of wormhole routing techniques in direct networks," *IEEE Computer* 26, Feb. 1993, pp. 62-76.
- [5] W.J. Dally and H. Aoki, "Deadlock-free adaptive routing in multicomputer networks using virtual channels," *IEEE Trans. Parallel and Distributed Systems*, vol. 4, no. 4, April 1993, pp. 466-475.
- [6] C.J. Glass and L.M. Ni, "Fault-tolerant wormhole routing in meshes," *Proc. 23th Ann. Int'l Symp. Fault-Tolerant Computing*, 1993, pp. 240-249.
- [7] D.H. Linder and J.C. Harden, "An adaptive and fault tolerant wormhole routing strategy for k -ary n -cubes," *IEEE Trans. Computers*, vol. 40, no. 1, Jan. 1991, pp. 2-12.
- [8] M.S. Chen and K.G. Shin, "Depth-first search approach for fault-tolerant routing in hypercube multiprocessors," *IEEE Trans. Parallel and Distributed Systems*, vol. 1, no. 2, April 1990, pp. 152-159.
- [9] M.S. Chen and K.G. Shin, "Adaptive Fault-tolerant routing in multicomputers," *IEEE Trans. Computers*, vol. 39, no. 12, Dec. 1990, pp. 1406-1416.

Fig.6 A routing example from (7,0) to (3,10)

- [10] A.A. Chien and J.H. Kim, "Planar-adaptive routing: low-cost adaptive networks for multi-processors," *Proc. 19th Int'l Symp. Computer Architecture*, 1992, pp. 268-277.
- [11] T.C. Lee and J.P. Hayes, "A fault-tolerant communication scheme for hypercube computers," *IEEE Trans. Computers*, vol. 41, no.10, Oct. 1992, pp.1242-1256.
- [12] R.V. Boppana and S. Chalassani, "Fault-tolerant routing with non-adaptive wormhole algorithms in mesh networks," *Supercomputing*, 1994, pp. 693-702.
- [13] S. Chalassani and R.V. Boppana, "Adaptive fault-tolerant wormhole routing algorithms with low virtual channels requirements," *Int'l Symp. Parallel Architecture, and Networks*, 1994, pp. 214-221.
- [14] R.V. Boppana and S. Chalassani, "Fault-tolerant wormhole routing algorithms for mesh networks," *IEEE Trans. Computers*, vol. 44, no. 7, July 1995, pp. 848-864.
- [15]
- [16] G.M. Chiu and S.P. Wu, "A fault-tolerant routing strategy in hypercube multicomputers," *IEEE Trans. Computers*, vol.45, no. 2, Feb. 1996, pp. 143-155.
- [17] C.C. Su and K.G. Shin, "Adaptive fault-tolerant deadlock-free routing in meshes and hypercubes," *IEEE Trans. Computers*, vol. 45, no. 6, June 1996, pp.666-682.
- [18] S. Chalassani and R.V. Boppana, "Communication in multicomputers with nonconvex faults," *IEEE Trans. Computers*, vol. 46, no. 5, May 1997, pp. 612-622.
- [19] S.P. Kim and T. Han, "Fault-tolerant wormhole routing in mesh with overlapped solid fault regions," *Parallel Computing*, vol. 23, 1997, pp. 1937-1962.

Fig. 7 Channel number assignment