Fault-Tolerant Wormhole Routing Algorithms in Meshes in the Presence of Concave Faults

Seungjin Park
Dept. of Computer Science
Michigan Tech. University
Houghton, MI 49931, U.S.A.
spark@mtu.edu

Jong-Hoon Youn
Dept. of Computer Science
Oregon State University
Corvallis, OR 97330, U.S.A.
jhyun@cs.orst.edu

Bella Bose
Dept. of Computer Science
Oregon State University
Corvallis, OR 97330, U.S.A.
bose@cs.orst.edu

Abstract

A fault ring is a connection of only nonfaulty adjacent nodes and links such that the interior of the ring contains only faulty components. This paper proposes two wormhole routing algorithms that deal with more relaxed shapes of fault rings than previously known algorithms [1, 2, 3] in the mesh networks. As a result, the number of components to be made disabled would be reduced considerably in some cases. First algorithm, called F4, uses four virtual channels and allows all four sides of fault rings to contain concave shapes. Second algorithm, F3, permits up to three sides to contain concave shapes using only three virtual channels. Both F3 and F4 are free of deadlock and livelock and guarantee the delivery of messages between any pair of nonfaulty and connected nodes in the network.

1. Introduction

Since processors in a multicomputer network need to communicate with other, efficient communication is essential to enhance the performance of the system. The wormhole-routing switching has been dominant for its low-latency communication, and it has been adopted by numerous parallel machines.

Two important issues in designing routing algorithms are deadlock and livelock freedom. These properties are necessary for the guaranteed delivery of a message to its destination. Another issue in designing routing algorithm is the extent of the fault information available at each node. In routing strategy used in the paper, each node needs to know only the local fault information [1, 2, 3, 4]. With this limited information the routing algorithms becomes simple, and thus the routing decision at each node becomes fast. However, the resulting path by the algorithms may not be the optimal.

A set of adjacent faults forms two types of fault shapes in mesh networks: *fault-ring* and *fault-chain*. If a connection of only nonfaulty adjacent nodes and links forms a ring such that the interior of the ring contains only faulty components, then the shape is called fault-ring (*f-ring* in short). When an f-ring touches one or more boundaries of the network, it becomes a fault-chain (*f-chain* in short).

An extensive amount of work has been done on fault-tolerant routing in mesh networks [1, 2, 3, 4]. Among them, we briefly explain the most relevant works to our algorithms. Boppana and Chalasani [1] have proposed fault-tolerant routing algorithms in mesh networks using wormhole technique. Their algorithms, based on the e-cube routing, are deadlock- and livelock-free using only local knowledge of faults. Both f-rings and f-chains are allowed in their fault model, however, the shape of f-rings is restricted to rectangle. Four virtual channels are used to implement the algorithm. Later Sui and Wang [3] have improved the algorithm using only three virtual channels. In [2] Boppana and Chalasani proposed an algorithm tolerating more general forms of f-rings, called *non-convex*. Let a, l_1, and l_2 be any nonfaulty node and two faulty links of an f-ring. In nonconvex fault model, no nonfaulty node, a, exists between two faulty links, l_1 and l_2, in the same row (respectively, column) of the mesh.

While permitting arbitrary fault patterns may still be an elusive goal, it would be desirable to relax the restriction on the shape of the f-rings. To achieve this goal, we present a routing algorithm which tolerates more relaxed fault models than those in [1, 2, 3] using four virtual channels. Further, we also show another algorithm using only three virtual channels for more restricted f-rings. Our algorithms are better than those in [1, 2, 3] in terms of the number of processors that should be made faulty. Each nonfaulty node in the proposed algorithms may need to know the status of nodes beyond its neighbor nodes. In order to do this, it may need additional time. Even though we consider only nonoverlapped f-rings in this paper, the proposed algorithms can easily be extended to deal with f-chains and overlaps by creating more virtual channels.

The rest of the paper is organized as follows. Section 2 describes the necessary information to understand the paper. Section 3 explains the new fault-tolerant algorithms. Conclusion is presented in Section 4.

2. Background

In this section we depict the fault models that have been adopted in the design of the proposed algorithms. In our fault model, both node and link failures are considered, i.e., either the entire processor element (node) and its associated router or any communication line (link) could fail. It is also assumed that the faults do not disconnect the network.

A *node* in column *a* and row *b* is represented by *(a, b)* in a 2D mesh. A *link* that connects node *(a, b)* and node *(c, d)* is represented by *(a, b)*↔*(c, d).*

Based on the number of faulty links incident to a node, each node on an f-ring falls into one of the following three categories:

1) *Convex node* – no faulty link incident to it.
2) *Concave node* – exactly two faulty links incident to it.
3) *Plain node* – only one faulty link incident to it.

A portion of an f-ring that consists of two convex nodes with only plain nodes, if any, between them is called *convex section*. Likewise, if a portion consists of two concave nodes and possibly with some plain nodes in between them is called *concave section*. If a convex section of an f-ring is projecting towards the West (respectively, East, South, North) side of the f-ring, it is called *West (*respectively, *East, South, North) convex section*. To distinguish the two convex nodes in a convex section, we call them the *upper*, the *lower*, the *left* and the *right convex nodes*, respectively, according to their relative positions in the section.

In our fault models, there might be several f-rings. We divide each f-ring into four portions: the *North, South, West and East sides*. All nodes and links between the upper convex node of the upper-most West (respectively, East) convex section and the lower convex node of the lower-most West (respectively, East) convex section form the West (respectively, East) side. Note that there exists at least one East and one West convex sections. If there is only one West (respectively, East) convex section, the West (respectively, East) side is between upper and lower convex node of the section. The North (respectively, South) side is the northern (respectively, southern) portion between the East and West sides.

EXAMPLE 1: In Fig. 1, some of the convex nodes are (0, 2), (0, 3), (1, 4), some of the concave nodes are (1, 2), (1, 3), (2, 4), (2, 5), and some of the plain nodes are (3, 7), (4, 3), (6, 2). {(1, 5), (1, 6), (1, 5)↔(1, 6)} and {(2, 7), (3, 7) (4, 7), (2, 7) ↔(3, 7), (3, 7) ↔(4, 7)} are some examples of convex section, and {(4, 6), (5, 6), (6, 6), (4, 6)↔(5, 6), (5, 6)↔(6, 6)} and {(2, 4), (2, 5), (2, 4) ↔(2, 5)} are some examples of concave section. {(8, 3), (8, 4), (8, 5), (8, 3)↔(8, 4), (8, 4)↔(8, 5)} is an example of East convex section, and {(2, 4), (2, 5), (2, 4) ↔(2, 5)} is a West concave section. (1, 6) is the upper convex node of the upper-most convex section, and (0, 2) is the lower convex node of the lower-most convex section. Therefore, the portion from node (0, 2) to (1, 6) is the West side, from (8, 3) to (8, 5) is the East side, from (1, 6) to (8, 5) is the North side, and from (0, 2) to (8, 3) is the South side.

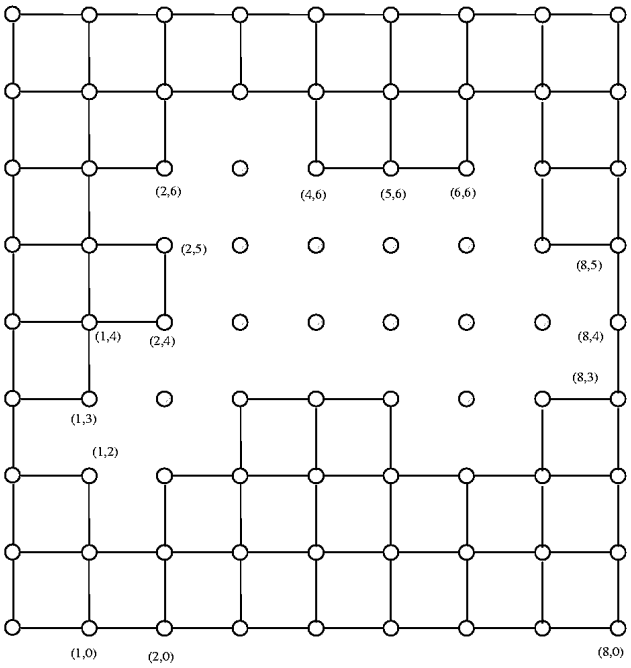

Figure 1. An example of an f-ring**.**

2.1 Fault model

Let us take a portion of a side of an f-ring and align it along the *x*-axis with the faults below it. Let $l_1 = (x_a, y_b) \leftrightarrow (x_a, y_{b+1})$, $l_2 = (x_c, y_d) \leftrightarrow (x_c, y_{d+1})$ be any two faulty links from the inside of the same f-ring. In our fault model, if $x_a = x_c$, every nodes between l_1 and l_2 are faulty.

For all nodes in the portion, if their *y* values do not decrease (resp., increase) as *x* values increase, the portion is called *monotonically increasing* (resp., *monotonically decreasing*). In Fig. 1, for example the portions from node (1, 6) to node (4, 7) and from node (1,1) to node (0, 2) are monotonically increasing, and the portions from node (7, 2) to (3, 3) and from (6, 7) to (8, 5) are monotonically decreasing.

If a side is monotonically increasing first and then monotonically decreasing, then it is called *convex side.* Likewise, if a side is monotonically decreasing first and then monotonically increasing, then it is called *concave side.* If it consists of a combination of any number of monotonically increasing portion and monotonically decreasing portion, then it is called *zigzag side*. Note that a convex (or a concave) side is a zigzag side.

As mentioned in Section 1, Chalasani and Boppana [1, 2] have proposed routing algorithms to handle the case in which each side of f-rings is only convex (it is called *nonconvex* in [2]). Examples of f-rings allowed in their algorithms are (a) and (b) in Fig. 2. However their algorithm may not work properly, if an f-ring contains some concave sides as shown in (c) and (d) of Fig 2. Therefore, if one wants to use the algorithms, concave sides should be made convex by converting some nonfaulty nodes to faulty. This results in the waste of costly resources. In this paper we propose a fault-tolerant routing algorithm which can tolerate f-rings with zigzag sides. Thus our algorithm works correctly for f-rings in Fig. 2 (a), (b), (c), and (d). However, Fig 2. (e) is not allowed in our algorithm because the North side of the f-ring is not a combination of only monotonically increasing and decreasing portions.

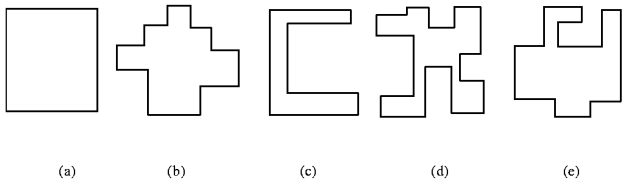

Figure 2. Several examples of f-rings.

2.2 Formation of fault rings

Messages are routed around the f-rings in two directions: clockwise and counter-clockwise. As shown in [1, 2, 3], only the fault status of the neighbor nodes is enough for any node to determine its two neighbor nodes on the f-ring in each direction. For example, in Fig. 1, since south link of node (2, 4) and east link of (1, 3) are faulty, it can be easily determined that the neighbor nodes of (1, 4) in counter-clockwise and clockwise directions are (1, 3) and (2, 4), respectively. These are also called *left* and *right* nodes, respectively. Refer to [2] for full description of the f-ring formation algorithm.

Each node in f-rings can easily be determined whether it is convex, concave or plain by the number of faulty links incident to it.

2.3 Node position on an f-ring

In addition to the knowledge about the neighbor nodes on f-rings, each node in our f-ring models has to know its *position* on the f-ring, i.e., which side it resides: North, South, East or West. Since an f-ring is a ring, every node on an f-ring has exactly two incident links on the f-ring. Let X_{ab} indicate a convex node with links in a and b directions, where $a, b \in$ {W(est), E(ast), S(outh), N(orth)}. Likewise, V_{ab} indicates a concave node. In the rest of the section, a *node* indicates a convex or concave node only, unless otherwise specified. Below we briefly describe the distributed algorithm for each node to attain its position information on the f-ring.

Step 1: Every node in the f-ring sends its incident f-ring link information to its neighbor nodes in both directions.
Step 2: Each node receives information from both neighbor nodes and determines, if possible, its position on the f-ring using the rules given in Table 1. For example, suppose the type of current node is X_{ES} and its left node is X_{EN}, then the two nodes and all plain nodes, if any, between them form a West convex section. Thus, in our fault model all convex, concave and plain nodes in this section are in the West side. Note that only the nodes between the same type, i.e., between convex and convex or between concave and concave, know their positions after Step 2.
Step 3. Nodes between different types, namely convex and concave, determine their positions. Suppose a node, v of type say convex, knows its position, $p \in$ {W, E, N, S}, this may be due to its left neighbor node, u, is of the same type-in this case convex. Furthermore suppose w is the nearest node on the right side of v, but not necessarily a neighbor, that knows its position $q \in$ {W, E, N, S}. Let V be all nodes between v and w, including the plain nodes. Now node v and w set the position of nodes in V as follows:

1) If $p=q$, then all nodes in V are assigned the same position p.
2) If $p \neq q$, then one of p or q must be N or S. Let us assume p $\in$ {N, S}. Then, all nodes in V are assigned p. It can be easily verified that if $p \neq q$, then p and q cannot have the values only from {N, S} or only from {E, W}.

Node type	Neighbor node type	Position
X_{ES}	Left node is X_{EN}	West
X_{ES}	Right node is X_{WS}	North
X_{WS}	Left node is X_{ES}	North
X_{WS}	Right node is X_{WN}	East
X_{WN}	Left node is X_{WS}	East
X_{WN}	Right node is X_{EN}	South
X_{EN}	Left node is X_{WN}	South
X_{EN}	Right node is X_{ES}	West
V_{ES}	Left node is V_{EN}	East
V_{ES}	Right node is V_{WS}	South
V_{WS}	Left node is V_{ES}	South
V_{WS}	Right node is V_{WN}	West
V_{WN}	Left node is V_{WS}	West
V_{WN}	Right node is V_{EN}	North
V_{EN}	Left node is V_{WN}	North
V_{EN}	Right node is V_{ES}	East

Table 1. Position of each node on an f-ring.

3. Routing algorithms

In this paper we follow the convention and notation used in [1, 2]. Let *M* denote a message to be routed from the current node, (a_c, b_c), to the destination node (a_d, b_d). Our algorithms are based on the e-cube routing, in which all messages are routed in two phases: in the first phase the message is routed along d_0 dimension (row dimension) until ($a_c = a_d$), and in the second phase it is routed along d_1 dimension (column dimension) to the destination. M is said to be *row message* if it is in its first phase and *column message*, otherwise. Further, row messages traveling from West to East (respectively, East to West) are *WE* (respectively, *EW*) messages. Likewise, column messages traveling from North to South (respectively, South to North) are *NS* (respectively, *SN*) messages. A row message can be changed to column message, but not vice versa in the e-cube routing.

Our first fault-tolerant routing algorithm, *F4*, requires four virtual channels and handles the fault rings with zigzag sides. We also propose another algorithm, *F3*, which, using only three virtual channels, routes a message in a mesh with f-rings containing at most three zigzag sides. There is no limit on the number of faults in both algorithms. Note that neither overlapped f-rings nor f-chains are allowed in our algorithms.

3.1. F4: A routing algorithm for fault rings with four zigzag sides

Let (a_c,b_c) be the current host node of the given message M. If $(a_c,b_c) = (a_d, b_d)$, then M has reached its destination and so it is consumed. Otherwise, M starts as either WE or EW message depending on its value of d_0 dimension in source and destination addresses. Once ($a_c = a_d$), M changes to column message and continues to travel towards its destination. M travels in the network based on the e-cube algorithm until it reaches an f-ring. Once it reaches the f-ring, two things can happen:

1. If its e-cube hop is on the f-ring, then it continues to travel in the f-ring, or
2. If its e-cube hop is blocked, then it is misrouted.

In both cases, the message traveling on an f-rings can use only the following virtual channels depending on the message type: WE message uses c_0, EW message uses c_1, NS message uses c_2, and *SN* message uses c_3 channel. The next hop for M is determined by the set of rules given in Table 2. The *Procedure Set-Status* sets the message status to *normal* or *misrouted* depending on the availability of the next e-cube hop. That is, if the next e-cube hop for M is not blocked by a fault, then its status is set to normal, and M is routed using the e-cube algorithm. Otherwise it is misrouted.

If a message takes an e-cube hop on a link that is not on an f-ring, it can use any virtual channel without causing deadlock. This is because the e-cube routing needs only one virtual channel per physical channel [2].

Whenever M's status becomes misrouted, its direction on the f-ring is determined by the *Procedure Set-Direction* using Table 3. Once its direction on the current f-ring is set, it will never be changed until *M* leaves the f-ring. Table 3 is established by the following rules in order to avoid deadlock.

1. When a WE message reaches the West side of an f-ring, route the message in counter clockwise direction if the destination is to the South of current node; otherwise route in clockwise direction. Note that the WE message never reaches the East side of the f-ring.
2. When a WE message reaches the North (resp., South) side of an f-ring, route the message clockwise (resp., counter clockwise) along the f-ring.
3. When a NS message reaches the North side of an f-ring, route the message in either direction.
4. When a NS message reaches the West side (resp., East) of an f-ring, route the message in counter clockwise (resp., clockwise) direction on the f-ring.

Similar rules are applied for EW and SN messages and are summarized in Table 3.

Procedure Set-Direction (M)

IF direction of M is not null, return.
Otherwise, set direction according to Table 3.

Procedure Set-Status (M)

IF (M is row message, and the next available e-cue hop is not on the f-ring), THEN set the status of M to normal and the direction of M to null.
IF (M is NS message, $a_c = a_d$, $b_c > b_d$, and the next e-cube hop is available), THEN set the status of M to normal and the direction of M to null.
IF (M is SN message, $a_c = a_d$, $b_c < b_d$, and the next e-cube hop is available), THEN set the status of M to normal and the direction of M to null.
Otherwise, set the status of M to misrouted and set-Direction (M).

Procedure Route-Message (M)

1 IF $(a_c = a_d)$ and $(b_c = b_d)$, THEN M reaches its destination. Consume M and return.
2 IF M is a row message and $(a_c = a_d)$, THEN change its type to SN if $(b_d > b_c)$ or NS if $(b_c > b_d)$.
3 SET-STATUS (M).
4 IF M is normal, use the e-cube hop.
Otherwise, route M on the fault ring in the specified direction.

Table 2. Fault-tolerant algorithm, F4**.**

Node position	Message Type	Misrouted Direction
North	WE	Clockwise (CW)
	EW	Counter Clockwise (CCW)
	NS	Either CW or CCW
	SN	CW if $a_c < a_d$, CCW if $a_c > a_d$
South	WE	CCW
	EW	CW
	NS	CW if $a_c > a_d$, CCW if $a_c < a_d$
	SN	Either CW or CCW
West	WE	CCW if $b_c > b_d$, CW if $b_c < b_d$, otherwise either CW or CCW
	EW	N/A
	NS	CCW
	SN	CW
East	WE	N/A
	EW	CW if $b_c > b_d$, CCW if $b_c < b_d$, otherwise either CW or CCW
	NS	CW
	SN	CCW

Table 3. Direction to be set for the misrouted messages on f-rings.

EXAMPLE 2: Consider the F4 routing of a message M from (1, 7) to (5, 0) in Fig. 3. M uses c_0 channel from (2, 7) to (4, 5), and uses c_2 from (5, 3) to (5, 1). At first, M is a normal WE message and is routed to (3, 7). Since its e-cube hop is blocked by the faulty link (3, 7)↔(4, 7), its status is set to *misrouted* at (3, 7). Because M's destination is south of (3, 7), its direction is set to the counter clockwise orientation and is routed along (3, 6)→(2, 6)→(2, 5). At (2, 5) M becomes normal message again and travels up to (5, 5). *M* becomes a column (NS) message at (5, 5), and travels to the south along (5, 5)→(5, 4)→(5, 3) as a normal message. At (5, 3) M's e-cube hop is blocked by the faulty link (5, 3) ↔ (5, 2). Thus, its status is set to *misrouted* at (5, 3) and misrouted in the counter clockwise orientation (the orientation is chosen randomly) along the f-ring to (5, 1). At (5, 1) M is set to a normal NS message, since the destination and current node are located on the same column and its e-cube hop is not blocked. M is routed to (5, 1)→(5, 0) as a normal NS message. Note that the hops from (5, 1) to (5, 0), (4, 5) to (5, 5), (5, 5) to (5, 4), and (5, 4) to (5, 3) are not on the f-ring. Therefore, as mentioned above, any of the four classes of virtual channels may be used for the hop.

We now prove the deadlock freedom in F4.

Lemma 1: Let m_1, m_2 and m_3 be the messages using the same virtual channels. Also let n_1, n_2 and n_3 be the nodes in f-rings f_1, f_2 and f_3, respectively. If m_1 travels from n_1 to n_2, and m_2 travels from n_2 to n_3, then there is a message m_3 that travels from n_1 to n_3.

Lemma 1 implies that any deadlock involving multiple f-rings can be checked by just considering between two f-rings.

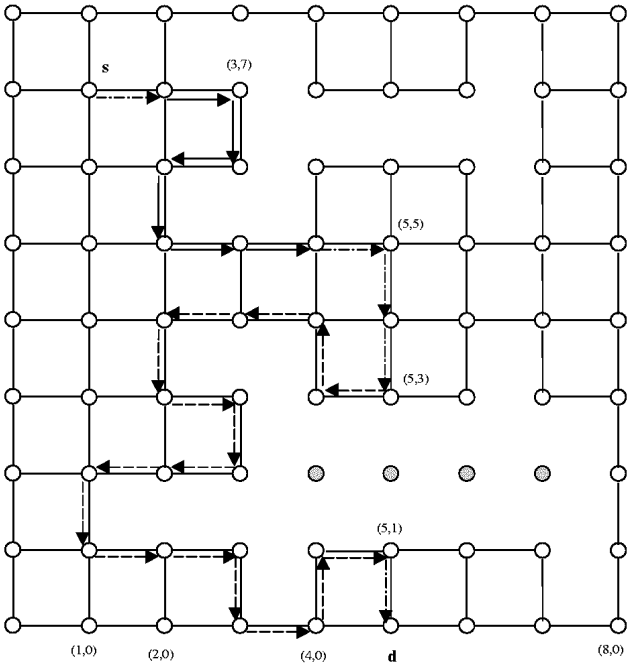

Figure 3. An example of F4 using four virtual channels. Solid arrows indicate the virtual channel c_0, and the dotted arrows indicate the virtual channel c_2. The other type of arrow is used for the e-cube hops that are not on the f-ring, and it can be any virtual channel.

Theorem 1: *F4 algorithm causes no deadlock in 2D meshes that contain any number of f-rings with zigzag sides.*

Proof: In F4 each message type uses an exclusive set of virtual channels for its travel to the destination. Row messages may change their types into column messages, but not vice versa. Further, WE messages cannot be changed into EW messages, and vice versa. Likewise, NS messages cannot be changed into SN messages, and vice versa. Thus, deadlock involving more than one type of messages cannot occur. Hence, to prove the deadlock freedom, it is sufficient to show that there is no deadlock among messages of a specific type.

In the following we provide the proof of deadlock freedom for WE messages (c_0 channels) only. Deadlock freedom for other message types can be proved in a similar manner. Our proof consists of two parts: no cyclic dependency among WE messages 1) in an f-ring, and 2) involving multiple f-rings.

1) Deadlock freedom with WE messages in an f-ring: According to F4, WE messages can travel either clockwise or counterclockwise on f-rings, and it can be easily shown that these two types of messages do not share the same link. Thus, no deadlock can occur in an f-ring.
2) Deadlock freedom with WE messages involving multiple f-rings: Suppose a WE message travels along the part, say p_1, of f_1 and reaches at the node b on f_2. In order to form a cycle, there must be a WE message that travels through b and reaches p_1. We prove the theorem by showing the non-existence of such a message. In the following we consider all possible cases that can occur with WE messages.
Case 1) Node b is at the North or South side of f_2.
According to F4, the message will never take EW channels. Therefore, it never reaches p_1. (It may reach other parts of f_1 again if any part of f_2 locates in the south concave section or the north concave section of f_1, but not p_1).
Case 2) Node b is at the West side of f_2. If the West side contains no concave section, a similar argument as in Case 1 above can be applied. If f_2 does contain concave sections, the WE message may travel clockwise (resp., counter clockwise) taking some EW channels along f_2 until it reaches west-most node, say d, of North side (resp., South side). Note that since overlapping of fault rings are not allowed, the path from b to d cannot reach p_1. After reaching node d, Case 1 can be applied.
Also, F4 does not allow WE messages to reverse its direction on a column.
Similar argument can be applied to EW, NS and SN messages. Thus, there is no deadlock in F4.

3.2. F3: A routing algorithm for f-rings with up to three zigzag sides using three virtual channels

To implement the virtual channels, extra hardware such as flit buffers and proper logic is needed. Besides, these virtual channels are usually implemented on physical channel by using time sharing strategy. Therefore, a less number of virtual channels may lead to a better performance of the system.

In this section we propose another fault tolerant routing algorithm for f-rings with at most three zigzag sides. To satisfy this property, we make either all West sides or all East sides of f-rings be of convex fault model. In the previous routing algorithm, F4, it can be observed that only half of the c_0 (respectively, c_1) virtual channel links in f-rings is used by WE (respectively, EW) messages. For full utilization of the channel capacity, the WE and the EW messages share the same virtual channel, c_0, in our modified algorithm, F3. Consequently, the channel usage by F3 is: 1) the WE and EW messages use c_0 channel, 2) the NS messages use c_1, and 3) the SN messages use c_2 channel. Except these channel assignment to message types, F3 is same as F4.

Theorem 2*: F3 causes no deadlock in 2D meshes.*

Proof: The same argument can be applied as shown in Theorem 1 for deadlock in an f-ring. Suppose two f-rings, f_1 and f_2 are involved in deadlock. Without loss of generality, assume that the location of f_1 is left of f_2. It can be seen that the only possible palces for deadlock between the two f-rings are East side of f_2 and West side of f_2. Let us assume that East sides of f-rings are convex. To satisfy the deadlock situation, the messages involved in the deadlock should have four or six "turns" in their paths [5]. However, since East side is concave, it is not possible to have "up-right" and "down-right" turns. Therefore, no deadlock occurs in F3.

In both F3 and F4, messages are misrouted only when they are blocked by f-rings. Otherwise, they proceed towards their desinations using the e-cube algorithm. Since there is a limited number of f-rings in the network, the amount of misrouted steps are also limited. Therefore, every message will eventually reach its destination without any livelock.

4. Conclusion

In this paper we have proposed two fault-tolerant wormhole routing algorithms in mesh networks. Both algorithms relax the restrictions in [1, 2, 3] to some extend. As a result, the number of components to be made disabled would be reduced considerably in some cases. First algorithm, F4, using four virtual channels, can tolerate any number of convex and concave portions in each side. Second algorithm, F3, uses only three virtual channels to tolerate fault rings with up to three zigzag sides. Both F3 and F4 are free of deadlock and livelock and guarantee the delivery of messages between any pair of nonfaulty and connected nodes in the network.

5. References

[1] R.V. Boppana and S. Chalasani, "Fault-Tolerant Wormhole Routing Algorithms for Mesh Networks," *IEEE Trans. Computers*, vol. 44, no. 7, pp. 848-864, July 1995.

[2] S. Chalasani and R.V. Boppana, "Communication in Multicomputers with Nonconvex Faults," *IEEE Trans. Computers*, vol. 46, no. 5, pp. 616-622, May 1997.

[3] P. Sui and S. Wang, "An Improved Algorithm for Fault-Tolerant Wormhole Routing in Meshes," *IEEE Trans. Computers*, vol. 46, no. 9, pp. 1040-1042, Sept. 1997.

[4] A. Chien and J. Kim, "Planar-Adaptive Routing: Low-Cost Adaptive Networks for Multiprocessors," *Proc., 19th Ann., Int'lSymp. Computer Architecture*, pp. 268-277, 1992.

[5] C. J. Glass and L. M. Ni, "The turn model for adaptive routing", Proceedings of the 19th International Symposium on Computer Architecture, pp. 278-287, May 1992.