

Fault-Tolerant Routing Algorithm for Meshes Without Using Virtual Channels

KUO-HSUAN CHEN AND GE-MING CHIU
*Department of Electrical Engineering and Technology
National Taiwan University of Science and Technology
Taipei, Taiwan 106, R.O.C.
E-mail: chiu@optimal.ee.ntit.edu.tw*

We present a fault-tolerant routing algorithm which requires no virtual channels for mesh networks. Our method employs the concepts of fault rings and fault chains, which were previously used with virtual channels, to facilitate fault-tolerant routing. Typically, the number of faults that can be tolerated in a mesh without virtual channels is very small, or the number of nonfaulty nodes that must be disabled is large. The method tolerates any number of faults without disabling a large number of nonfaulty nodes. The proposed algorithm tolerates any number of faults without disabling a large number of nonfaulty nodes. Moreover, only the nodes on fault rings and fault chains need to maintain a small amount of routing information. The algorithm avoids the formation of the rightmost column segment of a circular waiting path to ensure the property of deadlock freedom. Simulations have been conducted to evaluate the performance of our algorithm without virtual channels.

Keywords: deadlock-free, fault tolerance, routing, virtual channel, wormhole routing.

1. INTRODUCTION

The mesh topology has become a popular interconnection architecture for constructing massively parallel multiprocessors. Low-dimensional meshes have been used in several commercial machines. Processors (or nodes) of a mesh communicate with each other by sending messages through the network. Hence, efficient communication is critical to the performance of a mesh network. As the number and complexity of processors are increasing, so is the probability that some of the nodes may fail in a mesh. Hence, it is essential to design a fault-tolerant algorithm that can route messages in the presence of faulty components.

Recently, the most popular switching technique has been *wormhole routing* [6]. With wormhole routing, a message is divided into *flow control digits* (or *flits*). The flits are routed through the network one after another in a pipeline fashion. A flit of a message is designated as the *header flit*, which leads the message through the network. When the header flit is blocked due to a lack of output channels, all of the flits wait at their current nodes for available channels. Since more than one channel may be reserved for exclusive use during routing of a message, the order in which channels are reserved for a message must follow certain criteria so that deadlock is avoided.

Received May 2, 1997; accepted February 27, 1998.
Communicated by Jang-Ping Sheu.

A number of fault-tolerant routing algorithms have been proposed for meshes using wormhole routing. They can be generally classified into two categories: methods with virtual channels added, and methods without virtual channels. In [7], virtual channels were first introduced to assist the design of deadlock-free routing algorithms. Fault tolerance was not specifically addressed. Linder and Harden [11] used virtual channels to design fault-tolerant routing algorithms for meshes. However, their method requires a large number of virtual channels for only limited fault-tolerance capability. Chien and Kim [4] presented a partially adaptive algorithm to handle block faults in meshes. Three virtual channels per physical channel are required in their method. Boppana and Suresh [1] introduced the concept of *fault rings* to assist fault-tolerant routing in meshes. A message blocked by a faulty region advances by traveling through the associated fault ring along some specified direction. Four virtual channels per physical channel are used to avoid deadlock. Boura and Das [2] proposed an adaptive deadlock-free fault-tolerant routing algorithm for meshes. A node labeling technique is used to identify nodes that may cause routing difficulty. Messages are routed adaptively in healthy regions. This method can tolerate any number of faults by using three virtual channels per physical channel.

However, adding virtual channels to meshes is not free. It involves adding buffer space and complex control logic to the routers. It has been reported [3] that routers based on virtual channels require 2 to 3 times as many gates as those without virtual channels. In addition, the setup delays for routers with virtual channels are 1 to 2 times as long as those without virtual channels, and the flow control cycles are 1.5 to 2 times as long. Moreover, the addition of extra logic circuits and buffer space makes routers that use virtual channels more liable to failure and, thus, less reliable. This is undesirable from the fault-tolerant routing point of view addressed in this paper. In addition, deadlock-free routing schemes that are not based on using virtual channels may be used as the basic mechanisms for implementing adaptive routing schemes with virtual channels [5].

Recently, researchers have attempted to design fault-tolerant routing algorithms that require no virtual channels for a mesh. The algorithm proposed by Glass and Ni [9] is based on modification of the turn model [8]. It can tolerate up to $n - 1$ faults in an n -dimensional mesh. However, there are many fault patterns with n or more faults for which this algorithm is unable to route. Furthermore, in addition, it may occur that a packet may be blocked from advancing by a single fault in a two-dimensional mesh and must be discarded as a result. Libeskind-Hadas and Brandt [10] proposed an origin-based fault-tolerant routing algorithm for meshes. Their algorithm also routes messages without using virtual channels. The channels of the network are partitioned into two subnetworks called the *IN subnetwork* and *OUT subnetwork*, which are defined with respect to a chosen origin node. A message is first routed using channels in the *IN subnetwork*; the channels in *OUT subnetwork* are used when the message reaches the *outbox*. For fault tolerance purposes, some nonfaulty nodes are disabled so that all faulty regions are square. Each nonfaulty node must contain the distances to the nearest fault in each direction.

In this paper, we propose a fault-tolerant routing algorithm which requires no virtual channel and routes messages successfully as long as the network is not partitioned by faulty regions. Our method employs the concepts of fault rings/chains, which were used with virtual channels in [1]. Motivated by the turn model-based

routing scheme [9], our algorithm restricts the traveling directions of messages to ensure the property of deadlock freeness. The key to our algorithm is that the rightmost column segment of a circular waiting path, which is essential for a deadlock state, can never be formed; thus, deadlock is prevented. In general, the number of faults that can be tolerated in a mesh without virtual channels is small [9], or the number of nonfaulty nodes that must be disabled is large [10]. Our method tolerates any number of faults without disabling a large number of nonfaulty nodes. Moreover, the routing information that must be kept by each nonfaulty node is small. Only the nodes on fault rings/chains need to maintain this information. Simulations have been conducted to study the performance of our algorithm.

In the next section, necessary notations and definitions are given. Our fault-tolerant routing algorithm for two-dimensional meshes is presented in Section 3. Simulation results for two-dimensional meshes are shown in Section 4. Section 5 addresses the issue of extending the algorithm to higher dimensional meshes. Section 6 gives conclusions.

2. PRELIMINARIES

An n -dimensional mesh has $K_0 \times K_1 \times \dots \times K_{n-1}$ nodes, where n is the number of dimensions of the network, and K_i is the radix of dimension i , $0 \leq i \leq n-1$. A node X is identified by an n -element vector $(x_0, x_1, \dots, x_{n-1})$, where $0 \leq x_i \leq K_i-1$. Two nodes $X = (x_0, x_1, \dots, x_{n-1})$ and $Y = (y_0, y_1, \dots, y_{n-1})$ are connected in dimension i if and only if $x_i = y_i \pm 1$, and $x_j = y_j$ for all $j \neq i$. Two neighboring nodes are connected by a bidirectional channel, which consists of two unidirectional physical channels. No virtual channel is required in our mechanism. In this paper, we assume that faults are nonmalicious; i.e., a failed node simply ceases to work. Both the source and the destination nodes of a message are nonfaulty.

3. ROUTING IN TWO-DIMENSIONAL MESHES

We label the four sides of a two-dimensional (2D) mesh as East, West, South, and North. All of the nodes that have the same coordinates of dimension 0 constitute a *column*, and all of the nodes that have the same coordinates of dimension 1 constitute a *row*. In order to achieve deadlock-free routing in a faulty mesh without using virtual channels, we have to deactivate some of the nonfaulty nodes that may cause routing difficulty. We use a node labeling technique to convert faulty regions into regions rectangular in shape to facilitate message routing. A similar concept was adopted in [2].

Definition 1: A nonfaulty node X is defined as a deactivated node if X has two or more faulty or deactivated nodes.

A nonfaulty node that is not deactivated is called an *active* node. Deactivated nodes can be readily identified in a recursive manner. The set of deactivated and faulty nodes form one or more rectangular areas. Each such rectangular areas is

called a *faulty region*. Obviously, deactivated nodes are not desirable. To reduce the number of disabled nodes, a deactivated node is labeled as *unsafe* if it has at least one active neighboring node. Unsafe nodes can send or receive messages although they are not used as intermediate nodes for routing messages. For example, Fig. 1 shows a 10×10 mesh with 17 faulty nodes. Six faulty regions are identified in the figure. A message must be navigated around faulty regions in order to reach its destination. In the following, it is assumed that faulty regions do not partition the network. We use the concept of *fault rings* [1] to assist routing of messages around faulty regions. A fault ring is a set of active nodes that exactly enclose a faulty region. Fault rings can be easily formed in two steps as shown in Fig. 2. The procedure Form-Ring is similar to the one given in [1]. Here we use the notations E_X , W_X , S_X , and N_X to represent the neighbors of node X to the east, the west, the south, and the north, respectively. In the first step, an active node that is connected to an unsafe, deactivated, or faulty neighbor marks itself as a member of a fault ring. It then sends its status information to appropriate active neighbors. Based on the status information received from its neighbors, an active node may determine whether it is a corner node of a fault ring according to Table 1. The northeast corner node of a fault ring is called the *reference node* of the ring. The reference node of a fault ring circulates its coordinate around the ring; each node on the fault ring must record this coordinate to facilitate fault-tolerant routing. Forming a fault ring around a faulty region is impossible when the region touches one or more

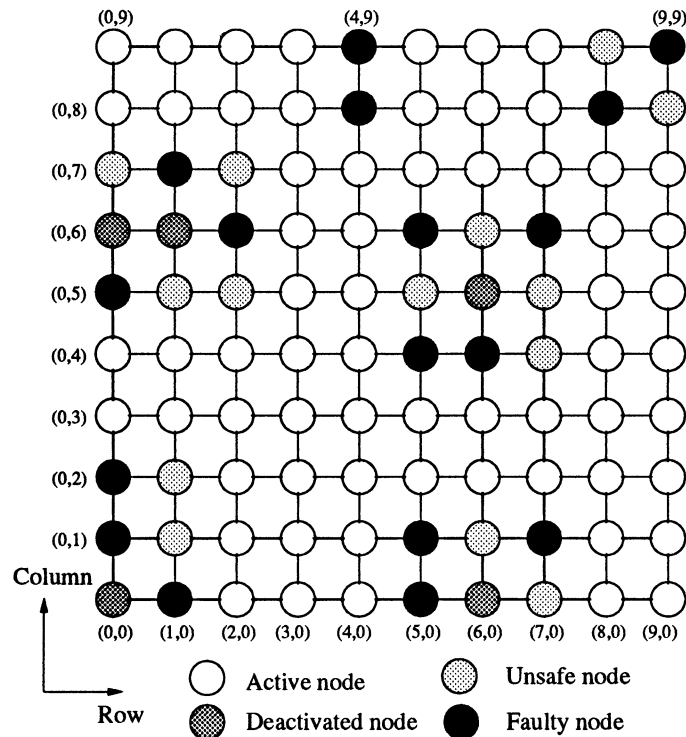


Fig. 1. An example of a faulty mesh.

```

Procedure Form-Ring /* current node is X */
  if (EX or WX is faulty, unsafe, or deactivated)
    send status information to SX and NX;
  if (SX and NX is faulty, unsafe, or deactivated)
    send status information to EX or WX;
  receive from all active neighbors their status information;
  determine whether X is corner nodes of fault rings according to Table 1;
    
```

Fig. 2. Procedure used to form fault rings.

Table 1. Table used to determine corner nodes of fault rings; X is the current node.

corner position	status information
northeast corner	both S_{W_X} and W_{S_X} are faulty, unsafe, or deactivated
northwest corner	both S_{E_X} and E_{S_X} are faulty, unsafe, or deactivated
southeast corner	both N_{W_X} and W_{N_X} are faulty, unsafe, or deactivated
southwest corner	both N_{E_X} and E_{N_X} are faulty, unsafe, or deactivated

boundaries of the mesh. In this case, a *fault string*, rather than a fault ring, is established around the faulty region. Fault strings that touch the East or North boundary are treated as fault rings during routing. However, the corresponding reference nodes for this type of string are given as follows. If a fault string touches the East boundary of the mesh, the coordinate of its reference node is set to (*don't care*, -1); otherwise it is set to (*don't care*, K_1). These reference nodes are nonexisting pseudo nodes. As described later, the dimension-1 coordinates of the nodes are given to facilitate routing on the fault strings. The fault strings that touch only the South boundary, West boundary, or both are used slightly differently from fault rings. These fault strings are called *fault chains* while fault rings may refer to real fault rings or the previous type of fault string. A fault chain that touches only the South boundary of the mesh is called an *s-chain*. Fig. 3 illustrates the eight types of fault strings in a 2D mesh. A node on a fault chain need not maintain the coordinate of the reference node. It is easy to see that an active node can fall on at most two distinct fault rings or fault chains in a 2D mesh; if it does, it must be a corner node for at least one of the two fault rings/chains. As a result, two fault rings/chains can share at most one bidirectional channel; that is, two rings/chains can overlap at most one bidirectional channel. For example, the faulty mesh in Fig. 1 is redrawn in Fig. 4 with the associated fault rings and chains indicated. Our fault-tolerant routing algorithm is based on forwarding of messages around faulty regions along the associated fault rings and fault chains in an appropriate manner so that deadlock is prevented. In the following, *row channels* refer to channels in dimension 0, and *column channels* refer to channels in dimension 1. A message is one of three message types at any time, and its type may change during routing.

Definition 2: (*Message type*) A message is called a *row-first* (RF) message if its destination, D , is located to the West of its source node, S ; that is, the coordinate of D in dimension 0 is smaller than that of S . The message is called a *column-first* (CF)

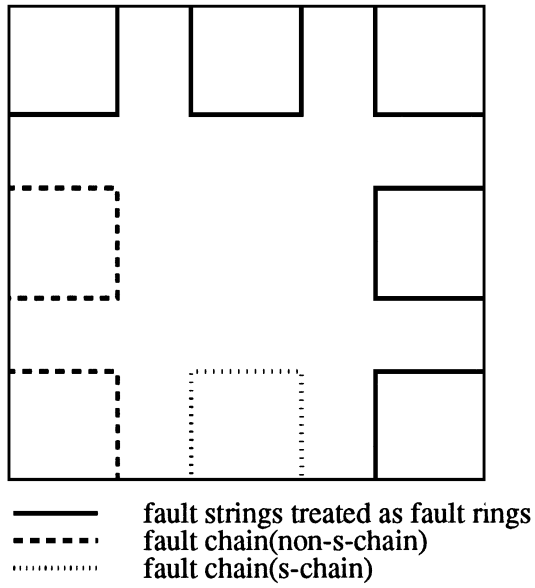


Fig. 3. Eight types of possible fault strings in a 2D mesh.

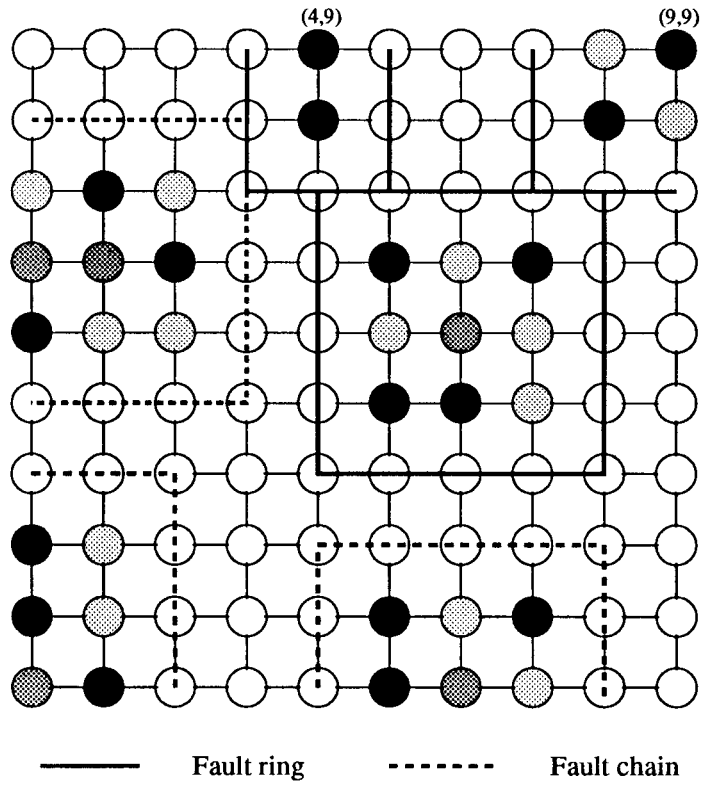


Fig. 4. Example of fault rings and fault chains.

message if it is not an RF message and S and D are not located in the same row; it is called a row-only (RO) message if S and D are located in the same row.

The message types are illustrated in Fig. 5. The basic idea of our routing algorithm is to forward a message along row channels or column channels first depending on whether it is an RF or a CF message. As shown in Fig. 5(a), an RF message may eventually change itself into a CF message when it reaches a node that is located in the same column as D . However, a CF message never changes itself into an RF message. Furthermore, as shown in Fig. 5(b), a CF message becomes an RO message when it reaches a node that is in the same row as its destination. An RO message never changes its type. In the following treatment, CF messages which intend to travel from north to south (respectively, south to north) are NS (respectively, SN) messages. A row channel is an EW (respectively, WE) channel if it travels from east to west (respectively, west to east). We can define SN and NS column channels similarly.

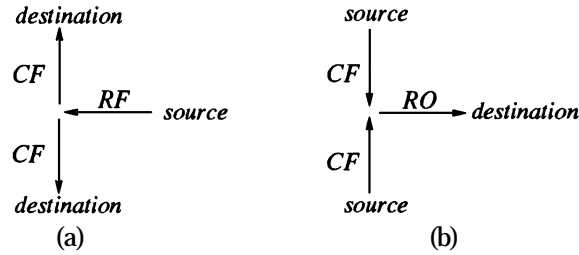


Fig. 5. Message types.

3.1 Routing Algorithm

Consider a message m_g that is sent from source node S to destination node D . Let C denote the current node of the header flit of m_g . Our fault-tolerant routing algorithm is presented in the following. The procedure Message-Route is the main function of our routing algorithm.

```

Procedure Message-Route
/* Message  $m_g$  is sent from source  $S$  to destination  $D$ ;  $C$  is the current node of header flit */
if ( $C$  is the destination  $D$ )
    Consume  $m_g$ ;
else
    if (current node  $C$  is  $S$ , and is unsafe)
        Send  $m_g$  to an active neighbor;
    else/* active node */
        begin
            Determine the message type (RF, CF, or RO) of  $m_g$ ;
            if ( $C$  is not on a fault ring or fault chain)
                Normal-Route( $m_g$ );
        end
    end

```

```

else
  if ( $C$  is on a fault ring)
    Ring-Route( $m_g$ )
  else
    Chain-Route( $m_g$ )
end

```

Procedure Normal-Route(m_g)

```

switch ( $m_g$ 's type)
case (RF message)
  Use EW channel to forward  $m_g$ ;
  exit;
case (CF message)
  if ( $m_g$  is NS message)
    Use NS channel to forward  $m_g$ ;
  else Use SN channel to forward  $m_g$ ;
  exit;
case (RO message)
  Use WE channel to forward  $m_g$ ;
  exit;

```

Procedure Ring-Route(m_g)

```

switch ( $m_g$ 's type)
case (RF message)
  if (EW channel is available)
    Use EW channel to forward  $m_g$ ;
  else
    Route  $m_g$  clockwise;
  exit;
case (CF message)
  if ( $m_g$  is SN message)
    if ( $C$  is on the North boundary of the fault ring)
      Normal-Route( $m_g$ );
    else
      if ( $C$  is on the west boundary of the fault ring and  $D$  is in the same column as  $C$ )
        Normal-Route( $m_g$ );
      else
        if ( $D$  is lower than the reference node of the ring)
          Route  $m_g$  counter-clockwise;
        else
          Route  $m_g$  clockwise;
    else /* NS message*/
    if ( $C$  is on the East or South boundary of the fault ring)
      Normal-Route( $m_g$ );
    else
      if ( $C$  (including  $S$ ) is on the West boundary of the ring and EW channel is
        available)

```

```

        Route  $m_g$  along EW channel;
    else
        Route  $m_g$  counter-clockwise;
    exit;
case (RO message)
    if ( $C$  is in the same row as  $D$  and WE channel is available)
        Use WE channel to route  $m_g$ ;
    else
        Route  $m_g$  counter-clockwise;
    exit;

```

Procedure Chain-Route(m_g)

```

switch ( $m_g$ 's type)
case (RF message)
    if (the fault chain is an s-chain)
        if (EW channel is available)
            Route  $m_g$  along EW channel;
        else
            Route  $m_g$  counter-clockwise;
    else /* not s-chain */
        if ( $C$  is in the same row as  $D$ )
            Route  $m_g$  along EW channel;
        else
            if ( $D$  is higher than  $C$ )
                Route  $m_g$  counter-clockwise;
            else
                Route  $m_g$  along clockwise direction;
    exit;
case (CF message)
    if ( $m_g$  is an NS message)
        if (the chain is an s-chain)
            Route  $m_g$  clockwise;
        else /* not s-chain */
            if (NS channel is available and  $D$  is not to the west of  $C$ )
                Route  $m_g$  along NS channel;
            else
                Route  $m_g$  clockwise;
    else /* SN message */
        if (SN channel is available and  $D$  is not to the west of  $C$ )
            Route  $m_g$  along SN channel;
        else
            Route  $m_g$  counter-clockwise;
    exit;
case (RO message)
    Route  $m_g$  clockwise;

```

If the current node is the destination, m_g is consumed. If source S is unsafe, m_g is first forwarded to an active neighbor on a fault ring or a fault chain. The header flit of each message includes a parameter which indicates the type (RF, CF, or RO) of the message. If the current node C is active and is not on any fault ring or fault chain, routing is governed by the procedure Normal-Route. During normal routing, if m_g is an RF message, it would first use an EW channel to destination D . It changes its type to CF when it reaches a node in the same column as D . Message m_g is forwarded along column channels (NS or SN) first if it is a CF message until it reaches the same row as D , where it becomes an RO message. An RO message is routed along WE channels as long as such channels are available.

When m_g encounters a faulty region, it is misrouted by traveling along the fault ring or fault chain that is associated with the faulty region. Either the procedure Ring-Route or Chain-Route is used to handle routing on a fault ring or a fault chain, respectively. Consider misrouting on a fault ring. If m_g is an RF message, it is forwarded along an EW channel if such a channel is available; otherwise it is routed clockwise around the ring. If m_g is a CF message, special care must be taken to avoid deadlock. If m_g is an SN message, the relative orientation between the reference node of the fault ring and the destination D is normally used to determine the direction of routing. If D is lower than the reference node of the fault ring, the message is routed counter-clockwise; otherwise it is routed clockwise. However, when this message reaches (or starts from) the North boundary of the ring, normal routing is used. For an NS message, normal routing is followed if current node C (including S) is on either the East or South boundary of the ring. However, there is an exception when an NS message reaches (or starts from) the West boundary of the fault ring. In this case, the message must be routed one hop to the west as long as such a channel is available. Such action is required to make our algorithm deadlock-free. Now consider the case in which m_g is an RO message. It simply travels on the fault ring counter-clockwise until it returns to the same row (i.e. the same height) as D and then moves forward to approach D . Intuitively, the traveling direction of a message on a fault ring is determined such that the rightmost column segment of a circular waiting path, which is essential for a deadlock state, can never be formed. Note that the procedure Ring-Route also applies to the class of fault rings that are actually fault strings with pseudo reference nodes. Recall that the dimension-1 coordinates of the reference nodes are set to either -1 or K_1 for the fault strings. These settings ensure that m_g will travel along a correct direction around the strings if it is an SN message.

The procedure Chain-Route is used to govern routing on a fault chain. Recall that a fault chain is called an s-chain if it touches only the South boundary of the mesh. For an RF message, the traveling direction on the fault chain is determined as follows. If the fault chain is an s-chain, then m_g is always routed along EW channels as long as such channels exist; otherwise, it is forwarded counter-clockwise. If the chain is not an s-chain, the direction of traveling is set to be counter-clockwise (respectively, clockwise) if the destination D is higher (respectively, lower) than the current node. Consider the case in which m_g is a CF message. For an NS message, it is routed clockwise if the chain is an s-chain. However, if the chain is not an s-chain, our algorithm requires that the message travel around the chain until it returns to a point which is at least as westward as D before it is

allowed to leave the chain. A similar requirement is also demanded for an SN message. If m_g is an RO message, it is routed clockwise on a fault chain.

An active node may fall on two distinct fault rings or fault chains. When a message reaches one such active node, the algorithm needs to determine which fault ring or chain to follow for subsequent routing. This can be easily done by comparing the coordinates of the associated reference nodes. If the message is an EW (respectively, WE, SN, and NS) message, the fault ring is selected whose reference node is to the west (respectively, east, north, and south) of the other. However, an RO message always stays on the same fault ring/chain in this case.

Similar to the turn model-based routing algorithm proposed in [9], our scheme restricts the traveling direction (and, so, the turns) of a message in nonfaulty regions. The algorithm in [9] allows prohibited turns to be taken to avoid a fault only when the fault occurs on the negative boundaries (west or south) of the mesh. Hence, it can only guarantee tolerance of one single fault in a 2D mesh. Moreover, a packet may be blocked from advancing by a single fault and must be discarded. Our method, on the other hand, allows prohibited turns to be taken on any fault ring or fault chain so that a message can navigate around the associated faulty region. The key to our algorithm is that the rightmost column segment of a circular waiting path can never be formed; thus, deadlock is prevented. Therefore, more faults can be tolerated by our scheme.

Consider the previous example which is redrawn in Fig. 6. Suppose that a message, m_1 , is sent from source $S_1 = (9, 1)$ to destination $D_1 = (7, 8)$. The message

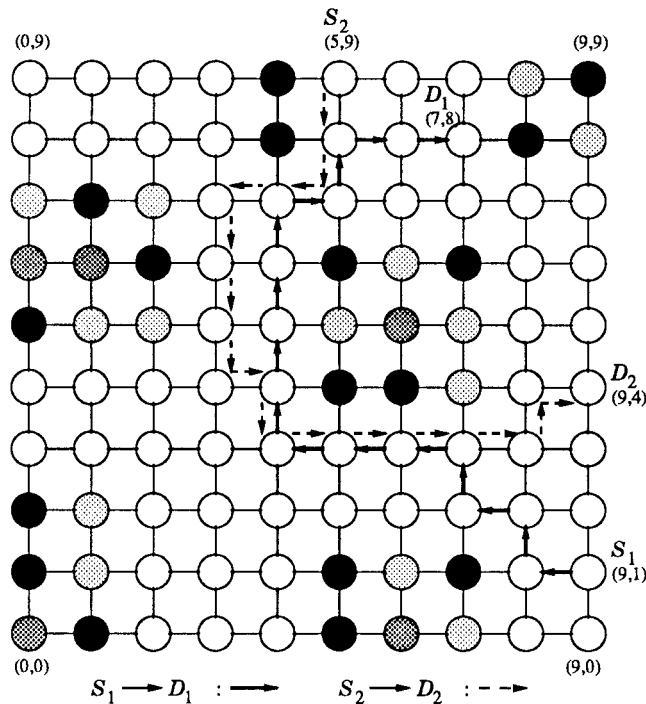


Fig. 6. Example of fault-tolerant routing in a 2D mesh.

starts as an RF message since D_1 is to the west of S_1 . Since S_1 is active, m_1 is forwarded to node (8, 1), which falls on an s-chain. Hence, m_1 is routed counter-clockwise. It becomes a CF message when it reaches node (7, 2), which is on the same column as D_1 . After moving one hop to the North, m_1 encounters a fault ring at (7, 3). Since D_1 is no lower than the reference node (8, 7), a clockwise direction is followed. Eventually, m_1 reaches node (4, 7), which is the northwest corner of the fault ring. Note that this node is also located on another fault string, which encloses a faulty region consisting of nodes (4, 8) and (4, 9). The fault string is treated as a fault ring with a reference node (*don't care*, 10). Since m_1 is an SN message, the algorithm determines that this fault ring is to be followed for subsequent routing. According to Ring-Route, m_1 is routed counter-clockwise on the ring; it becomes an RO message when it reaches node (5, 8), which is located in the same row as D_1 . Traveling two more hops through WE channels, m_1 reaches D_1 . Now consider sending message m_2 from source $S_2 = (5, 9)$ to destination $D_2 = (9, 4)$. m_2 starts as a CF message. Since m_2 is an NS message, it is forwarded to the South. When it reaches node (5, 7), which is located on the North boundary of another fault ring, the algorithm determines that the second fault ring is to be followed subsequently. According to Ring-Route, m_2 is routed counter-clockwise. The next intermediate node (4, 7) is the northwest corner of the fault ring. Hence, m_2 must be routed one hop to the west to node (3, 7), which is located on a fault chain. According to Chain-Route, m_2 is continuously forwarded using NS channels. It becomes an RO message at (3, 4) and eventually reaches D_2 as indicated in Fig. 6.

3.2 Proof of Deadlock-Freeness

Deadlock is caused by messages waiting on each other in a cycle. Consider a set of messages m_1, m_2, \dots, m_l where m_i waits for m_{i+1} for all $1 \leq i \leq l-1$. The sequence of channels which include the channels traversed by m_1 , followed by the channels traversed by m_2 from the node at which m_1 is blocked, is then followed by those traversed by m_3 from the node at which m_2 is blocked, and so on until the current node of m_l is called the *waiting path* of the set of messages. A deadlock is formed if m_l waits for m_1 , as a circular wait is generated by the messages. The associated waiting path for such a deadlocked situation is a circular path; here, the waiting path includes the channels that are traversed by m_1 from the node at which m_l is blocked, rather than from the source of m_1 .

Definition 3: A turn consists of a row channel and a column channel such that the source node of one of the channels is the destination node of the other; the common node of these two channels is called the turning node of the turn.

Essentially, a turn involves a change of traveling direction. There are eight types of turns, defined according to the associated directions. A turn is called an ES turn if it involves a change of direction from east to south. We can define the other seven types of turns similarly. Consider an ES (respectively, EN) turn, T_1 , and an SW (respectively, NW) turn, T_2 , where the column channels of these two turns are located in the same column of the mesh, and the column channel of T_1 is no lower (respectively, no higher) than that of T_2 .

Definition 4: T_1 is said to be aligned with T_2 if there exists a set of messages, m_1, m_2, \dots, m_l , such that m_1 takes T_1 , m_l takes T_2 , m_i waits for m_{i+1} for all $1 \leq i \leq l - 1$, and the column segment that starts from the turning node of T_1 and continues to that of T_2 is a subpath of the waiting path of the messages.

Lemma 1: Using the Message-Route algorithm, an EN turn cannot be aligned with an NW turn as long as the EN turn does not occur at the southeast corner node of a fault chain.

Proof: According to our algorithm, an EN turn can only be made in three cases: 1. an RO message encounters a fault ring, and makes an EN turn at the southeast corner node of the ring; 2. an SN message encounters a fault ring, and the destination of the message is lower than the reference node of the ring; and 3. a message makes an EN turn at the southeast corner node of a fault chain. Obviously we do not need to consider case 3. We will prove the lemma by contradiction. Assume that an EN turn in either case 1 or case 2 is aligned with a NW turn. Let T_1 and T_2 represent the EN turn and the NW turn, respectively. Note that in either case, T_1 turn is made in the southeast corner of a fault ring. By the contradiction assumption, there must exist a set of messages, m_1, m_2, \dots, m_l , such that m_1 takes T_1 , m_l takes T_2 , m_i waits for m_{i+1} for all $1 \leq i \leq l - 1$, and the column segment that starts from the turning node of T_1 and continues to that of T_2 is a subpath of the waiting path of the messages. Apparently, the turning node of T_2 cannot possibly be lower than the reference node of the ring. Hence, the reference node of the ring is on the aforementioned subpath of the waiting path. Therefore, there must exist some message m_i , $1 \leq i \leq l$, such that m_i will be routed through the SN column channel immediately below the reference node of the ring. However, the following arguments show that this is not possible. If m_i is an RF message at the neighboring node immediately below the reference node, it can only move south (i.e., clockwise). Hence, it cannot possibly use the above-mentioned SN channel. Suppose that m_i is an RO or a CF message at that node. In this case, m_i 's destination must be lower than the reference node; thus, m_i again cannot possibly traverse the SN channel. Hence, the existence of m_i is not possible. \square

Lemma 2: Using the Message-Route algorithm, an ES turn cannot be aligned with an SW turn as long as the column segment between the turning nodes of the ES and the SW turn does not include the east boundary of any f-chain.

Proof: According to our algorithm, an ES turn can only be made in two cases: 1. an RO message traveling east encounters a fault ring and makes an ES turn at a node on the West boundary of the ring; 2. a message (of any type) makes an ES turn in the northeast corner node of a fault chain. Note that, if an ES turn is made by a message in the northeast corner node of an f-chain, as described in case 2, and it is aligned with an SW turn; then, it is obvious that the turning node of the SW turn must be no higher than the southeast corner node of the f-chain. The column segment between the turning nodes of the ES and the SW turns includes the east boundary of the f-chain in this case. Therefore, we do not need to consider case 2. Now we will prove the lemma by contradiction. Assume that an ES turn of case

1 is indeed aligned with an SW turn. Let T_1 and T_2 represent the ES turn and the SW turn, respectively. Recall that, in this case, the turning node of T_1 is located on the West boundary of a fault ring. Let R_f denote this fault ring. The turning node of T_1 is neither the northwest corner nor the southwest corner node of R_f . According to the contradiction assumption, there must exist some set of messages, m_1, m_2, \dots, m_l , such that m_1 will take T_1 , m_l will take T_2 , m_i will wait for m_{i+1} for all $1 \leq i \leq l-1$, and the column segment that starts from the turning node of T_1 and continues to that of T_2 is a subpath of the waiting path. Message m_1 is an RO message.

Case (a): m_l is an RO message

Message m_l cannot possibly make an SW turn (T_2) in this case.

Case (b): m_l is an RF message

In this case, an SW turn (T_2) can be made in the southeast corner node of a fault ring or a fault chain after the message encounters the ring or chain. Such a ring cannot possibly be R_f otherwise, T_1 and T_2 will be located on different columns of the mesh. Also, the turning node of T_2 must be lower than the southwest corner node of R_f . This is because two rings or chains can only overlap over at most one channel, and any boundary of a fault ring/chain must include at least two channels. Therefore, there must exist some message m_i , $1 \leq i \leq l$, such that m_i travels through both NS column channels immediately above and below the southwest corner node of R_f . Note that m_1 , which is an RO message, must make an SE turn at this node; thus, m_i cannot be m_1 . The same argument leads to the conclusion that m_i is not an RO message. m_i cannot possibly be an RF message either. This is also because two rings / chains can only overlap over at most one channel. Consider the case in which m_i is a CF message. According to procedure Ring-Route, m_i cannot be an SN message. Hence, m_i must be an NS message since our algorithm requires that an NS message, which either starts from a source node on the West boundary of R_f or encounters the West boundary of R_f during routing, be routed one step toward the west as long as a channel is available. In the case considered, such a channel is available at or above the neighboring node of the southwest corner node of R_f to the north. Therefore, m_i cannot possibly travel through the NS channel immediately above the southwest corner node of R_f . Hence, the existence of m_i is not possible.

Case (c): m_l is a CF message

In this case an SW turn (T_2) can be made by m_l if m_l is an NS message and encounters a fault ring. Apparently, this ring cannot be R_f . Consider two subcases here.

Case (c.1): the turning node of T_2 is no lower than the southwest corner node of R_f

In this case, the column channel of T_2 is located on the West boundary of R_f since our algorithm requires that an NS message, which either starts from a source node on the West boundary of R_f or encounters the West boundary of R_f during routing, be routed one step toward the west as long as a channel is available. In the case considered here, such a channel is available at or above the source node of the column channel of T_2 . Therefore, m_j cannot possibly travel through the NS channel of T_2 . Hence, T_2 cannot be made by m_j .

Case (c.2): the turning node of T_2 is lower than the southwest corner node of R_f .

The same argument as given in Case (b) also applies here. □

Theorem 1: Message-Route algorithm is deadlock-free in a 2D mesh.

Proof: We will prove the theorem by contradiction. Assume that there exists a set of messages m_1, m_2, \dots, m_l that are deadlocked. Thus, the associated waiting path is a circular one. Since 180-degree turns cannot be taken by Message-Route, the waiting path must include both row and column channels. Consider any column segment, denoted as $C S_r$, of the waiting path that is located in the rightmost column; i.e., no other column channels on the path are more eastward than the channels of $C S_r$. If the channels of $C S_r$ are SN (respectively, NS) channels, there is an EN (respectively, ES) turn that is aligned with an NW (respectively, SW) turn via $C S_r$, where both turns are on the waiting path. According to Lemma 1 (respectively, Lemma 2), such alignment cannot exist unless the EN turn (respectively, the ES turn) occurs in the southeast corner (respectively, the northeast corner) of a fault chain. Thus, $C S_r$ must contain the East boundary of a fault chain. However, a circular path cannot possibly be formed in this case as a fault chain touches the South boundary, East boundary, or both of the mesh, and $C S_r$ is the rightmost segment of the waiting path. □

4. SIMULATION EXPERIMENTS

Extensive simulations have been conducted to evaluate the performance of our routing algorithm without virtual channels under different fault patterns. Boura and Das [2] presented a fault-tolerant routing scheme in which a node labeling technique similar to ours was used. However, their algorithm uses three virtual channels per physical channel in order to ensure the property of deadlock freeness. We also compare our algorithm with theirs.

We considered square meshes in the simulation. Mesh sizes simulated were 10×10 and 15×15 . Up to 10% of faults were simulated in the experiments. We assumed that two unidirectional channels existed between two neighboring nodes. All of the physical channels had the same bandwidth of 20 flits/cycle. Each input channel had a buffer the size of a single flit. For the algorithm of [2], each virtual channel had a buffer the size of a single flit. In the simulation, we used a uniform traffic pattern; that is, a processor sent a message to any other active nodes with equal probability. Messages were generated at time intervals chosen from a negative

exponential distribution. Each message was assumed to be 20 flits in length as commonly used in the literature [1, 2]. The processors routed messages in an asynchronous manner. Conflict of requests for an output channel by multiple messages were resolved in a random manner. The performance metric used in our simulation was the average message latency. For each given number of faults, 1000 fault patterns were randomly selected for simulation. For each fault pattern, a simulation was run for a total of 30000 cycles. Performance data were not collected in the first 10000 cycles to allow the system to stabilize.

Fig. 7 plots the average message latencies achieved by our fault-tolerant routing algorithm against traffic rates simulation results of our fault-tolerant routing algorithm under various numbers of faulty nodes in a 10×10 mesh. The offered traffic is given as a percent of the maximum theoretical throughput [9]. Though they are not shown, simulations on a 15×15 mesh showed trends that were very similar to the one plotted in Fig. 7. For comparison purposes, also included in the figure are the performance data obtained using the algorithm proposed in [2] with three virtual channels. In addition to no consideration of overhead, the algorithm in [2] was simulated under 5% and 15% time overhead. The results show that faults, in general, may significantly affect the performance of the routing algorithms. As the number of faulty nodes increases, the network saturates at a lower traffic rate. The degradation of performance can be explained as follows.

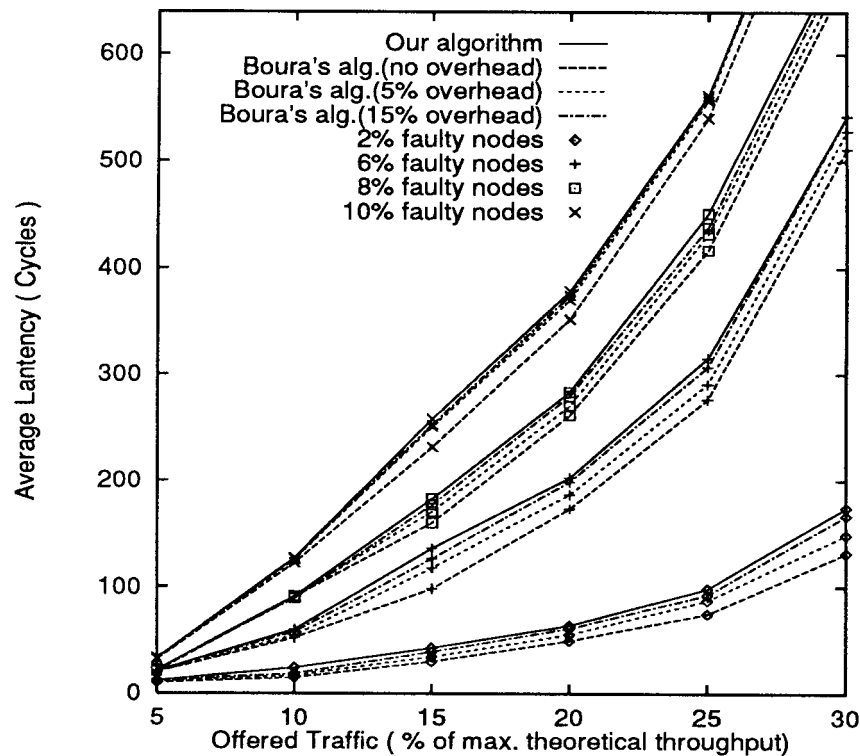


Fig. 7. Average message latencies versus offered traffic.

Contention for channels on fault rings/chains becomes heavier when the number of faulty nodes increases, thus making these channels congested and a source of bottlenecks. While the algorithm in [2] exhibits better performance than ours with the addition of three virtual channels, the improvement diminishes as the number of faulty nodes increases. Moreover, the performance of our algorithm is very close to that of [2] when time overhead. In fact, the performance of our algorithm is almost identical to that of [2] with a mere 5% overhead when the percentage of faulty nodes is 10%. This surely motivates the use of our algorithm for fault-tolerant routing as it requires no virtual channels; thus, the routers are less costly and more reliable.

5. EXTENSION TO HIGHER DIMENSIONAL MESHES

In this section, we will show how our routing algorithm can be extended to higher dimensional meshes with some restrictions imposed. To facilitate presentation, we will describe the algorithm for 3-dimensional (3D) meshes. Let x , y , and z represent dimensions 0, 1, and 2, respectively. We use x^+ and x^- to denote the directions in which the coordinate corresponding to dimension x is increasing and decreasing. The same is defined for dimensions y and z . The deactivation procedure as defined in Definition 1 can be applied, and each faulty region becomes a *faulty block* [1]. Message types RF, CF, and RO are defined in a fashion similar to Definition 2 by replacing the column dimension with the yz -plane. In fact, the basic idea of our algorithm lies in extending the column dimension to the yz -plane. The notion of a fault ring now becomes that of a *fault box*. Each fault box has six 2D planes, which are, respectively, represented by the x^+ -plane, x^- -plane, y^+ -plane, y^- -plane, z^+ -plane, and z^- -plane. The following presentation is restricted to meshes in which faults do not occur on boundaries.

Similar to 2D routing, an RF message traverses x^- channels as long as such channels are available. A CF message with source (x_s, y_s, z_s) and destination (x_d, y_d, z_d) attempts to reach node (x_s, y_d, z_d) before it traverses x^+ channels. On a yz -plane, messages are routed using Message-Route with dimensions y and z replacing dimensions 0 and 1. The fault-tolerant routing algorithm is modified as follows. When an RF message encounters a fault box at its x^+ -plane, it navigates around the box by traveling through the y^- or z^- channels until it reaches either the y^- -plane or z^- -plane. It then uses the x^- channels to move forward. The message becomes a CF message when it reaches a node with the same x coordinate as the destination. It can be routed to the destination using the aforementioned fault-tolerant algorithm. Consider that a message m_g starts as a CF message. The algorithm attempts to route m_g to an intermediate node that has the same y and z coordinates as the destination in the yz -plane where the source is located. Note that such a node may fall in a faulty block and thus be unreachable. The intersection of a fault box and a yz -plane constitutes a fault ring in the plane. If the intermediate node is indeed unreachable, our algorithm for 2D meshes can always lead m_g to the fault ring that is associated with the node. m_g subsequently uses the x^- or x^+ channels to reach either the x^- -plane or x^+ -plane of the associated fault box. The choice of traveling direction is governed by the following rules. If m_g is on the y^+ or z^+ boundary of the fault ring, it is routed along the x^- direction.

Suppose that m_g is on the y or z boundary of the fault ring. It is routed along the x^* direction if the x coordinate of the destination is larger than that of the current node; otherwise, the x direction is followed. As in 2D routing, a message that reaches (or starts from) the x -plane of a fault box must move one step toward the x direction using the x channels as long as such channels are available. When an RO message encounters a fault box at its x -plane, it is routed around the box by traversing either the y or z channels. Though proof is omitted, our algorithm can be shown to be deadlock-free.

5. CONCLUSIONS

In this paper, we have presented a fault-tolerant routing algorithm for a mesh network with faulty nodes. Some nonfaulty nodes are deactivated to construct rectangular faulty regions. A set of criteria is used to govern the traveling direction of a message when the message encounters a faulty region. By avoiding the formation of the rightmost column segment of a circular waiting path, the routing algorithm is deadlock-free without the use of virtual channels.

ACKNOWLEDGMENTS

The authors thank the referees for their insightful comments. We are indebted to Mr. Chun-Long Chen for very knowledgeable discussion. This work was supported in part by the National Science Council of the Republic of China under grant NSC-87-2213-E011-045.

REFERENCES

1. R. V. Boppana and S. Chalasani, "Fault-tolerant wormhole routing algorithms for mesh networks" *IEEE Transactions on Computer*, Vol. 44, No. 7, 1995, pp. 848-864.
2. Y. M. Boura and C. R. Das, "Fault-tolerant routing in mesh networks," in *Proceedings of 1995 International Conference on Parallel Processing*, 1995, pp. I-106-I-109.
3. A. A. Chien, "A cost and speed model for k-ary n-cube wormhole routers," in *Proceedings of Hot Interconnects '93*, 1993.
4. A. A. Chien and J. H. Kim, "Planar-adaptive routing: Low-cost adaptive networks for multiprocessors," *The 19th Annual International Symposium on Computer Architecture*, 1992, pp. 268-277.
5. W. J. Dally and H. Aoki, "Deadlock-free adaptive routing in multicomputer networks using virtual channels," *IEEE Transactions on Parallel and Distributed Systems*, Vol. 4, No. 4, 1993, pp. 466-475.
6. W. J. Dally and C. L. Seitz, "The tours routing chip," *Journal of Distributed Computing*, Vol. 1, No. 3, 1986, pp. 187-196.

7. W. J. Dally and C. L. Seitz, "Deadlock-free message routing in multiprocessor interconnection networks," *IEEE Transactions on Computer*, Vol. 36, No. 5, 1987, pp. 547-553.
8. C. J. Glass and L. M. Ni, "The turn model for adaptive routing," in *Proceedings of 19th Annual International Symposium on Computer Architecture*, 1992, pp. 278-287.
9. C. J. Glass and L. M. Ni, "Fault-tolerant wormhole routing in meshes without virtual channels," *IEEE Transactions of Parallel and Distributed Systems*, Vol. 7, No. 6, 1996, pp. 620-635.
10. R. Libeskind-Hadas and E. Brandt, "Origin-based fault-tolerant routing in the mesh," in *Proceedings of First IEEE Symposium on High-Performance Computer Architecture*, 1995, pp. 102-111.
11. D. H. Linder and J. C. Harden, "An adaptive and fault-tolerant wormhole routing strategies for k-ary n-cubes," *IEEE Transactions on Computer*, Vol. 40, 1991, pp. 2-12.



Kuo-Hsuan Chen (陳國軒) was born on March 28, 1971, in Taipei, Taiwan, the Republic of China. He received the M.S. degree from National Taiwan University of Science and Technology in 1997. His research interests include parallel and distributed systems.



Ge-Ming Chiu (邱舉明) received the B.S. degree from National Cheng-Kung University, Taiwan, in 1976, the M.S. degree from Texas Tech University in 1981, and the Ph.D. degree from the University of Southern California in 1991, all in electrical engineering. He is currently an associate professor in the Department of Electrical Engineering and Technology at National Taiwan University of Science and Technology, Taipei, Taiwan. His research interests include distributed computing, fault-tolerant computing, and parallel processing. Dr. Chiu is a member of the IEEE Computer Society.