Fault-Tolerant Adaptive and Minimal Routing in Mesh-Connected Multicomputers Using Extended Safety Levels

Jie Wu

Department of Computer Science and Engineering

Florida Atlantic University

Boca Raton, FL 33431

jie@cse.fau.edu

Abstract

The minimal routing problem in mesh-connected multicomputers with faulty blocks is studied. 2-dimensional (2-D) meshes are used to illustrate the approach. A sufficient condition for minimal routing in 2-D meshes with faulty blocks is proposed. Unlike many traditional models that assume all the nodes know global fault distribution, our approach is based on the concept of an *extended safety level* which is a special form of *limited fault information*. The extended safety level information is captured by a vector associated with each node. When the safety level of a node reaches a certain level (or meets certain conditions), a minimal path exists from this node to any nonfaulty nodes in 2-D meshes. Specifically, we study the existence of minimal paths at a given source node, limited distribution of fault information, and minimal routing itself. We propose three fault-tolerant minimal routing algorithms which are adaptive to allow all messages to use any minimal path. We also provide some general ideas to extend our approaches to other low-dimensional mesh-connected multicomputers such as 2-D tori and 3-D meshes. Our approach is the first attempt to address adaptive and minimal routing in 2-D meshes with faulty blocks using limited fault information.

Keywords: Fault tolerance, mesh-connected multicomputers, minimal routing.

1 Introduction

In a multicomputer system a collection of processors (also called nodes) work together to solve large application problems. These nodes communicate and coordinate their efforts by sending and receiving messages through the underlying communication network. Thus, the performance of such a multicomputer system is dependent on the end-to-end cost of communication mechanisms. Routing time of messages is one of the key factors critical to the performance of multicomputers. Basically, routing is the process of transmitting data from the *source* node to the *destination* node in a given system.

Mesh-connected topology is one of the most thoroughly investigated network topologies for multicomputer systems. It is of importance due to its simple structure and its good performance in practice, and is becoming popular for reliable and high-speed communication switching. Mesh-connected topologies, also called *m-ary n-dimensional meshes*, have an n-dimensional grid structure with m nodes in each dimension such that every node is connected to two other nodes in each dimension by a direct communication. Mesh-connected topologies include n-dimensional meshes, tori, and hypercubes. These topologies have desirable properties of regularity, balanced behavior, and a large number of alternative paths. Examples of commercial products based on n-dimensional hypercubes (n-cubes) include the Ncube's nCUBE and the Thinking Machine's Connection Machine which is a hypercube interconnected bit-serial SIMD machine. The new generation of multicomputers that use 2-dimensional (2-D) meshes includes the MIT J-machine [7], the Symult 2010 [19], and the Intel Touchstone [16]. The GRAY T3D [12] system uses a 3-D torus.

The *safety-level-based* (or *safety-vector-based*) routing [22], [25], a special form of *limited-global-information-based* routing, is a compromise between local-information- and global-information-based approaches. In this type of routing neighborhood fault information is captured by an integer or a binary vector associated with each node. For example, in a binary hypercube, if a node is associated with a safety level l, then there is at least one Hamming distance path (also called minimal path) from this node to any node within the l-Hamming-distance. Using the level or vector associated with each node, a routing algorithm can obtain a minimal path and require a relatively simple process to collect and maintain fault information in the neighborhood (such information is called *limited global information*). Therefore, limited-global-information-based routing can be more cost effective than routing based on either global or local information. The safety-level-based routing has been successfully applied to high-dimensional mesh-connected topologies such as binary hypercubes, but it is less efficient when directly applied to low-dimensional mesh-connected topologies such as 2-D meshes and tori.

In this paper we extend the safety level concept to low-dimensional mesh-connected multicomputers and use 2-D meshes as an example. The challenge is to find a minimal path in a mesh with faulty blocks. At the same time, the amount of limited-global-information should be kept to a minimum and it should be easy to obtain and maintain. In a 2-D mesh, the extended safety level information is captured by a vector associated with each node. When the safety level of a node reaches a certain level (or meets certain conditions), a minimal path exists from this node to any nonfaulty nodes in 2-D meshes. Specifically, we address the issues of existence of a minimal path at a given source node, limited distribution of fault information, and minimal routing itself. We propose three fault-tolerant minimal routing algorithms which are adaptive to allow all messages to use any minimal path. Our approach is the first attempt to address minimal routing in 2-D meshes with faulty blocks using limited fault information. Although our study focuses only on 2-D meshes, the results can be easily extended to other low-dimensional mesh-connected multicomputers such as 2-D tori and 3-D meshes.

This paper is organized as follows: Section 2 presents preliminaries and motivation of this study. Section 3 proposes a sufficient condition for minimal routing. An extended safety level is defined based on this condition. Section 4 offers three minimal and adaptive routing algorithms based on the safety status of the source node and/or limited global information in 2-D meshes. Section 5 provides a stronger sufficient condition for minimal routing and discusses possible extensions of the extended safety level concept to other low-dimensional mesh-connected multicomputers. Issues of deadlock-free and livelock-free routing are also discussed. Related works are considered and compared in Section 6. Section 7 concludes this paper.

2 Preliminaries

In this section, we review some basic concepts of routing in 2-D meshes and fault models. We also point out some general problems of existing approaches as part of the motivation for this paper.

M-ary n-dimensional meshes

An m-ary n-dimensional (n-D) mesh with m^n nodes has an interior node degree of $2n$ and the network diameter of $m(n-1)$. Each node u has an address $(u_1, u_2, ..., u_n)$, where $0 \leq u_i \leq m-1$. Two nodes $u : (u_1, u_2, ..., u_n)$ and $v : (v_1, v_2, ..., v_n)$ are connected if their addresses differ in one and only one element (dimension), say dimension i; moreover, $|u_i - v_i| = 1$. Basically, nodes along each dimension are connected as a linear array.

Each node in a 2-D mesh is labeled as (i, j). We do not specify the size of a mesh. Therefore, the westmost node of row i is labeled as $(-\infty, j)$, Similarly, labels (∞, j), (i, ∞), $(i, -\infty)$ are used for eastmost, northmost, and southmost nodes, respectively.

Minimal routing in 2-D meshes

Routing is a process of sending a message from a source to a destination. Throughout this paper, the source is $(0,0)$ and the destination is (i,j), with $i,j>0$. A routing is *minimal* if the length of the routing path from the source to the destination is the distance between these two nodes, i.e., Hamming distance $|i|+|j|$. We consider here only minimal routing, i.e., the source node should not start a routing if there does not exist a minimal path. Therefore, there should be a simple method at the source node to determine the existence of a minimal path in a system with faulty blocks. A more challenging issue is to find a minimal path (if there exists one) by avoiding faulty blocks in the system.

The simplest routing algorithm is *deterministic* which defines a single path between the source and destination nodes. X-Y routing is an example of deterministic and minimal routing in which the message is first forwarded along the X dimension and is then routed along the Y dimension. *Adaptive* routing algorithms, on the other hand, support multiple paths between the source and destination. *Fully adaptive and minimal* routing algorithms allow all messages to use any minimal path.

Block fault model

Most literature on fault-tolerant routing use disconnected rectangular blocks ([1], [2], [4], [15], [20]) to model node faults (link faults are treated as node faults) and to facilitate routing in 2-D meshes. First, a node labeling scheme is defined and this scheme identifies nodes that cause routing difficulties. Adjacent nodes with labels (including faulty nodes) form faulty rectangle regions [20]:

Definition 1: *In a 2-D mesh a nonfaulty node is initially labeled enable; however, its status is changed to disable if there are two or more disable or faulty neighbors. Connected disable and faulty nodes form a faulty block.*

For example, if there are three faults $(1,1)$, $(1,2)$, and $(2,1)$, the corresponding faulty block is a rectangle containing nodes $(1,1)$, $(1,2)$, $(2,1)$, and $(2,2)$. Clearly, there are three possible labels of a node: faulty, nonfaulty and enable, and nonfaulty and disable. This faulty block can be represented as [1:2, 1:2]. In general, $[x:x', y:y']$ represents a rectangle with four corners: (x,y), (x,y'), (x',y) and (x',y'). In a 2-D mesh each faulty block is a rectangle and the distance between any two faulty blocks is at least three [2].

The convex nature of a faulty block facilitates simple and deadlock-free routing (as we will show it later). Therefore, a nonfaulty node that is marked disable (i.e., it is inside a faulty block) will be treated as a faulty node. We assume that both source and destination nodes are outside faulty


y
d: (i,j)

blocks, i.e., their status are nonfaulty and enable. In a separate paper [24] we discuss approaches to activate disable nodes while still keep the convex feature of the faulty block.

Problems with existing approaches

Almost all the existing fault-tolerant routing algorithms ([1], [2], [4], [15], [20]) use local fault information for block faults. Normally faulty block information is associated with adjacent nodes of each faulty block. Each intermediate node (including the source node) is not aware of the existence and location of a faulty block before reaching one. Minimal routing may not be possible even if there exists one. More seriously, there may be traffic congestion at paths adjacent to each faulty block.

Figure 1 (a) shows a simple routing example with source $(0,0)$ and destination (i,i), and there is one faulty block. The reason for selecting $i = j$ in the destination is to justify the fact that Y-bound and X-bound routings have the same probability when a random routing is used. Node (l,k) is the intersection of two adjacent lines of the faulty block and this intersection is the closest to the source. To reach (i,i) through a minimal path the routing message must visit a node in the section between $(l,0)$ and (l,i) of line $x = l$. If the routing message goes across line $x = l$ in the section between $(l,0)$ and (l,k), a minimal routing path is generated; otherwise, the routing is nonminimal (by going around the faulty block, first north bound and making a first available north-east turn). Assuming that a random routing is used, there is a total of $\binom{l+r}{r}$ ways to forward a message from (0,0) to (l,r), $0 \leq r \leq i$. Therefore, there is a total of $\sum_{r=0}^{i} \binom{l+r}{r}$ different ways to reach a node along the line $x = l$. Among these only $\sum_{r=0}^{k} \binom{l+r}{r}$ will lead to minimal paths. Table 1 shows the percentage of minimal routing for different selections of l's and i's. It is assumed that $k = i/2$; that is, half of the nodes along line $x = l$ are blocked by the faulty block. However, the percentages of minimal routing are below 50% and this situation gets worse for large l. This result shows that the percentage of minimal routing can be very low for certain distributions of faulty blocks even if there exist many minimal paths.

Let us consider another related problem. If there are multiple routings with source nodes in the region of the rectangle $[0:l, 0:k]$ with a common destination (i,i), many random routings will end up with nonminimal by going around the faulty block. Clearly, congestion may occur along adjacent nodes of the faulty block.

3 Extended Safety Level

In this section, we will provide a sufficient condition for the existence of a minimal path from a given node. We then define the extended safety level that captures limited global fault information

perimeters	$l = 5$, $i = 10$	$l = 5$, $i = 20$	$l = 5$, $i = 30$	$l = 10$, $i = 20$
percentage	27.78%	40.91%	44.29%	16.55%
perimeters	$l = 10$, $i = 30$	$l = 15$, $i = 20$	$l = 15$, $i = 30$	$l = 20$, $i = 30$
percentage	31.77%	0.86%	11.75%	0.52%

Table 1: Percentage of minimal routing for different values of l and i ($k = i/2$).

at each node. This extended safety level will be used to determine the existence of a minimal path.

A sufficient condition

The following is an important theorem that provides a sufficient condition at the source node for the existence of a minimal path to any destination and it serves as the basis of our approach.

THEOREM 1: *Assume that node $(0,0)$ is the source and node (i,j) is the destination. If there is no faulty block that goes across the X and Y axes, then there exists at least one minimal path from $(0,0)$ to (i,j), i.e., the length of this path is $|i| + |j|$. This result holds for any location of the destination and any number and distribution of faulty blocks in a given 2-D mesh.*

Proof: We prove this theorem by induction on t, the number of faulty blocks. If $t = 1$, we have a situation as shown in Figure 1 (b) with one faulty block $[x : x^{'}, y : y^{'}]$. That is, four corners of this faulty block are (x,y), $(x,y^{'})$, $(x^{'},y)$, $(x^{'},y^{'})$, L_1, L_2, L_3, and L_4 are four adjacent lines of the faulty block as defined in Figure 1 (b). In this way, the region ($x \geq 0$ and $y \geq 0$) is partitioned into eight subregions: R_1, R_2, R_3, R_4, R_5, R_6, R_7, and R_8 (see Figure 1 (b)). Clearly, points in regions R_1, R_2, R_3, R_7, and R_8 can be reached using any minimal routing algorithms. X-Y routing (first X and then Y) can be used to reach any points in R_6. Y-X routing (first Y and then X) can reach points at R_4. Either X-Y or Y-X routing can reach points at R_5.

Assume that the theorem holds for $t = k - 1$. When $t = k$, we draw a line L: $y = j$ that goes through destination (i,j). If there is no faulty block that goes through the section of L between points $(0,j)$ and (i,j), the minimal path is from $(0,0)$ to $(0,j)$ and then from $(0,j)$ to (i,j) (a Y-X routing); otherwise, let faulty block B be the closest faulty block to (i,j) that goes through line L (see Figure 2 (a)). Again, we construct four adjacent lines of faulty block B. Let node u be the intersection of adjacent lines L_1 and L_4 (see Figure 2 (a)). A minimal path can be constructed from u to (i,j) as follows: From u go north along line L_4 until reaching the intersection of L_4 and L, and then go east along line L until reaching (i,j). Based on the faulty block definition, this

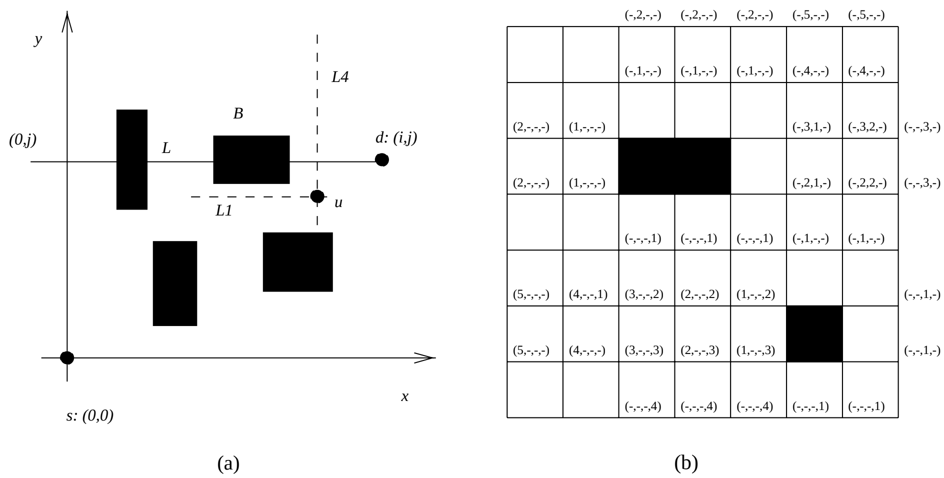

Figure 2: (a) Finding a minimum path. (b) A faulty 8×8 mesh with extended safety levels associated with unsafe nodes.

path is healthy, i.e., it does not go through any faulty block.

Now we construct a path from $(0,0)$ to u. Note that any points in B will not be an intermediate node of a minimal path between u and $(0,0)$. That is, faulty block B does not have any effect on the existence of a minimal path between $(0,0)$ and u. Therefore, we can remove block B. Because the number of faulty blocks is now $k-1$ based on the induction assumption, there exists a minimal path between $(0,0)$ and u. Combining the minimal path from $(0,0)$ to u and the one from u to (i,j), we have a minimal path from $(0,0)$ to (i,j). □

Extended safety level

The safety level concept [25] was originally proposed to capture limited global information in n-cubes. In this model each node is assigned a safety level l, $0 \leq l \leq n$. A node with a safety level $l = n$ is called *safe*; otherwise, it is called *unsafe*. A faulty node is assigned the lowest level 0. The safety level of a nonfaulty node is recursively defined in terms of its neighbors' safety levels. If a node has a safety level l, there is at least one Hamming distance path (minimal path) from this node to any node within l-Hamming-distance. Therefore, the safety level associated with each node can be used to guide the routing message through a minimal path.

The extended safety level in faulty 2-D meshes is represented by a vector. The following definition gives an extended safety level definition for 2-D meshes.

DEFINITION 2: *The extended safety level of a node in a given 2-D mesh is a 4-tuple: (E, S, W, N),* *where E stands for the distance from this node to the closest faulty block to its east. S, W, and N are defined in a similar way. Symbol − is used if there is no fault in the corresponding direction. A node with (-, -, -, -) as its safety level is called safe.*

Based on this definition a safe node is the one without faulty blocks along the east, south, west, and north directions. All the other nodes are unsafe. Figure 2 (b) shows a 8×8 mesh with two faulty blocks. The unsafe nodes are listed in the figure together with their extended safety levels. Based on the extended safety level definition and Theorem 1, we can easily prove the following strengthened result (of Theorem 1) about minimal routing.

THEOREM 2: *Assume source node (0,0) has an extended safety level (E, S, W, N) and the destination node is (i,j), with $i, j > 0$. A minimal path exists if there is no faulty block that goes through the section between 0 and i of the X axis and the section between 0 and j of the Y axis, then there exists at least one minimal path from (0,0) to (i,j). More formally, a minimal path exists if $i \leq E$ and $j \leq N$.*

When the source and destination nodes are randomly distributed, say, source $(i^{'}, j^{'})$ and destination (i, j), the conditions in Theorem 2 can be changed to the following: $|i - i^{'}| \leq W$ (if $i < i^{'}$), $|i - i^{'}| \leq E$ (if $i > i^{'}$), $|j - j^{'}| \leq S$ (if $j < j^{'}$), and $|j - j^{'}| \leq N$ (if $j > j^{'}$).

We can extend the safe node definition to include the ones that meet the conditions in Theorem 2. Such a node is called *extended safe* (with respect to a particular destination) if it meets the conditions in Theorem 2. For general cases, we can use the above extended conditions. For example, node (2,1) in Figure 2 (b) is unsafe based on Definition 2 (assuming node (0,0) is at the lower right corner); however, it is extended safe with respect to destination node (4,3). The extended safety level information can be collected through iterative rounds of message exchange among neighboring nodes. Since each node only needs node information at the same row and column, n rounds are sufficient in an $n \times n$ mesh.

We only assume that the extended safety level is updated in a timely manner without showing how and when. In reality at least three approaches can be used here. (1) *Pessimistic approach*: The updating process is initiated between certain time intervals. In this case we need to select the length of a time interval which can be either fixed or dynamic based on the frequency of the status change, which in turn is dependent on the frequency of fault occurrence. (2) *Optimistic approach*: The updating process is initiated only when at least one node changes its status; that is, no action is required until a new status change is detected. (3) *Hybrid approach*: When there is an infrequent routing applications that require frequent routing operations between two consecutive status changes, i.e., safety level of each node is compensated by the time saved for each routing during this period.

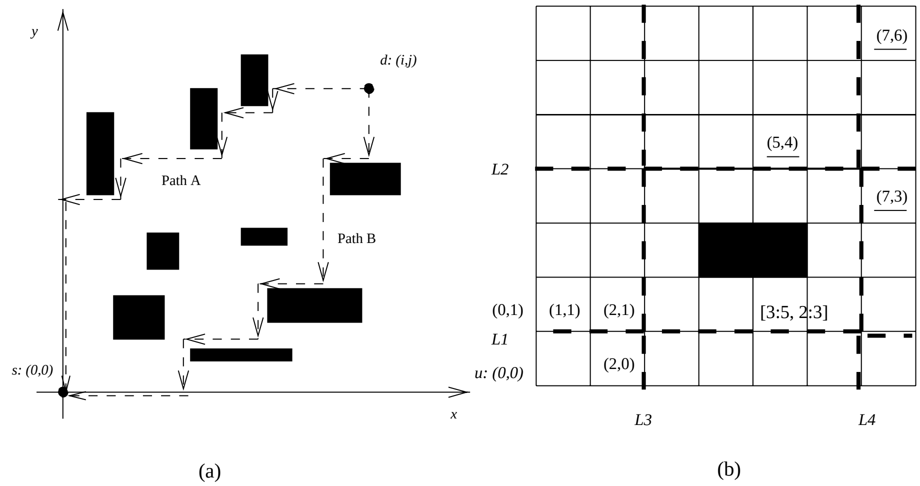

Figure 3: (a) A sample RMP. (b) An example of source-directed routing.

Region of minimal paths (RMP)

The rest of this paper focuses on finding one or all the minimal paths between a source and a destination. We introduce here the concept of *region of minimal paths* (RMP) that includes all the intermediate nodes of minimal paths for a given source and destination pair. That is, nodes and only nodes in this region are used to construct a minimal path. Once the boundary of RMP is known we can construct any minimal path to support a fully adaptive routing. Assume that (0,0) and (i, j) are source and destination nodes, respectively. In a regular mesh without faults, the corresponding RMP is a rectangle $[0 : i, 0 : j]$ with four corners: $(0, 0)$, $(0, j)$, $(i, 0)$, and (i, j).

We now show a method of constructing RMP in a 2-D mesh with faulty blocks: If source (0,0) is extended safe with respect to destination (i, j), we can always find the RMP bounded by two paths, Path A and Path B, connecting the source and destination nodes (see Figure 3 (a)). Starting from node (i, j), Path A is constructed by going west (negative X) until reaching the Y axis and then by going along the Y axis to reach source $(0, 0)$. If the path hits a faulty block, it goes around the faulty block by going south (negative Y). Make a south-west turn whenever possible and continue going west (negative X). Path B is constructed in a similar way. Starting from node (i, j), Path B is constructed by going south (negative Y) until reaching the X axis and then by going along the X axis to reach source $(0, 0)$. The RMP is the area enclosed by Path A and Path B (excluding faulty blocks). If the path hits a faulty block, it goes around the faulty block by going west (negative X). Make a west-south turn when possible and continue going south.

Note that if (0,0) is not extended safe, the above approach may not be able to find the RMP even if one exists. For example, suppose $(x, 0)$ and $(0, y)$ are the only two faulty nodes (blocks). If $x < i$ and $y < j$, the above procedure fails to generate a minimal path while there exists one between (0,0) and (i, j).

The construction of RMP shows that if (0,0) is extended safe respect to (i, j), it is easy to construct a minimal path from (i, j) to $(0, 0)$ without knowledge of fault distribution. Based on the proof of Theorem 1, finding a minimal path from $(0, 0)$ to (i, j) is not easy, it needs information of fault distribution. These properties will serve as the basis of the proposed routing algorithms discussed in the next section.

THEOREM 3: *If source* $(0, 0)$ *is extended safe with respect to* (i, j)*, both Path A and Path B exist. The region enclosed in these two paths (constructed using the above procedure) is the RMP of source (0,0) with respect to destination* (i, j)*.*

Proof: The existence of Path A is straightforward from the proof of Theorem 1. The existence of Path B can be done in a similar way. We then show that all the points inside the region are intermediate nodes of some minimal path. Randomly selecting a node (x, y) in $[0 : i, 0 : j]$, we can construct a minimal path from (x, y) to (i, j) and another one from (x, y) to $(0, 0)$. Intermediate nodes in these paths are constructed progressively and randomly at each step. Note that in the construction of the path between (x, y) to (i, j), if either Path A or Path B is met, the remaining path should follow Path A or Path B to the destination. Combine these two paths and use the fact that $0 \leq x \leq i$ and $0 \leq y \leq j$, the resultant path is a minimal path between (i, j) and $(0, 0)$.

Now we prove that any nodes outside the region cannot be an intermediate node of a minimal path. In this case the corresponding minimal path (a direct one from (0,0) to (i, j)) must go across Path A or Path B. If the routing message went outside the RMP by going across Path A (or Path B), it will eventually go east-bound (or north-bound) to across Path A (or Path B) at least once more in order to reach (i, j). In either case it contradicts to the construction procedure of Path A (or Path B). □

4 Fault-Tolerant Adaptive and Minimal Routing

Assume that node $u : (0, 0)$ is extended safe with respect to $v : (i, j)$ (again $i, j > 0$). We propose the following three fault-tolerant adaptive and minimal routing algorithms.

- *Source-directed routing*: Routing starts from $u : (0, 0)$ once source u verified its safety status with respect to destination v; however, faulty block information is distributed among nodes along adjacent lines of each faulty block to facilitate a minimal routing process.

- *Destination-directed routing*: Routing starts from $v : (i, j)$ (hence, u becomes destination). To ensure the existence of a minimal path, source v needs to know the safety status of destination u. No additional information is needed in the routing process.

- *Mixed-approach routing*: Routing starts from $u : (0, 0)$ once source u verified its safety status with respect to destination v. A signal is first sent to destination v which returns two signals to establish two boundary paths of RMP. No additional information is needed during the routing process.

Source-directed routing

In source-directed routing, routing starts from u once it verified its safety status against destination v, i.e., u is extended safe with respect to v. Therefore, feasibility check is simple and there is no need to disseminate safety information. However, to find a minimal from u to v, faulty block information needs to be distributed among nodes on the adjacent lines of each faulty block; i.e., lines L_1, L_2, L_3 and L_4 as shown in Figure 1 (b), The above distribution of faulty block information is another form of limited global information. Again, faulty block information can be also distributed through iterative rounds of message exchange among neighboring nodes. Clearly, since information is only distributed to two adjacent rows and columns, n rounds are sufficent in an $n \times n$ mesh. Other than nodes along adjacent lines of a faulty block, no other nodes know the existence of the faulty block.

In source-directed routing, any adaptive minimal routing can be used until one adjacent line (L_1 or L_3) of a faulty block is met. Such a line can be either *noncritical* or *critical*. If the selection of two eligible neighbors does not affect the minimal routing, then the line is noncritical; otherwise, it is critical. In the case of noncritical the adaptive minimal routing continues without interruption. In the case of a critical line the selection should follow the rules below:

- (L_1 is met) If the destination is in region R_6 (see Figure 1 (b) by treating s and d as u and v, respectively), L_1 is critical and the routing message should stay on line L_1 until reaching the intersection of L_1 and L_4 of the faulty block; otherwise, L_1 is noncritical and the next step can be randomly selected.

- (L_3 is met) If the destination is in region R_4, L_3 is critical and the routing message should stay on L_3 until reaching the intersection of L_3 and L_2 of the faulty block; otherwise, L_3 is noncritical and the next step can be randomly selected.

Let us consider several routing examples in Figure 3 (b). We assume that source is $(0, 0)$. The faulty block information [3:5, 2:3] (with four corner nodes (3,2), (3,3), (5,2), and (5,3)) is distributed to nodes along four adjacent lines of the block. In this case, since we assume the destination is at the north-east side of the source, faulty block information is distributed to nodes (0,1), (1,1), (2,1) and (2,0). Figure 3 (b) also shows several destinations (with their addresses underlined) which correspond to different routing cases. Initially, routing starts from node $(0,0)$ and randomly goes towards the north-east direction. If the routing message hits line L_3 (i.e., node (2,0)), then line L_3 is critical for destination (5,4) and is noncritical for destinations (7,3) and (7,6). If it hits line L_1 (i.e., node (0,1) or (1,1)), then line L_1 is critical for (7,3) and is noncritical for destinations (5,4) and (7,6).

The above approach can also be applied to multiple-faulty-block cases if faulty blocks are *independent*, i.e., each of the four adjacent lines of a faulty block does not intersect with another faulty block. Situations are more complicated if two and more faulty blocks are *intersected*, i.e., one of the four adjacent lines of a faulty block intersects with another one. In this case, faulty block information may have to be distributed to nodes along adjacent lines of the intersected faulty block. The detailed discussion on handling of intersected multiple faulty blocks can be found in [23].

Destination-directed routing

In destination-directed routing, routing starts from node v to node u if u is extended safe with respect to v. Therefore, safety information needs to be distributed among nodes. In addition, each node knows the status of its adjacent nodes, which can be either faulty, nonfaulty and enable, or nonfaulty and disable. Therefore, an adjacent node of a faulty block knows the existence of the faulty block. However, there is no need for the node to know the size and orientation of the faulty block.

The routing algorithm again consists of two parts: feasibility check and routing. Feasibility check is used to check if it is possible to perform a minimal routing. This can be easily done by comparing the relative coordinates of v and u with the safety status of destination u. Any adaptive and minimal routing for regular 2-D meshes can be applied to find a minimal path from node v to node u as long as u is extended safe with respect to v. An intuitive explanation is that because of the convex nature of a faulty block, each block can at most block one dimension. Therefore, at least one dimension remains free for any source and destination pair that spans two dimensions. When the source (or an intermediate node) and destination pair spans only one dimension, the condition associated with destination u ensures that there is no faulty block along that dimension. Actually, Path A and Path B are two special minimal paths: one is the northmost and the other is the eastmost (see Figure 3 (a) by treating s and d as u and v, respectively).

Mixed-approach routing

The mixed-approach routing combines desirable features from both source-directed and destination-directed approaches. The mixed-approach routing starts from node u (hence, its feasibility check can be easily done at source u). A special signal is sent from u to v. Upon receiving the signal, node v returns two signals to establish Path A and Path B, two boundary lines of RMP. Once source u received two signals, routing starts adaptively without using any fault information.

The following is a detailed procedure: (1) The source node sends a signal to the destination node following a path which may or may not be minimal. (2) Upon receiving the signal the destination node sends two signals: one west-bound and one south-bound. The west-bound signal establishes Path A of RMP and the south-bound signal generates Path B of RMP (see Figure 3 (a)). (3) Once the source node receives both returning signals from the destination node which means that RMP has been established, it sends the routing message using any adaptive and minimal routing. (4) Once the boundary of Path A (or Path B) is met, the remaining routing should follow Path A (or Path B) to reach the destination.

Note that this algorithm applies only to an extended safe source node; otherwise, the source node may not be able to receive two signals from the destination. Also, this algorithm works well when there are many messages from the source to the same destination. Two paths established by two returning signals will serve as boundaries to ensure that only minimal paths are generated. This approach resembles the one in circuit-switching where a path is first constructed before the routing message is forwarded along the path. In mixed-approach routing, the time used in establishing boundary lines will be compensated for by the time saved at each routing.

To check the applicability of the proposed three routing algorithms, we conducted a simulation to determine of the safety status of u and v involved in a routing process. $(s(u), s(v))$ represents the status of u and v with respect to each other, i.e., either extended safe (denoted as $safe$) or not extended safe (denoted as $unsafe$). Therefore, there are four possible situations: $(unsafe, unsafe)$, $(unsafe, safe)$, $(safe, unsafe)$, and $(safe, safe)$. The simulation results are shown in Figure 5 (a). The simulation study was conducted on a 100×100 mesh with a random generation of source and destination nodes with the number of faults ranging from 1 to 200. For each given number of faults we run 50,000 cases of random distribution for each of the three parameters: fault, source, and destination. Faulty blocks are generated from a given fault distribution using Definition 1. Note that at least one of the proposed routing algorithms can be applied if u or v is extended safe. Therefore, the only infeasible case is when both u and v are unsafe, i.e., the pattern $(unsafe, unsafe)$. From Figure 5 (a) it is clear that the percentage for pattern $(unsafe, unsafe)$ is small for a reasonable number of faults, say within 30 faults in a 100×100 mesh. Note that patterns $(unsafe, safe)$ and $(safe, unsafe)$ have almost the same percentage.

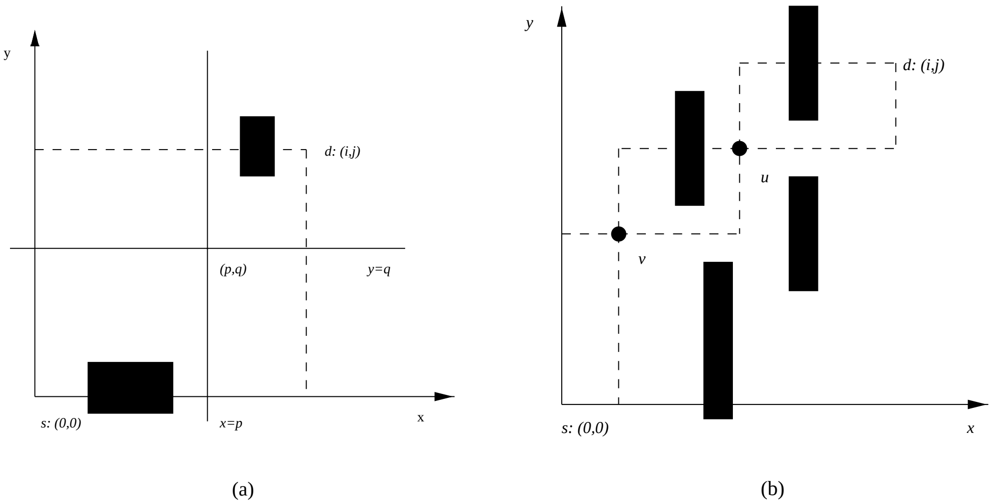

Figure 4: (a) An example that Theorem 2 cannot be applied. (b) An example that Theorem 4 cannot be applied.

5 Extensions

In this section, we consider possible enhancements of the sufficient condition as stated in Theorem 1 and propose extensions of the extended safety level to other low-dimensional mesh-connected multicomputers such as 2-D tori and 3-D meshes.

Enhanced sufficient conditions

The result in Theorem 2 is a sufficient condition but not a necessary one. That is, we may not be able to find a minimal path using Theorem 2 even if one exists. In the example of Figure 4 (a), since there is one faulty block that goes across the X axis, Theorem 2 cannot be applied even if there exists one minimal path. Note that even if we exchange the role of the source and destination nodes, Theorem 2 still cannot be used because there is another faulty block that invalidates the sufficient condition at the new source (i, j). The following theorem shows a stronger sufficient condition for the existence of a minimal path.

THEOREM 4: *Assume that node $(0, 0)$ is the source and node (i, j) is the destination. If there exist p and q with $0 \leq p \leq i$ and $0 \leq q \leq j$ such that there is no faulty block that goes across lines $x = p$ and $y = q$, there exists at least one minimal path from $(0, 0)$ to (i, j), i.e., the length of this path is $|i| + |j|$. This result holds for any location of the destination and any number and distribution of faulty blocks.*

Proof: Consider point (p, q) that goes across both lines $x = p$ and $y = q$. Based on Theorem 2 there exists a minimal path from (p, q) to $(0, 0)$ and a minimal path from (p, q) to (i, j). Clearly, a path formed by combining these two paths is a minimal path from $(0, 0)$ and (p, q). □

In the example of Figure 4 (a) a minimal path exists which is constructed by first forming a minimal path from $(0, 0)$ to (p, q) and then from (p, q) to (i, j). However, Theorem 4 still provides only a sufficient condition. For example, in the example of Figure 4 (b), Theorem 4 fails because any line $y = q$ where $0 \leq q \leq j$ will go across one or two of the faulty blocks. However, by appropriately inserting two intermediate nodes u and v as shown in Figure 4 (b), we can show that a minimal path exists by applying Theorem 2 to the source-destination pairs: (s, v), (v, u), (u, d).

We conducted a simulation study on a 100×100 mesh with random generation of source and destination nodes with the number of faults ranging from 1 to 200. Results show that using the condition of Theorem 2 (the cond1 curve in Figure 5 (b)), we can still achieve a relatively high percentage of safe source nodes. However, this percentage decreases as we increase the number of faults. The percentage based on the condition of Theorem 4 (the cond2 curve in Figure 5 (b)) is much closer to the one based on the optimal one (the optimal curve in Figure 5 (b)), i.e., a sufficient and necessary condition[1]. Note that the optimal curve stays close to 1 even if we increase the number of faults to 200.

Extensions to 2-D tori and 3-D meshes

Another issue is to extend the extended safety level concept to other mesh-connected multicomputers such as 2-D tori and 3-D meshes. A 2-D torus is a 2-D mesh with wraparound connections. Since a 2-D mesh is a subgraph of a 2-D torus, any solutions for 2-D meshes can be directly applied to 2-D tori. However, since a 2-D torus has extra connections, solutions can be simplified and cost can be reduced. Another difference is that a faulty block in a 2-D torus may affect the safety level of a node in both directions of a dimension because of the wraparound links. In a 3-D mesh (and torus) the extension may not be straightforward. The following summarizes the main ideas and a detailed discussion can be found in [3]. (a) Fault model: Clearly the faulty block concept is no longer suitable since connected faulty components may span three dimensions. The concept of a *faulty cube* is used that spans three dimensions. A similar procedure used to form a faulty block can be applied to form a faulty cube. (b) Limited distribution of fault information: In source-directed routing in a 3-D mesh (or torus), faulty cube information should be distributed among nodes along six *adjacent surfaces* of the faulty cube. (c) Minimal routing protocol: The same minimal routing protocol is applied to any dimensional mesh-connected multicomputer. (d) Fault tolerance: For a

[1] Although we have not found the sufficient and necessary condition, we can still determine the existence of a minimal path from a given system configuration through exhaustive searching.

high-dimensional mesh more node-disjoint paths are offered; and hence, a higher degree of fault tolerance is expected.

Deadlock-free and livelock-free routing

Deadlock due to dependencies on consumption resources (such as channels) is a fundamental problem in routing [8]. A deadlock involving several routing processes occurs when there is a cyclic dependency for consumption channels. *Livelock* occurs when a message travels around its destination node, never reaching it because the channels required to do so are occupied by other messages. Livelock is relatively easy to avoid, actually, any minimal routing is livelock-free [8].

Unlike many non-minimal fault-tolerant routing algorithms, the deadlock issue in the proposed model can be easily solved through the use of *virtual network* where a given physical network consists of several virtual networks. Each virtual network consists virtual channels arranged in such a way that no cycle exists among channels, i.e., there is no *intra-virtual-network cycle*.

We can partition a 2-D mesh into four virtual subnetworks +X+Y, +X-Y, -X+Y, and -X-Y (Figure 6 shows two of these subnetworks). Depending on the relative location of the source and destination nodes, one of the four virtual subnetworks is selected and the corresponding routing can be completed within the selected subnetwork without using any other subnetwork. In this way, any *inter-virtual-network cycle* is avoided. In source-directed or mixed-approach routing, we assume that source is $(0,0)$ and destination is (i,j). Therefore, only the +X+Y subnetwork is used. In destination-directed routing, only the -X-Y subnetwork is used with the role of source and destination exchanged. Converting to virtual channels usage, our approach only needs two virtual channels compared with three used in [2].

6 Related Work

In general, fault-tolerant routing schemes depend on the following four decisions: (1) Type of fault region: *convex* and *concave*. (2) Distribution of fault information: *local-information*, *global-information*, and *limited-global-information*. (3) Routing protocol: *progressive* and *backtracking*. (4) Optimality: *minimal* and *nonminimal*.

Disconnected rectangular blocks (convex fault regions) are the most commonly used fault model. Many fault-tolerant routing algorithms are similar in terms of fault tolerance capability and are based on local-fault information, i.e., the fault information is distributed only among the neighbors of faulty blocks. Intermediate nodes do not know the location of faults unless they are neighbors of faults. Therefore, nonminimal paths are used to bypass faults. Since fault regions are

convex, a routing protocol can still be progressive without backtracking. Several fault-tolerant routing approaches have been proposed for non-convex faulty regions [21] and for faults without any distribution constraints [9].

Some nonminimal routing algorithms are based on backtracking which can be easily implemented using the traditional *packet switching* technique. However, they are not suitable for *wormhole routing*, because every channel reserved by the header of the message appears in the final path and the header cannot backtrack and release reserved links. The *pipelined circuit switching* as a variant of wormhole switching is proposed in [9], and resembles a circuit switching that supports backtracking. Another approach to tolerate faults without distribution constraints is based on the *turn model* [10], a model which is originally intended for deadlock free routing. Again it uses local-fault information and is progressive; however, its fault tolerance capability is rather limited: $n-1$ faults can be tolerated in n-D meshes.

All the above approaches use local information. Simplicity is their major advantage; however, optimality is impossible to reach. Therefore, approaches in this category are heuristic in nature. The global-information-based approach assumes each node knows the global distribution of faults. However, this approach is too expensive in collecting and maintaining fault information and it is not scalable. Normally, a routing table is associated with each node. Methods differ in the way the routing table at each node is constructed, maintained, and updated [11], [17]. *Looping* is one of the serious problems which occurs when the paths implied from the routing tables of all nodes contain loops. Looping may cause slow convergence of table updates after a disturbance (fault) occurs.

Very little previous work describes how to model or represent limited global information in mesh-connected networks. To our knowledge there are only three models in the literature:

l-distance knowledge: In this model it is assumed that each node knows the status of all the components (links and nodes) within l-Hamming-distance. This model has been used for unicasting and multicasting [14]. This approach cannot guarantee optimality, because with its limited knowledge a routing might still go to a state where all the minimal paths are blocked by faulty components or even a dead end state where backtracking is required. In addition, it must maintain at each node a relatively large table which contains the status of components. Moreover, the use of this information is not easy because there are many entries in the table. Note that this approach was originally proposed for hypercubes but it can be easily extended to 2-D meshes.

Safe and unsafe: In hypercubes, each faulty node is treated as an independent unit without forming any fault region. Lee and Hayes [13] proposed a scheme that classifies nodes into: *faulty*, *safe*, and *unsafe*. A nonfaulty node is either safe or unsafe. A nonfaulty node is unsafe if and only if there are at least two unsafe or faulty neighbors. There is a minimal path from a safe node to any node in a hypercube. To calculate each node's status an $O(n^2)$-round iterative algorithm is applied

at each node in an n-cube. The major problem with this model is that it is overly pessimistic and it needs a relatively expensive process in collecting information. The above problems are alleviated in Wu and Fernandez [26] by weakening the definition of unsafe node. More specifically, in the enhanced safety status a nonfaulty node is unsafe if and only if there are two faulty neighbors or there are at least three unsafe or faulty neighbors. A unicasting algorithm based on the above enhanced safe node concept is proposed by Chiu and Wu [5]. They showed that a path of length no more than the Hamming distance between the source and the destination plus four can always be established. The *safety level* concept is one further step to extend the safe node concept. If a node has a safety level l, there is at least one Hamming distance path from this node to any node within l-Hamming-distance. In [25] Wu showed that the safety level concept covers a larger set of safe nodes than the ones based on Lee-Hayes' and Wu-Fernandez' definitions and it can be used in the various routing in faulty hypercubes more effectively.

Row/column faults and dead end: Cunningham and Avresky [6] proposed a fault-tolerant routing for 2-D meshes based on another way of defining safe/unsafe nodes where faults are categorized as row/column faults and dead end (east, west, north, south). Therefore, more fault information is maintained at each node compared with our model. A node is unsafe if it cannot guarantee the delivery of a message to all the other safe nodes. A nonfaulty node's status (safe or unsafe) is decided based on row/column faults and dead end information. This approach is progressive but nonminimal.

7 Conclusions

In this paper we have proposed a sufficient condition for minimal routing in 2-D meshes with faulty blocks. Unlike many traditional models that assume all the nodes know global fault distribution or only adjacent fault information, our approach is based on the concept of limited fault information. Specifically, we have proposed three fault-tolerant adaptive and minimal routing algorithms based on the proposed extended safety level information associated with each node in 2-D meshes.

Our approach is the first attempt to provide insight on the design of fault-tolerant and minimal routing in mesh-connected multicomputers. The proposed extended safety level model is a practical model in meshes that captures fault information in a concise format and supports various applications such as minimal routing in faulty environments. This study shows that the safety level for hypercubes can still by effectively used in low-dimensional meshes with a proper extension. Our future research will focus on extending the proposed approach to high-dimensional meshes and to collective communications [18] which include multicast, broadcast, and barrier synchronization.

References

[1] R. V. Boppana and S. Chalasani. Fault tolerant wormhole routing algorithms for mesh networks. *IEEE Transactions on Computers.* 44, (7), July 1995, 848-864.

[2] Y. M. Boura and C. R. Das. Fault-tolerant routing in mesh networks. *Proc. of 1995 International Conference on Parallel Processing.* 1995, I 106- I 109.

[3] X. Chen and J. Wu. Minimal routing in 3-D meshes using extended safety levels. TR-CSE-97-14, Dept. of Computer Science and Engineering, Florida Atlantic University, Feb. 1997.

[4] A. A. Chien and J. H. Kim. Planar-adaptive routing: Low cost adaptive networks for multiprocessors. *Proc. of the 19th International Symposium on Computer Architecture.* 1992, 268-277.

[5] G. M. Chiu and S. P. Wu. Fault-tolerant routing strategy in hypercube multicomputers. *IEEE Transactions on Computers.* 45, (2), Feb. 1996, 143-155.

[6] C. M. Cunningham and D. R. Avresky. Fault-tolerant adaptive routing for two-dimensional meshes. *Proc. of the 1st International Symposium on High Performance Computing Architecture.* 1995, 122-131.

[7] W. J. Dally. The J-machine: System support for Actors. *Actors: Knowledge-Based Concurrent Computing.* Hewitt and Agha (eds.), MIT Press, 1989.

[8] J. Duato, S. Yalamanchili, and L. Ni. *Interconnection Networks: An Engineering Approach.* IEEE Computer Society. 1997.

[9] P. T. Gaughan, B. V. Dao, S. Yalamanchili, and D. E. Schimmet. Distributed, deadlock-free routing in faulty, pipelined, direct interconnection networks. *IEEE Transactions on Computers.* 45, (6), June 1996, 651-665.

[10] G. J. Glass and L. M. Ni. Fault-tolerant wormhole routing in meshes without virtual channels. *IEEE Transactions on Parallel and Distributed Systems.* 7, (6), June 1996, 620-636.

[11] J. Jubin and J. D. Tornow. The DARPA packet radio network protocols. *Proceedings of the IEEE.* 75, (1), January 1987, 21-32.

[12] R. K. Koeninger, M. Furtney, and M. Walker. A shared memory MPP from Cray research. *Digital Technical Journal.* 6, (2), Spring 1994, 8-21.

[13] T. C. Lee and J. P. Hayes. A fault-tolerant communication scheme for hypercube computers. *IEEE Transactions on Computers.* 41, (10), Oct. 1992, 1242-1256.

[14] A. C. Liang, S. Bhattacharya, and W. T. Tsai. Fault-tolerant multicasting on hypercubes. *Journal of Parallel and Distributed Computing.* 23, (3), Dec. 1994, 418-428.

[15] R. Libeskind-Hadas and E. Brandt. Origin-based fault-tolerant routing in the mesh. *Proc. of the 1st International Symposium on High Performance Computer Architecture.* 1995, 102-111.

[16] S. L. Lillevik. The Touchstone 30 gigaflop DELTA prototype. *Proc. of the 6th Distributed Memory Computing Conference.* 1991, 671-677.

[17] J. M. McQuillan, I. Richer, and E. C. Rosen. The new routing algorithm for ARPANET. *IEEE Transactions on Communications.* 28, (5), May 1980, 711-719.

[18] D. K. Panda. Issues in designing efficient and practical algorithms for collective communication on wormhole-routed systems. *Proc. of the 1995 ICPP Workshop on Challenges for Parallel Processing.* Aug. 1995, 8-15.

[19] C. L. Seitz. The architecture and programming of the Ametek Series 2010 multicomputer. *Proc. of the 3rd Conference on Hypercube Concurrent Computers and Applications.* Jan 1988, I 33 - I 36.

[20] C. C. Su and K. G. Shin. Adpative fault-tolerant deadlock-free routing in meshes and hypercubes. *IEEE Transactions on Computers.* 45, (6), June 1996, 672-683.

[21] Y-J Suh, B. V. Dao, J. Duato, and S. Yalamanchili. Software based fault-tolerant oblivious routing in pipelined networks. *Proc. of the 1995 International Conference on Parallel Processing.* 1995, I 101 - I 105.

[22] J. Wu. Adaptive fault-tolerant routing in cube-based multicomputers using safety vectors. *IEEE Transactions on Parallel and Distributed Systems.* 9, (4), April 1998, 321-334.

[23] J. Wu. Fault-tolerant adaptive and minimal routing in mesh-connected multicomputers using extended safety levels. *Proc. of the 18th International Conf. on Distributed Computing Systems.* May 1998, 428-435.

[24] J. Wu. On constructing faulty orthogonal convex polygons in 2-d meshes. TR-CSE-98-5, Department of Computer Science and Engineering, Florida Atlantic University, Jan. 1998.

[25] J. Wu. Reliable unicasting in faulty hypercubes using safety levels. *IEEE Transactions on Computers.* 46, (2), Feb. 1997, 241-247.

[26] J. Wu and E. B. Fernandez. Reliable broadcasting in faulty hypercube computers. *Microprocessing and Microgramming.* 39, 1993, 43-53.