A Distributed Formation of Orthogonal Convex Polygons in Mesh-Connected Multicomputers *

Jie Wu
Department of Computer Science and Engineering
Florida Atlantic University
Boca Raton, FL 33431

Abstract

The rectangular faulty block model is the most commonly used fault model in designing a fault-tolerant and deadlock-free routing algorithm in mesh-connected multicomputers. The convexity of a rectangle facilitates simple and efficient ways to route messages around fault regions using relatively few virtual channels to avoid deadlock. However, such a faulty block may include many nonfaulty nodes which are disabled, i.e., they are not involved in the routing process. Therefore, it is important to define a fault region that is convex, and at the same time, to include a minimum number of nonfaulty nodes. In this paper, we propose a simple and efficient distributed algorithm that can quickly construct a set of special convex polygons, called orthogonal convex polygons, *from a given set of rectangular faulty blocks in a 2-D mesh (or 2-D torus). The formation of orthogonal convex polygons is done through a labeling scheme based on iterative message exchanges among neighboring nodes.*

1. Introduction

In a mesh-connected multicomputer, processors (also called nodes) exchange data and coordinate their efforts by sending and receiving messages through the underlying mesh network. Thus, the performance of such a system depends heavily on the end-to-end cost of communication mechanisms. Routing is the process of transmitting data from one node to another node in a given system. As the number of nodes in a mesh-connected multicomputer increases, the chance of failure also increases. At the same time, applications that run on such a system are often critical and may have real-time constraints. Therefore, the ability to tolerate failure is becoming increasingly important, especially in routing.

Most literatures on fault-tolerant routing use disjoint rectangular blocks ([1], [2], [6]) to model node faults (link faults can be treated as node faults) and to facilitate routing in 2-D meshes. First, a node labeling scheme that identifies nodes (faulty and nonfaulty) that cause routing difficulties is defined and such nodes are called *unsafe nodes*. Connected unsafe nodes form a faulty rectangular region called a a *faulty block*.

Faulty blocks can be easily established and maintained through message exchanges among neighboring nodes. The convexity of each faulty block facilitates a simple fault-tolerant and deadlock-free routing using relatively few virtual channels ([6] and [7]). This feature is also a necessary condition for progressive routing, where the routing process never backtracks. The absence of backtracking in turn is a necessary condition for minimal routing, where the destination is reached through a minimal path from the source. A fault-tolerant minimal routing algorithm for 2-D meshes has been developed in [9] using the faulty block model. There are several studies ([2] and [8]) on fault-tolerant routing that can handle non-rectangular fault regions, such as H-shape, L-shape, T-shape, U-shape, and $+$-shape fault regions. Despite all the desirable features of the faulty block model, a major problem is that a faulty block may include many nonfaulty nodes treated as faulty (with the unsafe label). Although some efforts have been made either to enhance the faulty block definition to include fewer nonfaulty nodes in a faulty block [5] or to activate some boundary nonfaulty nodes in a faulty block as in [1] and [6], the above problem still exists.

A convex region (polygon) is defined as a region (polygon) P for which the line segment connecting any two points in P lies entirely within P. If we change the "line segment" in the standard convex region definition to "horizontal or vertical line segment", the resultant region is called an *orthogonal convex region (polygon)* [4]. Clearly, a faulty block is a special orthogonal convex region. In 2-D meshes, the boundary lines of a region are either horizontal or vertical. Therefore, each region is a polygon. In the subsequent discussion, we use the terms polygon and region interchangeably. In this paper, we consider the following problem: For a given faulty block, activate the maximum

*This work was supported in part by NSF grant CCR 9900646 and grant ANI 0073736. Email: jie@cse.fau.edu.

number of nonfaulty nodes in the block subject to the condition that the resultant region(s) is still an *orthogonal convex polygon(s)*.

A simple and efficient distributed algorithm is presented in this paper that determines a set of small orthogonal convex polygons to cover all the faults in a given faulty block. Let $d(B)$ denote the diameter of faulty block B, while $\max\{d(B)\}$ represents the maximum diameter of all the faulty blocks in the faulty mesh. This algorithm is based on iterative message exchanges among neighboring nodes. Specifically, this approach consists of two phases. In phase one, disjoint faulty blocks are constructed through $\max\{d(B)\}$ rounds of message exchanges among neighbors in a given faulty mesh. In phase two, some nonfaulty nodes in faulty blocks are activated, by removing them from the associated faulty blocks, through up to another $\max\{d(B)\}$ rounds of message exchanges between neighbors.

We show that a resultant region, called a *disabled region*, generated after removing activated nodes from the given faulty block is the smallest orthogonal convex polygon that covers all the faults in the region. Note that there may have several disabled regions generated from a given faulty block. In addition, we show that the number of nonfaulty nodes covered in these disabled regions (from a given faulty block) is no more than that in the smallest orthogonal convex polygon that includes all the faulty nodes in the original faulty block. We note that for certain cases, a disabled region can be further partitioned and more nonfaulty nodes in the region can be removed. This brings the following open problem: For a given faulty block, find a set of orthogonal convex polygons that covers all the faults in the faulty block and contains a minimum number of nonfaulty nodes. This problem is conjectured to be NP-complete [3].

2. Preliminaries

We consider only node faults and assume that faulty nodes just cease to work. Also, each nonfaulty node knows the status of its neighbors only; that is, there is no *a priori* global information of fault distribution. A 2-D $n \times n$ mesh with n^2 nodes has an interior node degree of 4 and a network diameter of $2(n-1)$. Each node u has an address (u_x, u_y), where $u_x, u_y \in \{0, 1, ..., n-1\}$. Two nodes u: (u_x, u_y) and v: (v_x, v_y) are connected if their addresses differ in one and only one dimension, say dimension x. Moreover, $|u_x - v_x| = 1$. Similarly, if they differ in dimension y, then $|u_y - v_y| = 1$.

Definition 1: *A region is* orthogonal convex *if and only if the following condition holds: For any horizontal or vertical line, if two nodes on the line are inside the region, all the nodes on the line that are between these two nodes are also inside the region.*

The difference between a standard convex region and an orthogonal convex region is that the line in the latter is restricted to only horizontal and vertical, whereas the line in the former can be along any direction in a standard convex region. Clearly, T-shape, L-shape, and +-shape fault regions are orthogonal convex polygons, whereas U-shape and H-shape fault regions are non-orthogonal convex polygons.

An application of orthogonal convex regions in achieving fault-tolerant routing in 2-D meshes has been discussed in Chalasani and Boppana's *extended e-cube routing* [2].

3. Node Status

Although the faulty block model for 2-D meshes has been studied by many researchers, there is still no agreement about terminology. Here we introduce a set of concepts and terminology and express some existing models and concepts in terms of the ones in the set. Three orthogonal classifications of nodes in 2-D meshes are given: (1) faulty vs. nonfaulty, (2) safe vs. unsafe, and (3) enabled vs. disabled.

All nodes are either faulty or nonfaulty (healthy). To construct disjoint faulty blocks in 2-D meshes, nonfaulty nodes are further classified into safe and unsafe nodes. Unsafe nodes will be included in faulty blocks and implicitly disabled (i.e., they are treated as faulty). Normally, a faulty block is constructed by first identifying unsafe nodes defined as follows:

Definition 2a: *All faulty nodes are* unsafe. *A nonfaulty node is* unsafe *if it has two or more unsafe neighbors; otherwise, it is* safe.

A *faulty block* consists of connected unsafe nodes. Let u: (u_x, u_y) and v: (v_x, v_y) be two nodes in a 2-D mesh, $d(u, v) = |u_x - v_x| + |u_y - u_y|$ denotes the *distance* between u and v. The *distance* between two faulty blocks A and B is defined as $d(A, B) = \min_{u \in A, v \in B}\{d(u, v)\}$. It has been shown that faulty blocks in 2-D meshes are disjoint rectangles and the distance between any two faulty blocks is at least 3. Figure 1 (a) shows an example of a faulty block where black nodes represent faulty nodes and gray nodes represent nonfaulty but unsafe nodes. The faulty block is bounded by adjacent safe nodes (which are not shown in the figure). Note that under this definition of boundary lines, faulty nodes (u_x, u_y) and (u_x+1, u_y+1) (without any other faulty nodes) are contained in one single region.

Clearly, a faulty block may include many nonfaulty nodes, which is an undesirable feature. To reduce the number of nonfaulty nodes in a faulty block, the following enhanced definition of safe/unsafe nodes is used.

Definition 2b: *All faulty nodes are* unsafe. *A nonfaulty node is* unsafe *if it has an unsafe neighbor in both dimensions; otherwise, it is* safe.

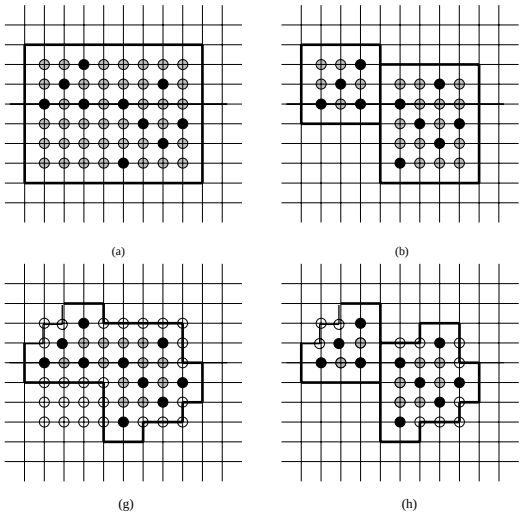

Figure 1. Faulty blocks and disabled regions.

Note that the difference between Definitions 2a and 2b is the following: When a node has exactly two unsafe neighbors and both of them are along the same dimension, this node is an unsafe node based on Definition 2a and a safe node based on Definition 2b.

Figure 1 (b) shows the result of applying Definition 2b to the same example of Figure 1 (a). In this case, there are two disjoint faulty blocks. It can be proved that the distance between two faulty blocks is at least 2. Besides, the total number of nonfaulty nodes included in faulty blocks is less than the one under Definition 2a.

To further reduce the number of nonfaulty nodes in a faulty block, the concept of enabled/disabled nodes can be introduced. Basically, a nonfaulty but unsafe node can be made enabled in a faulty block, i.e., this node can be excluded from the faulty block. *The enabled status of a node is defined based on the enabled/disabled status of its neighbors, rather than depending on safe/unsafe status of its neighbors* as in the definition either by Boura and Das [1] or by Su and Shin [6].

Definition 3 (Wu): *All faulty nodes are marked* disabled. *All safe nodes are marked* enabled. *An unsafe node is initially marked* disabled, *but it is changed to the enabled status if it has two or more enabled neighbors.*

A *disabled region* consists of connected disabled nodes. Clearly, disabled regions are disjoint and the distance between any two disabled regions is at least 2.

Based on the two sets of definitions for node status: one for safe/unsafe nodes based on either Definition 2a or Definition 2b and the other for enabled/disabled nodes based on Definition 3, a faulty node must be unsafe and disabled. For a nonfaulty node, there are three possible cases: (1) safe and enabled, (2) unsafe and enabled, and (3) unsafe and disabled. Figures 1 (c) and (d) show the results of applying the enabled/disabled rule to the examples of Figures 1 (a) and (b), respectively.

To ensure that all boundary nodes in a mesh are treated the same as interior nodes, four additional lines are added which are adjacent to the boundaries lines of the mesh[1]. These additional lines become the new boundaries of the mesh. Nodes along these additional lines are called *ghost nodes* which are safe but they do not participate in any activities, such as in a routing process. Among nodes in the given mesh, only enabled nodes will participate in routing activities.

The enabled/disabled node definition in Definition 3 involves one subtle issue and deserves more discussion. Unlike the "recursive" definition of safe/unsafe status, all nodes are initially marked as disabled or enabled. Node status can be changed later from disabled to enabled following the rule in Definition 3. This is to ensure that the concept of disabled/enabled node is *well-defined*, i.e., each node has one and only one possible assignment of enabled/disabled status. Suppose the enabled/disabled rule is defined recursively as follows: *All faulty nodes are marked disabled. All safe nodes are marked enabled. An unsafe node is enabled if it has two or more enabled neighbors; otherwise, it is marked disabled.* For a given system configuration, unsafe nodes may have "double status", i.e., two or more different enabled/disabled assignments are possible that both satisfy this definition.

Figure 2 (a) shows an example of a faulty block with its upper right block containing only nonfaulty nodes and with the remaining nodes in the faulty block being faulty. Based on the recursive definition of enabled/disabled status, the upper right corner node should be marked enabled. Iteratively we enable all the nonfaulty nodes in the upper right block. Figure 2 (b) shows a similar example, however, the block that contains only nonfaulty nodes is located at the upper center of the faulty block. In this case, we can either enable all the nonfaulty nodes in the block or disable them. Therefore, these nonfaulty nodes have double status. Based on Definition $3c$, all the nodes in the faulty block of Figure 2 (b) have the disabled status. Similar problems exist in all safe/unsafe definitions (Definition 2a and Definition 2b). In fact, each nonfaulty node should be assigned the safe status initially.

Two distributed algorithms are given: one for deciding safe/unsafe status and another one for enabled/disabled status. The first algorithm generates a set of faulty blocks and the second one produces a set of disabled regions that are orthogonal convex polygons. Both algorithms follow the same structure where each node exchanges its status with

[1]The boundary problem does not exist in a 2-D tori with wraparound connections.

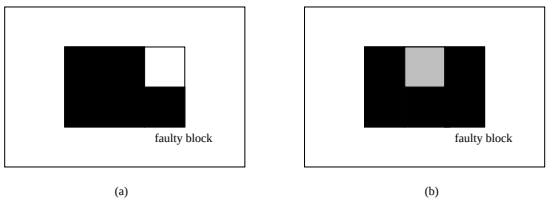

Figure 2. Two sample faulty blocks.

Safe/unsafe status:
all faulty nodes are initialized to unsafe;
all nonfaulty nodes are initialized to safe;
repeat
doall
(1) nonfaulty node u exchanges its status with its neighbors;
(2) change u's status to unsafe if it has an unsafe neighbor in both dimensions
odall
until there is no status change

Enabled/disabled status:
all unsafe nodes are initialized to disabled;
all safe nodes are initialized to enabled;
repeat
doall
(1) nonfaulty but unsafe node u exchanges its status with its neighbors;
(2) change u's status to enabled if it has two or more enabled neighbors.
odall
until there is no status change

its neighbors and changes its status based on the collected neighbors' status. To simplify our discussion, each iterative algorithm is assumed to be synchronous and each round of exchange and update is done in a lock-step mode. The algorithm for safe/unsafe status is based on Definition 2b. The enabled/disabled status is based on the proposed enabled/disabled rule (Definition 3).

Consider an example of a 2-D mesh with three faulty nodes: (1,3), (2,1), and (3,2). Using the safe/unsafe rule, one faulty block $\{(i,j)|i,j \in \{1,2,3\}\}$ is constructed. Using the enabled/disabled rule, the faulty block is split into two disabled regions: $\{(1,3)\}$ and $\{(2,1),(3,2)\}$. All the nonfaulty nodes in the faulty block are enabled.

4. Properties

In this section, we first show that after removing all the enabled nodes (based on Definition 3) in a faulty block, a disabled region containing adjacent faulty and disabled nodes is an orthogonal convex polygon.

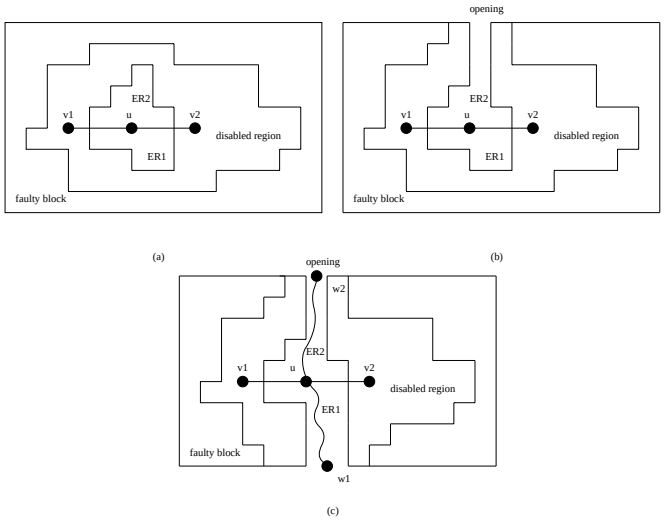

Figure 3. Three cases of disabled regions.

Theorem 1: *A disabled region is an orthogonal convex polygon.*

Proof: Assume that the disabled region is concave. We can find a horizontal (or vertical) line $[v_1, v_2]$ with two nodes v_1 and v_2 on the line both being inside the region, but some node u on the line within these two nodes is outside the region. Consider an *enabled region* ER that includes connected enabled nodes (including node u) in the original faulty block. Line $[v_1, v_2]$ partitions ER into two disjoint enabled regions ER_1 and ER_2. An enabled region is said to have an *opening* if it includes a node w that has a neighbor outside the original faulty block and node w is called an *opening point*.

The following two cases are considered: (1) If either ER_1 or ER_2 does not have an opening, say ER_1, based on the enabled/disabled rule, nodes in ER_1, as well as nodes in ER that are on line $[v_1, v_2]$, should all be marked disabled. This brings a contradiction to the assumption that u is marked enabled. In Figure 3 (a) both ER_1 and ER_2 do not have an opening and in Figure 3 (b) ER_1 does not have an opening and ER_2 has an opening. (2) If both ER_1 and ER_2 have an opening (see Figure 3 (c)), assume that w_1 and w_2 are two opening points in ER_1 and ER_2, respectively; then a path in ER connecting from w_1 to u and from u to w_2 disconnects the disabled region into at least two components, with one containing node v_1 and the other containing node v_2. This contradicts the assumption that the disabled region is connected.

■

Next we show that each disabled region is the smallest orthogonal convex polygon that contains all the faults within its region. First, we introduce a special node called *corner node* and three relevant lemmas.

Definition 4: *A* corner node *in a disabled region is a node that has at least one neighbor, along each dimension, that*

is outside the disabled region.

Lemma 1: *In a disabled region, each corner node is a faulty node.*

Proof: If a corner node of a disabled region is a nonfaulty node, based on Definition 4, it has two enabled neighbors, one along each dimension, that are outside the disabled region. Based on the enabled/disabled rule of Definition 3, this corner node is marked enabled and should be excluded from the fault region. This brings a contradiction. ■

Lemma 2: *For any node u in a disabled region, if the 2-D space is divided into four quadrants induced by horizontal and vertical lines through node u, each quadrant, which includes part of the x and y axes and the origin, contains at least one corner node.*

Proof: For any node u in a disabled region, we divide the space into four quadrants induced by horizontal and vertical line through node u. More specifically, the four regions are defined as follows: quadrant $(+,+)$ with $x \geq 0$ and $y \geq 0$, quadrant $(+,-)$ with $x \geq 0$ and $y \leq 0$, quadrant $(-,+)$ with $x \leq 0$ and $y \geq 0$, and quadrant $(-,-)$ with $x \leq 0$ and $y \leq 0$.

Without loss of generality, we only examine one quadrant, say quadrant $(+,+)$. Independent of the selection of node u, each quadrant contains at least one node in the disabled region which is node u itself (the origin). Among nodes in the disabled region that are in quadrant $(+,+)$, we first select nodes that have the maximum y value and these nodes are denoted as $(+, y_{max})$. Then, among $(+, y_{max})$ we select one node that has the maximum x value and this node is denoted as (x_{max}, y_{max}). Clearly, node (x_{max}, y_{max}) is a corner node because it has one neighbor outside the polygon (which may be a ghost node on the boundary of the 2-D mesh) along each dimension. Note that it is possible that more than one corner node exists in quadrant $(+,+)$. ■

Note that four quadrants overlap with each other either along the adjacent x (or y) axis or at the origin or both. One special case occurs when each quadrant contains only one faulty node which is the origin. In this case, all four quadrants are the same and contain the origin node only.

Lemma 3: *Let u be a node and let B denote a disabled region. If u is not contained in B then at least one quadrant, as defined in Lemma 2, does not contain any nodes in B.*

Proof: When a quadrant contains nodes in B, at least one node in B is on the x or y axis; otherwise, we prove Lemma 3, since all the other three quadrants do not contain any nodes in B. (Because B is connected and covers two quadrants, it must include one node on either the x or y axis.) Without loss of generality, assume that there is a node in B on the x axis and it is on the positive side of it, i.e., it is in both quadrants $(+,+)$ and $(+,-)$. If there is another node in B on the negative side of the x axis, i.e., it is in both $(-,+)$ and $(-,-)$, then the line segment that contains both nodes violates the definition of the orthogonal convex polygon, because the origin u is also on this segment but it is outside B. Therefore, there is at least one node in B along the positive side of the y axis and one node in B along the negative side of the y axis to ensure that both quadrants $(-,+)$ and $(-,-)$ contain at least one node in B and that nodes in B that are in $(-,+)$ and $(-,-)$ are connected without including any nodes on the negative side of the x axis. Then the y axis as a line segment violates the definition of the orthogonal convex polygon (origin u is outside B). ■

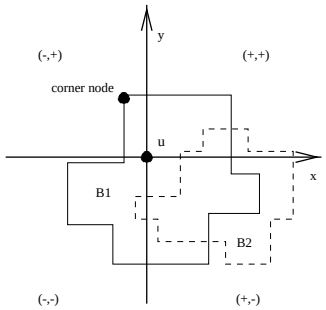

Figure 4. Two convex polygons that cover all the faults within the region.

Theorem 2: *Each disabled region is the smallest orthogonal convex polygon that covers all the faulty nodes within the region.*

Proof: Assume that B_1 is the disabled region under consideration and B_2 is another disabled region that contains all the faulty nodes within the region. If B_2 is smaller than B_1, there is at least one node, say u, that is inside B_1 but outside B_2. Use node u as the origin and draw one horizontal line (called the x axis) and one vertical line (called the y axis). By doing so, the whole space is divided into four quadrants based on the values of x and y in (x, y): $(+,+)$, $(+,-)$, $(-,+)$, $(-,-)$, and each quadrant includes part of the x and y axes and the origin u. Since B_2 is a convex polygon and node u is outside B_2, based on Lemma 3, there is at least one quadrant that does not contain any nodes in B_2, say quadrant $(-,+)$ as shown in Figure 4 without loss of generality. Because node u is inside B_1 and there is at least one corner node in quadrant $(-,+)$ based on Lemma 2, each corner node must be a faulty node based on Lemma 1. This result contradicts the assumption that B_2 contains all the faulty nodes within the region. ■

Corollary: *Given a faulty block, the number of nonfaulty nodes covered in disabled regions, generated by applying the proposed enabled/disabled rule on the faulty block, is no more than the one in the smallest orthogonal convex polygon that contains all the faulty nodes in the faulty block.*

Note that our result is optimal under the assumption that each disabled region cannot be further partitioned. For certain cases, such as the ones in Figures 1 (c) and (d), a disabled region can still be partitioned into several disjoint orthogonal convex polygons. This brings the following open problem: For a given faulty block, find a set of orthogonal

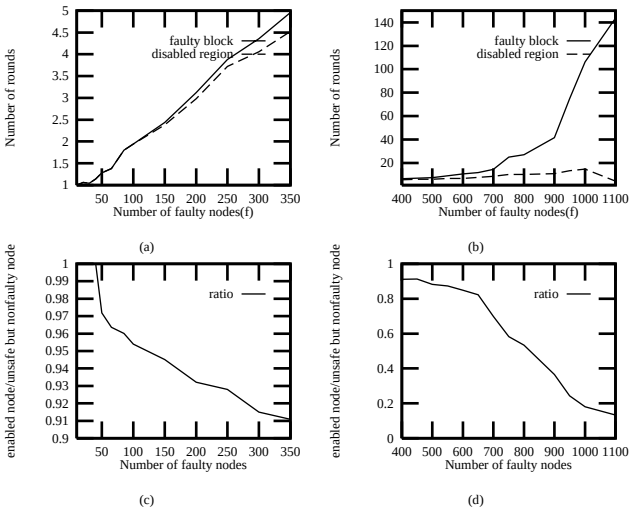

Figure 5. Simulation results.

convex polygons that cover all the faults in the faulty block with a minimum number of nonfaulty nodes inside these polygons. This problem is conjectured to be NP-complete [3].

5. Simulation

The following simulation study has been conducted: (1) For a given $n \times n$ mesh (with $n = 100$ in the simulation) and a given number of faults f (with $10 \leq f \leq 1100$ in the simulation), the averages of the maximum numbers of rounds needed to determine faulty blocks and disabled regions (after the formation of faulty blocks) are calculated (see Figures 5 (a) and (b)). (2) For each faulty block that can be reduced to a set of orthogonal convex polygons, the average percentage of enabled nodes among unsafe but nonfaulty nodes in the faulty block is calculated (see Figures 5 (c) and (d)).

In our simulation study, faults (f in all) are randomly selected among 100×100 nodes in the mesh. The simulation results show that the averages of the maximum numbers of rounds needed to construct faulty blocks and then disabled regions are both relatively row, much lower than the diameter of the mesh. The average number for disabled regions (after the formation of faulty blocks) is lower than the number for faulty blocks, because disabled regions are generated out of faulty blocks. The average percentage of enabled nodes among unsafe but nonfaulty nodes in faulty blocks stays very high, especially when the number of faults is relatively low. This high percentage is in part due to the fact that a random distribution tends to generate a set of small faulty blocks and the fact that nonfaulty nodes in small faulty blocks are easy to be enabled. In summary, the simulation results confirm the effectiveness (high percentage of enabled nodes) of our approach with a relatively low cost (small number of rounds).

6 Conclusions

In this paper, we have proposed a simple and efficient distributed algorithm that can quickly construct a set of orthogonal convex polygons containing all the faulty nodes in a given rectangular faulty block. We have shown that the number of nonfaulty nodes covered in these orthogonal convex polygons is no more than the one in the smallest orthogonal convex polygon that includes all the faulty nodes in the faulty block. Moreover, each orthogonal convex polygon is the smallest one that contains all the faults it covers. The simulation results confirm the cost-effectiveness of the approach, i.e., orthogonal convex polygons can be generated quickly from a given set of faulty blocks. The convexity of a fault region facilitates efficient fault-tolerant and deadlock-free routing. Based on the results of this paper, we can provide a refined fault model to efficiently support several routing objectives, including optimality and freedom of deadlock.

References

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

[2] S. Chalasani and R. V. Boppana. Communication in multicomputers with nonconvex faults. *IEEE Transactions on Computers*. 46, (5), May 1997, 616-622.

[3] D. Z. Chen. private communication.

[4] F. P. Preparata and M. I. Shamos. *Computational Geometry: An Introduction*. Springer-Verlag, 1985.

[5] J. D. Shih. Adaptive fault-tolerant wormhole routing algorithms for hypercube and mesh interconnection networks. *Proc. of the 11th International Parallel Processing Symposium*. April 1997, 333-340.

[6] 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.

[7] P. H. Sui and S. D. Wang. An improved algorithm for fault-tolerant wormhole routing in meshes. *IEEE Transactions on Computers*. 46, (9), September 1997, 1040-1042.

[8] Y. C. Tseng, M. H. Yang, and T. Y. Juang. An Euler-path-based multicasting model for wormhole-routed networks with multi-destination capability. *Proc. of the 1998 International Conference on Parallel Processing*. August 1998, 366-373.

[9] J. Wu. Fault-tolerant adaptive and minimal routing in mesh-connected multicomputers using extended safety levels. *IEEE Trans. on Parallel and Distributed Systems*. 11, (2), Feb. 2000, 149-159.