A New Fault Information Model for Fault-Tolerant Adaptive and Minimal Routing in 3-D Meshes

Zhen Jiang
Department of Computer Science
Information Assurance Center
West Chester University
West Chester, PA 19383
zjiang@wcupa.edu

Jie Wu
Department of Computer Science and Engineering
Florida Atlantic University
Boca Raton, FL 33431
jie@cse.fau.edu

Dajin Wang
Department of Computer Science
Montclair State University
Upper Montclair, NJ 07043
wang@pegasus.montclair.edu

Abstract

In this paper we rewrite the Minimal-Connected-Component (MCC) model in 2-D meshes in a fully distributed manner without using global information so that not only can the existence of a Manhattan distance path be ensured at the source, but also such a path can be formed by routing decisions made at intermediate nodes along the path. We propose the MCC model in 3-D meshes and extend the corresponding routing in 2-D meshes to 3-D meshes. We consider the positions of the source and destination when the new faulty components are constructed. Specifically, all faulty nodes will be contained in some disjoint faulty components and a healthy node will be included in a faulty component only if using it in the routing will definitely cause a non-minimal routing path. A distributed process is provided to collect and distribute MCC information to a limited number of nodes along so-called boundaries. Moreover, a sufficient and necessary condition is provided for the existence of a Manhattan distance path in the presence of our faulty components. As a result, only the routing having a Manhattan distance path will be activated at the source and its success can be guaranteed by using the boundary information in routing decisions at the intermediate nodes. The Monte Carlo simulation results of our new fault information model show substantial improvement in the percentage of success Manhattan routing conducted in 3-D meshes.

Index Terms*:* 3-D meshes, adaptive routing, distributed algorithms, fault information models, minimal routing.

ACRONYMS

2-D meshes	2-dimensional meshes
3-D meshes	3-dimensional meshes
MCC	minimal connected components
$(+X/+Y)$-routing	Manhattan routing from $(0,0)$ to (d_x, d_y) ($d_x, d_y \geq 0$) in 2-D meshes
$(+X/+Y/+Z)$-routing	Manhattan routing from $(0,0,0)$ to (d_x, d_y, d_z) ($d_x, d_y, d_z \geq 0$) in 3-D meshes

NOTATION

s	source node
d	destination node
$[x : x', y : y']$	a rectangular region in 2-D meshes with four vertexes (x, y), (x, y'), (x', y'), and (x', y)
$[x : x, y : y']$	a line segment along the Y dimension with two end points (x, y) and (x, y') and, respectively, we have a line segment along the X dimension $[x : x', y : y]$
$M(c)$	an MCC with the initialization corner c
$Q_X(c)/Q_Y(c)/Q_Z(c)$	the region extending from $M(c)$ along the $X/Y/Z$ dimension that should be forbidden for a Manhattan routing to enter
$Q'_X(c)/Q'_Y(c)/Q'_Z(c)$	the corresponding critical region of $M(c)$
$(+Y-X)$-corner	the corner of a section of an MCC in 3-D meshes parallel to plane $z = 0$ that has the minimum coordinate along the X dimension of those which have the maximum coordinate along the Y dimension and, respectively, we have $(+Y-Z)$-, $(+X-Y)$-, $(+X-Z)$-, $(+Z-Y)$-, and $(+Z-X)$-corners
$(+Y-X)$-edge	the edge of an MCC in 3-D meshes that only contains its $(+Y-X)$-corners and, respectively, we have $(+Y-Z)$-, $(+X-Y)$-, $(+X-Z)$-, $(+Z-Y)$-, and $(+Z-X)$-edges
$[x : x', y : y', z : z']$	a cuboid region in 3-D meshes with eight vertexes (x, y, z), (x, y, z'), (x, y', z'), (x, y', z), (x', y, z), (x', y, z'), (x', y', z'), and (x', y', z)
$[x : x, y : y', z : z']$	a rectangular region on the X plane in 3-D meshes and, respectively, we have rectangles $[x : x', y : y, z : z']$ and $[x : x', y : y', z : z]$ on the Y and Z planes

1 Introduction

In a multicomputer system, a collection of processors (or nodes) work together to solve large application problems. These nodes communicate data and coordinate their efforts by sending and receiving packets through the underlying communication network. Thus, the performance of such a multicomputer system depends on the end-to-end cost of communication mechanisms. The routing time of packets is one of the key factors critical to the performance of multicomputers. Basically, routing is the process of transmitting data from one node, called the source node, to another node, called the destination node. It is necessary to present the *Manhattan routing*, i.e., the minimal routing, which always routes the packet to the destination through a Manhattan distance path [1], so that the destination can be reached in the quickest way.

The *mesh-connected topology* [8], [11] is one of the most thoroughly investigated network topologies for multicomputer systems. Like 2-dimensional (2-D) meshes, 3-D meshes are lower dimensional meshes that have been commonly discussed due to structural regularity for easy construction and high potential of legibility of various algorithms. Some multicomputers were built based on the 3-D meshes [3], [11]. As the number of nodes in a mesh-connected multicomputer system increases, the chance of failure also increases. The complex nature of networks also makes them vulnerable to disturbances. Therefore, the ability to tolerate failure is becoming increasingly important for Manhattan routing [13], [18], [21].

In designing a fault-tolerant routing, one of the most important issues is to select an appropriate fault model. Most existing literature [4], [5], [6], [7], [9], [12], [14], [21], [22] uses the simplest orthogonal convex region to model node faults (link faults can be treated as node faults by disabling the corresponding adjacent nodes). In 2-D meshes, a routing will be blocked from entering the rectangular shape region, also called rectangular fault block, because using any non-faulty node inside it may cause a detour and make the routing non-minimal (see Figure 1 (a)). When the rectangular fault block is extended to cuboid in 3-D meshes, most routings blocked by the rectangle-shaped section of this cuboid on one plane can find a Manhattan distance path along an alternative plane (see Figure 1 (b)). That is, some non-faulty nodes inside the fault cuboid can be used in Manhattan routing and are unnecessarily disabled to form the cuboid-shaped fault region. In the worst case, a few nodes can disable the entire mesh and, further, block any communication (see Figure 1 (c)).

In this paper, we focus on the minimal fault region for Manhattan routing, in which the non-faulty contained are reduced as much as possible. Because the global information models are not suitable for large-scale and complex grid-connected networks, we also focus on a practical implementation of the fault information model in a fully distributed manner to make the whole system more scalable where each node knows only the status of its neighbors. The contributions are listed as follows:

- We introduce the minimal fault region MCC model for Manhattan routing in 3-D meshes. A node will be included in an MCC of 3-D meshes if and only if using it in a routing will cause a detour and make the route non-minimal. In this way, our MCC not only can prevent the routing from using a non-faulty node inside the area where a detour must be made, but also will not block any possible Manhattan distance path. Thus, each MCC has to follow a certain shape. To our knowledge, this is the first attempt to achieve a minimal fault region in 3-D meshes. We consider the positions of the source and destination when the MCCs are constructed.

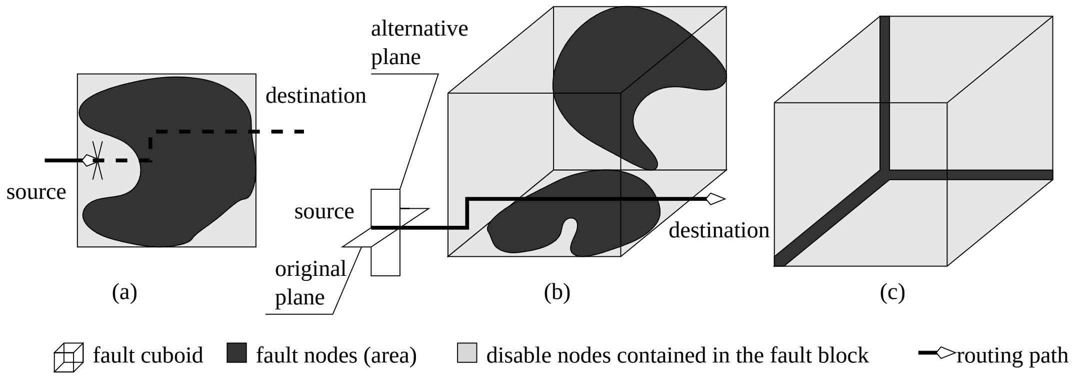

Figure 1. (a) Routing blocked from entering a rectangular faulty block in 2-D meshes. (b) The use of alternative plane in Manhattan routing after the block in the original plane. (c) A sample of fault cuboid that contains too many non-faulty nodes.

- We provide a fully distributed method via information exchanges among neighbors to collect MCC information and distribute to a limited number of nodes, also called boundaries. The boundaries exactly surround the region in which the routing cannot find the Manhattan distance path. The process of boundary construction is not trivial.

- The boundary information is introduced to Wu's adaptive routing [17], where the path is formed by routing decisions at intermediate nodes. Only a Manhattan routing will be activated and its success can be guaranteed.

- Extensive simulation is developed in the Monte Carlo method [2] to determine the number of non-faulty nodes included in MCCs in 3-D meshes and the rate of successful Manhattan routing under the MCC model. The results obtained are compared with the best currently known results.

A short summary of our approach follows. First, without using global information, we rewrite Wang's MCC model in 2-D meshes [15] which is a refinement of the rectangular fault block model. After each non-faulty node in Wang's MCC is labeled, the shape of the fault region is identified in our identification process. Then, the identified information of this MCC will be propagated along the boundaries [10], [16]. After that, Wu's adaptive routing in [17] with two phases is extended to the Manhattan routing in 2-D meshes: In phase one, Wang's sufficient and necessary condition for the existence of a Manhattan distance path, which has been rewritten without using global information, is used to ensure the Manhattan routing at the source node. In phase two, the routing process at each intermediate node, including the source, will forward the message to the next node along the path. It uses the boundary information to keep the route minimal while not missing any Manhattan distance path. Second, our information processes in 2-D meshes are extended to 3-D meshes. We introduce the MCC in 3-D meshes. A labeling process is proposed to identify all nodes inside MCCs. The information propagation along 2-dimensional surfaces in both identification process and boundary construction is implemented through message transmission between two neighboring nodes along one of those three dimensions X, Y, and Z. With this boundary information, Wu's routing in [17] is extended to the Manhattan routing in 3-D meshes.

The remainder of the paper is organized as follows. Section 2 introduces some necessary notations and preliminaries. Section 3 provides our boundary construction for MCCs in 2-D meshes and our boundary-information-based minimal and adaptive routing. In the same section, Wang's sufficient and necessary condition for the existence of a Manhattan distance path is rewritten without using global information. Section 4 presents the MCC model in 3-D meshes and extends the corresponding processes in 2-D meshes to 3-D meshes. A node labeling scheme for non-faulty nodes in each MCC and the construction of boundaries for an MCC are presented. Section 5 provides a sufficient and necessary condition for the existence of a Manhattan distance path in 3-D meshes. In Section 6, our Manhattan routing in 3-D meshes is provided. Its performance improvement compared with the best existing routing is shown in the Monte Carlo simulation results in Section 7. Its improvement on the cost of information model is also shown in Section 7. Section 8 concludes this paper and provides ideas for future research.

2 Preliminary

A k-ary n-dimensional mesh with k^n nodes has an interior node degree of $2n$ and the network diameter is $(k-1)n$. Each node u has an address $(u_1, u_2, ..., u_n)$, where $0 \leq u_i \leq k-1$. Two nodes $(v_1, v_2, ..., v_n)$ and $(u_1, u_2, ..., u_n)$ are neighbors if their addresses differ in one and only one dimension, say dimension i; moreover, $|v_i - u_i| = 1$. Basically, nodes along each dimension are connected as a linear array. In a 2-D mesh, each node u is labeled as (x_u, y_u) and the Manhattan distance between two nodes u and v, $D(u,v)$, is equal to $\mid x_v - x_u \mid + \mid y_v - y_u \mid$. Assume node s is the source node, u is the current node, and d is the destination node. Simply, for a node $u(x_u, y_u)$, node $v(x_u + 1, y_u)$ is called the $+X$ neighbor of u. Respectively, $(x_u - 1, y_u)$, $(x_u, y_u - 1)$, and $(x_u, y_u + 1)$ are $-X$, $-Y$ and $+Y$ neighbors of node u in a 2-D mesh. When node v is a neighbor of node u, v is called a *preferred neighbor* if $D(v,d) < D(u,d)$; otherwise, it is called a *spare neighbor*. Respectively, the corresponding connecting directions are called *preferred direction* and *spare direction*. In general, $[x : x', y : y']$ represents a rectangular region with four vertexes: (x, y), (x, y'), (x', y'), and (x', y). Specifically, $[x : x, y : y']/[x : x', y : y]$ represents a line segment along the Y/X dimension. In a Manhattan routing, the length of the routing path from source node s to destination node d is equal to $D(s,d)$. The Manhattan routing is also called *minimal* routing. Without loss of generality, assume $x_s = y_s = 0$ and $x_d, y_d \geq 0$. The corresponding Manhattan routing is also called $(+X/+Y)$-routing. In this paper, the Manhattan routing in 2-D meshes and $(+X/+Y)$-routing are used alternatively. Similarly, in a 3-D mesh, $(0,0,0)$ is the source node, $u(x_u, y_u, z_u)$ is the current node, $d(x_d, y_d, z_d)$ ($x_d, y_d, z_d \geq 0$) is the destination node, and the Manhattan distance between two nodes u and v, $D(u,v)$, is equal to $\mid x_v - x_u \mid + \mid y_v - y_u \mid + \mid z_v - z_u \mid$. $(x_u + 1, y_u, z_u)$, $(x_u - 1, y_u, z_u)$,

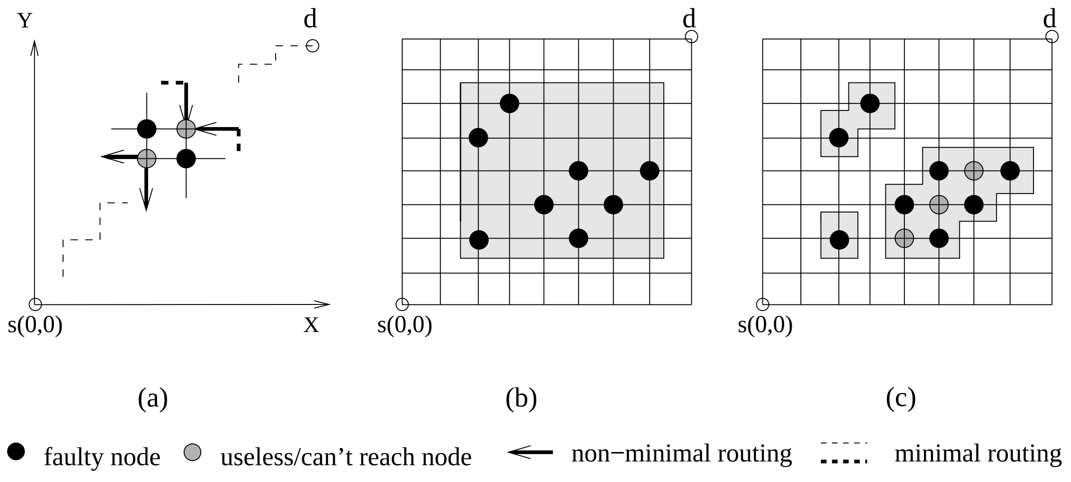

Figure 2. (a) Definition of useless and can't-reach nodes. (b) Sample of rectangular faulty block. (c) The corresponding MCCs

(x_u, y_u+1, z_u), (x_u, y_u-1, z_u), (x_u, y_u, z_u+1) and (x_u, y_u, z_u-1) are $+X$, $-X$, $+Y$, $-Y$, $+Z$, and $-Z$ neighbors of node u. The Manhattan routing in 3-D meshes is also called $(+X/+Y/+Z)$-routing.

The formation of MCC in 2-D meshes [15] is based on the notions of *useless* and *can't-reach* nodes (see Figure 2 (a)): A node labeled useless is a node such that once an $(+X/+Y)$-routing enters it, the next move must take either the $-X$ or $-Y$ direction, making the routing non-minimal. A node labeled can't-reach is a node such that for an $(+X/+Y)$-routing to enter it, a $-X$ or $-Y$ direction move must be taken, making the routing non-minimal. The node status labels of faulty, useless, and can't-reach can be determined through a labeling procedure. All faulty, useless, and can't-reach nodes are also called unsafe nodes. The labeling procedure is given in Algorithm 1 and it can quickly identify the non-faulty nodes in MCCs. Each active node collects its neighbors' status and updates its status. Only those affected nodes update their status. Eventually, neighboring unsafe nodes form an MCC. Figure 2 (a) shows the idea of the definition of useless and can't-reach nodes. Figure 2 (c) shows some samples of MCCs for the routing from $(0, 0)$ to (x_d, y_d) ($x_d, y_d \geq 0$).

Algorithm 1: Labeling procedure of MCC for the routing from $(0, 0)$ to (x_d, y_d) ($x_d, y_d \geq 0$)

1. Initially, label all faulty nodes as *faulty* and all non-faulty nodes as *safe*.
2. If node u is safe, but its $+X$ neighbor and $+Y$ neighbor are faulty or useless, u is labeled *useless*.
3. If node u is safe, but its $-X$ neighbor and $-Y$ neighbor are faulty or can't-reach, u is labeled *can't-reach*.
4. The nodes are iteratively labeled until there is no new useless or can't-reach node.
5. All faulty, useless, and can't-reach nodes (other than safe nodes) are also called *unsafe* nodes.

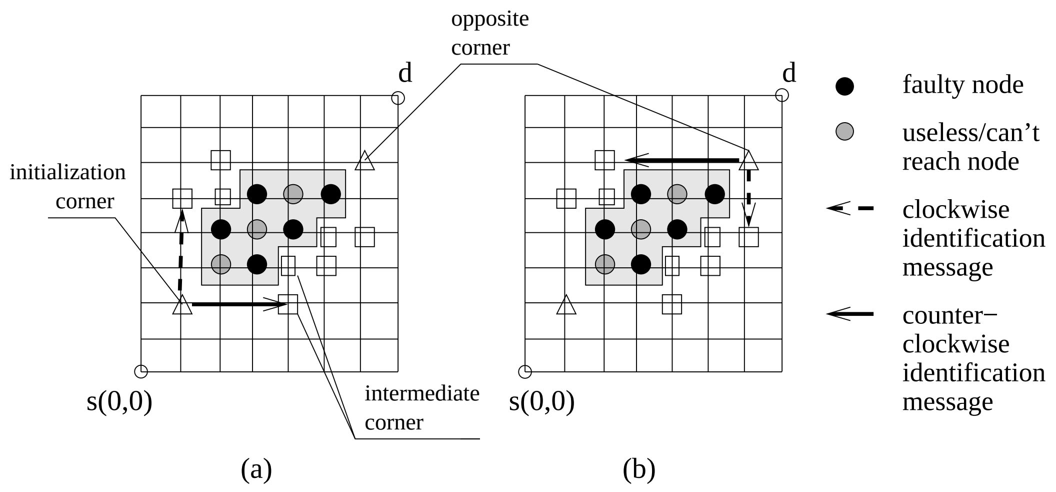

Figure 3. (a) Identification process activated at the initialization corner. (b) Identified information re-sending.

3 Boundary information in MCC model in 2-D meshes

In this section, we provide a distributed process to collect the information of each MCC in 2-D meshes and distribute it along the boundaries so that not only the existence of a Manhattan distance path can be ensured at the source node but also such a routing can be achieved successfully by routing decisions at intermediate nodes along the path. The new routing process provided in this section can find a Manhattan distance path from the source to the destination whenever such a path exists.

3.1 Corner and boundary of MCC in 2-D meshes

To collect the information of all MCCs for the routing process, each MCC needs to identify its fault region. Any node inside the fault region of an MCC is called an unsafe node. Otherwise, it is called a safe node. Any safe node with an unsafe neighbor in an MCC is called an *edge* node of that MCC. A *corner* is a safe node with two edge neighbors of the same MCC in different dimensions or a safe node with two unsafe neighbors of the same MCC in different dimensions. After the labeling procedure, the identification process starts from an *initialization corner*. The initialization corner is a corner with two edge neighbors of the same MCC in the $+X$ and $+Y$ dimensions. A safe node with two edge neighbors of the same MCC in the $-X$ and $-Y$ dimensions is called the *opposite corner*.

From that initialization corner, two identification messages, one clockwise and one counter-clockwise, are initiated. Each message carries partial region information. First, they will be sent to its two edge neighbors. Such propagation will continue along the edges until the messages reach the opposite corner of the same MCC. When the clockwise message passes through an intermediate corner $u(x_u, y_u)$, node information

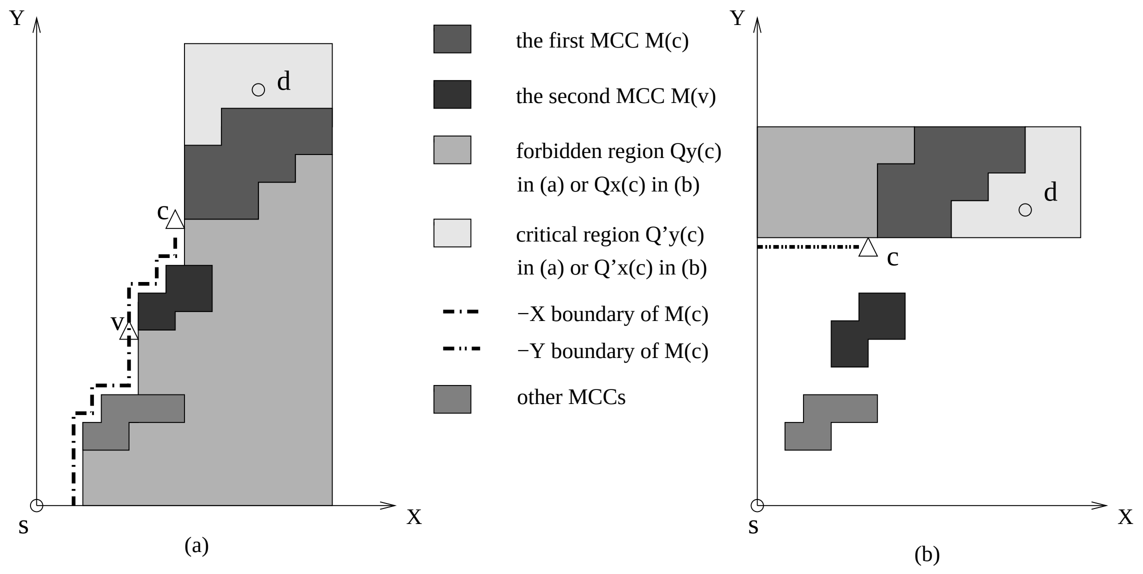

Figure 4. Samples of boundary construction under the MCC model in 2-D meshes.

(x_u, y_u) will be attached to the message. This information will be used at the opposite corner to form the shape of this MCC. Similarly, the counter-clockwise message will also bring the node information of every intermediate corner it passed through to the opposite corner. After these two messages meet at the opposite corner, the propagation will continue and bring the identified information back to the initialization corner. This time, no new intermediate corner needs to be identified and no new information will be added into each message. Figure 3 shows a sample of the identification process.

An MCC has only one initialization corner $c(x_c, y_c)$ and one opposite corner $c'(x_{c'}, y_{c'})$. If two identification messages cannot meet at that opposite corner, or if any of them finds the shape changed when it is sent back to c, it suggests that this MCC is not stable. The message is discarded to avoid generating incorrect MCC boundary information. If only one message is received at the initialization corner, the other has been discarded in the propagation procedure and this message should also be discarded. Normally, a TTL (time-to-live) is associated with each identification message and the corresponding message will be discarded once the time expires.

In 2-D meshes, an MCC with initialization corner c is denoted by $M(c)$. An $(+X/+Y)$-routing message should avoid entering the region right below it in the $+X$ direction if the destination is right above it. The first region is called the *forbidden region*, denoted by $Q_Y(c)$. The corresponding region right above $M(c)$ is called *critical region*, denoted by $Q'_Y(c)$. Similarly, the routing message should avoid entering the forbidden region $Q_X(c)$ on the left side of $M(c)$ in the $+Y$ direction if the destination is in the critical region $Q'_X(c)$ on the right side of $M(c)$. To guide the routing process, two boundary messages will be initiated when two identification messages are both received at node c. One boundary message, also called $-X$ boundary, will carry the information $M(c)$, $Q_Y(c)$, and $Q'_Y(c)$ and propagate to all the nodes along the boundary

Algorithm 2: Identification process and boundary construction of an MCC

1. Identification of edge nodes, the initialization corner $c(x_c, y_c)$, intermediate corners, and the opposite corner $c'(x_{c'}, y_{c'})$.
2. Identification process of MCC $M(c)$: (a) From node c, two identification message (one clockwise and one counter-clockwise) are sent along the edge nodes of $M(c)$ until they reach node c'. (b) Partial region information, including the node information of all intermediate corners and corner c, is transferred to form the shape of $M(c)$ at node c'. (c) After they meet at node c', the propagation will continue until the identified information reaches back to node c. (d) The stable shape of $M(c)$ can be ensured at node c when two identification messages are both received. Meanwhile, the forbidden and critical regions ($Q_X(c)$, $Q_Y(c)$, $Q'_X(c)$, $Q'_Y(c)$) are identified.
3. $-X$ / $-Y$ boundary construction of $M(c)$: A boundary construction is activated at node c after it receives two identification messages. The information of $M(c)$, $Q_Y(c)$ / $Q_X(c)$, and $Q'_Y(c)$ / $Q'_X(c)$ is propagated along the boundary line $x = x_c$ / $y = y_c$. When the propagation intersects another MCC, say $M(v)$, it will make a turn in the $-X$ / $-Y$ direction and go along the edges of $M(v)$. Eventually, it will join the same boundary of $M(v)$. Since then, the forbidden region of $M(v)$, $Q_Y(v)$ / $Q_X(v)$, will merge into that of $M(c)$, $Q_Y(c)$ / $Q_X(c)$.

line $x = x_c$ until it reaches the edge of this 2-D mesh. When this boundary line intersects with another MCC ($M(v)$), a turn in the $-X$ direction is made. After that, it will go along the edges of $M(v)$ to join the same boundary of $M(v)$ at the initialization corner v. At that corner v, $Q_Y(v)$ merges into $Q_Y(c)$ ($Q_Y(c) = Q_Y(c) \cup Q_Y(v)$, see Figure 4 (a)). Similarly, another boundary propagation, construction of $-Y$ boundary, carrying $M(c)$, $Q_X(c)$, and $Q'_X(c)$ will go along $y = y_c$ and make a turn in the $-Y$ direction if necessary (see Figure 4 (b)). The whole procedure is shown in Algorithm 2.

3.2 Sufficient and necessary condition for the existence of a Manhattan distance path in 2-D meshes

The MCC model includes much fewer non-faulty nodes in its fault region than the conventional rectangular model in 2-D meshes. Many non-faulty nodes that would have been included in rectangular faulty blocks now can become candidate routing nodes. As a matter of fact, MCC is the "ultimate" minimal fault region; that is, no non-faulty node contained in an MCC will be useful in a Manhattan routing. A routing that enters a non-faulty node in the MCC would force a step that violates the requirement for a Manhattan routing. In other words, the MCC is a fault information model that provides the maximum possibility to find a Manhattan routing in the presence of faults. If no Manhattan routing exists under the MCC model, there will be absolutely no Manhattan routing. In [15], a sufficient and necessary condition was provided for the existence of a Manhattan distance path. This can be rewritten as the following:

Lemma 1: *A routing does not have a Manhattan distance path iff there exists an MCC $M(c)$ that (a)*

$s \in Q_X(c) \wedge d \in Q'_X(c)$, or (b) $s \in Q_Y(c) \wedge d \in Q'_Y(c)$.

Proof: A sequence of MCCs (M_1, M_2, ..., M_n), such that

1. M_1 contains a node $(0, y_1)$ and $0 < y_1 < y_d$,
2. M_n contains a node (x_d, y_n) and $0 < y_n < y_d$,
3. For all M_i and M_{i+1}, $1 \leq i \leq n-1$,

$$\begin{aligned} \min\{a \mid (a,b) \in M_{i+1}\} &\leq \max\{u \mid (u,v) \in M_i\} \\ &\leq \max\{x \mid (x,y) \in M_{i+1}\} \end{aligned}$$

and

$$\max\{v \mid (u,v) \in M_i\} < \max\{y \mid (x,y) \in M_{i+1}\}$$

is called Type-I sequence in [15] (see Figure 5 (a)). It is obvious that there exists such a sequence if and only if there exits an MCC M(c) that $s \in Q_Y(c)$ and $d \in Q'_Y(c)$.

Similarly, a sequence of MCCs (M_1, M_2, ..., M_n), such that

1. M_1 contains a node $(x_1, 0)$ and $0 < x_1 < x_d$,
2. M_n contains a node (x_n, y_d) and $0 < x_n < x_d$,
3. For all M_i and M_{i+1}, $1 \leq i \leq n-1$,

$$\begin{aligned} \min\{b \mid (a,b) \in M_{i+1}\} &\leq \max\{v \mid (u,v) \in M_i\} \\ &\leq \max\{y \mid (x,y) \in M_{i+1}\} \end{aligned}$$

and

$$\max\{u \mid (u,v) \in M_i\} < \max\{x \mid (x,y) \in M_{i+1}\}$$

is called Type-II sequence (see Figure 5 (b)). There exists a Type-II sequence if and only if there exists an MCC M(c) that $s \in Q_X(c)$ and $d \in Q'_X(c)$. Based on the result presented in [15] that a Manhattan routing can be found if and only if neither Type-I sequence nor Type-II sequence exists, the statement is proved to be true. For the details, refer to [15]. ■

Theorem 1: *A routing does not have a Manhattan distance path if and only if there exists an MCC in which (a) $d \in Q'_Y(c)$ and its $-X$ boundary does not intersect with the segment [$0:x_d$, $0:0$], or (b) $d \in Q'_X(c)$ and its $-Y$ boundary does not intersect with the segment [$0:0$, $0:y_d$].*

Proof: When $d \in Q'_Y(c)$, based on our construction for $-X$ boundary in Algorithm 2, $s \in Q_Y(c)$ if and only if the $-X$ boundary does not intersect with the segment [$0:x_d$, $0:0$]. When $d \in Q'_X(c)$, similarly, $s \in Q_X(c)$ if and only if the $-Y$ boundary does not intersect with the segment [$0:0$, $0:y_d$]. With Lemma 1, the statement is easy to prove. ■

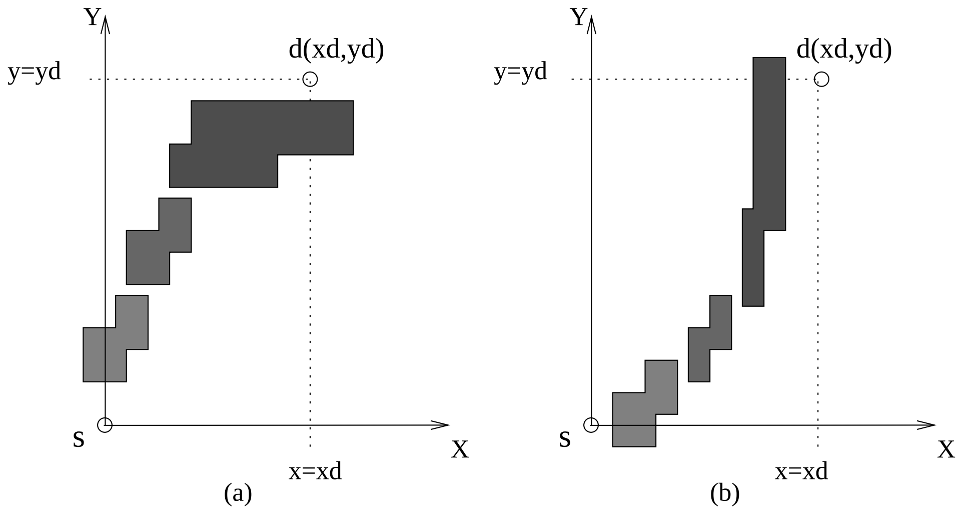

Figure 5. (a) Type-I MCC sequence. (b) Type-II MCC sequence.

3.3 Boundary-information-based routing under the MCC model in 2-D meshes

Wu proposed a minimal and adaptive routing in n-D meshes in [17]. It can easily be extended to a routing in 2-D meshes under the MCC model (see Algorithm 3). In this routing, at the source node s, the feasibility check is first activated to make sure that the Manhattan routing exists. Otherwise, the routing will stop. First, after a 180 degree self-rotation, the source node s will play the role of the destination in Theorem 1. It sends two detection messages, one along the $-X$ boundary from s ($+X$ boundary in the original mesh network before the rotation) and one along the $-Y$ boundary from s ($+Y$ boundary in the original mesh network). The first one is to check if a type-I sequence [15] exists or not. If it can reach the segment $[0 : x_d, y_d : y_d]$ in the original network before rotation (return “YES”), it will make the condition (b) in Theorem 1 false. In other words, there is no type-I sequence. Similarly, the second one is to check if a type-II sequence exists or not. If the source s knows both segments can be reached, based on the sufficient and necessary condition in Theorem 1, a Manhattan routing from d to s exists. That is, a Manhattan routing from s to d exists.

At each node along the routing path, including the source node s, the routing process basically has two preferred directions: $+X$ and $+Y$. The boundary information of an MCC at the current node will help the routing process avoid entering the forbidden region by excluding the corresponding preferred direction from the candidates of the forwarding direction. After that, any fully adaptive and minimal routing process could be applied to select the forwarding direction and forward the routing message along this direction to the corresponding neighbor. The procedure of feasibility check and routing decision using boundary information can also be seen in the samples in Figure 6.

Algorithm 3: Routing from $s(0,0)$ to $d(x_d, y_d)$ $(x_d, y_d \geq 0)$

1. Feasibility check: At source s, send two detection messages (the first along the $+X$ boundary from s and the second along the $+Y$ boundary from s) until they reach the line $y = y_d$ or line $x = x_d$. If it intersects with the segment [0 : x_d, y_d : y_d] / [x_d : x_d, 0 : y_d], return "YES" to node s; otherwise, return "NO". If any return is "NO", stop the routing since there is no Manhattan distance path.
2. Routing decision and message sending at the current node u, including the source s:
   (a) Add all the preferred directions into the set of candidates of forwarding directions F and find all the recorded MCCs.
   (b) For each $M(c)$ found, exclude $+Y/+X$ direction from F if $d \in Q_X(c)/Q_Y(c)$.
   (c) Apply any fully adaptive and minimal routing process to select a forwarding direction from set F.
   (d) Forward the routing message along the selected forwarding direction to the next node.

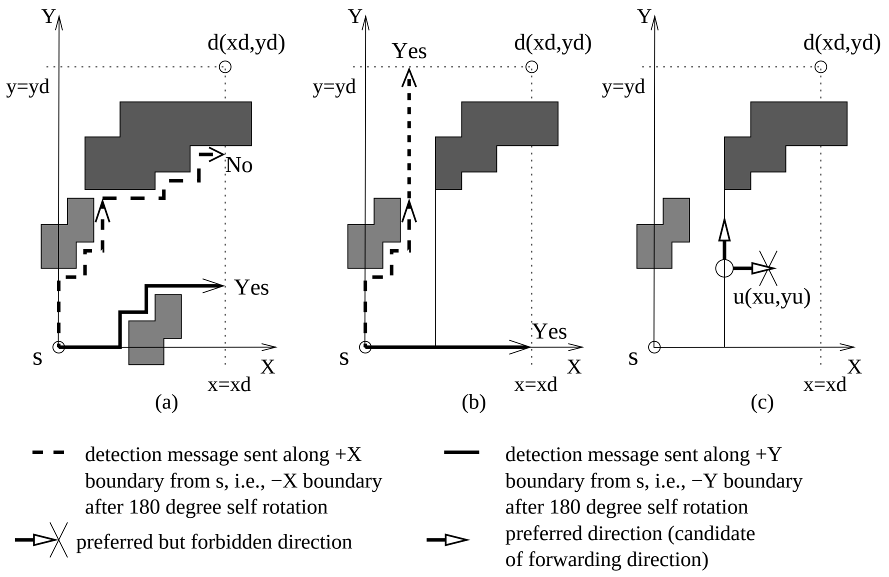
Y
d(xd,yd)
y=yd
No
Yes
s
X
x=xd
(a)
Y
Yes
d(xd,yd)
y=yd
Yes
s
X
x=xd
(b)
Y
d(xd,yd)
y=yd
u(xu,yu)
s
X
x=xd
(c)
detection message sent along +X boundary from s, i.e., −X boundary after 180 degree self rotation
detection message sent along +Y boundary from s, i.e., −Y boundary after 180 degree self rotation
preferred but forbidden direction
preferred direction (candidate of forwarding direction)

Figure 6. (a) Feasibility check for a case without any Manhattan distance path. (b) Feasibility check to ensure the existence of a Manhattan distance path. (c) Routing decisions in routing process to construct a Manhattan distance path.

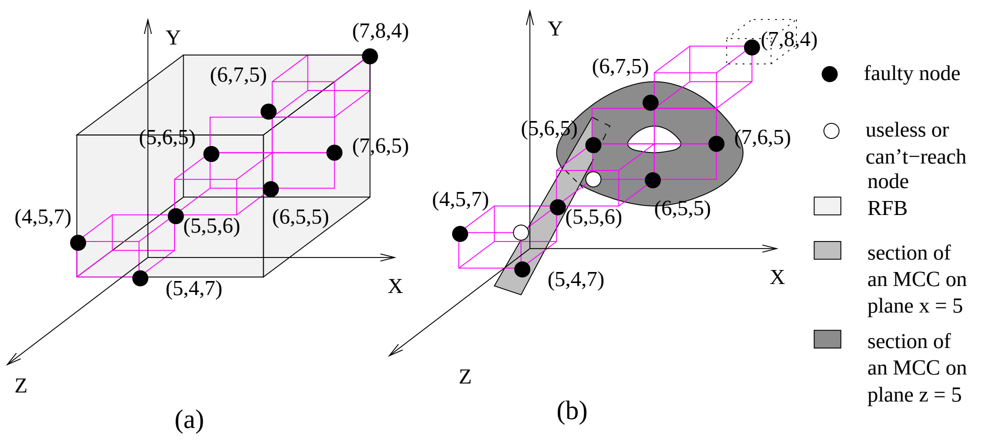
Y
X
Z
(7,8,4)
(6,7,5)
(5,6,5)
(7,6,5)
(4,5,7)
(5,5,6)
(6,5,5)
(5,4,7)
(a)
(b)
faulty node
useless or can't−reach node
RFB
section of an MCC on plane x = 5
section of an MCC on plane z = 5

Figure 7. (a) Sample cuboid faulty block in 3-D meshes. (b) The corresponding MCCs.

4 MCC model in 3-D meshes

In this section, we present our distributed solution for constructing MCCs and propagating the region information in 3-D meshes. First, the status of each node inside a fault region is identified in a labeling process. Then, each 2-D section and its neighboring section in a 3-D fault region are identified in an identification process. After that, the information of 2-D sections is collected along the edges in the edge construction. With this information, the region is identified as an MCC and the information of its shape, forbidden region, and critical region is formed. Finally, in the boundary construction, the formed information will be propagated along the boundaries to prevent the routing from entering the forbidden region.

4.1 Labeling process

A useless node u in an MCC in 2-D meshes has two useless or faulty neighbors in the $+X$ and $+Y$ directions. Based on the label scheme in Algorithm 1, any $(+X/+Y)$-routing will be blocked by the faulty nodes if it enters node u. For a non-faulty node u in 3-D meshes, if it has only two useless or faulty neighbors in the $+X$ and $+Y$ directions, the routing message can still route around the fault region in the $+Z$ direction. Therefore, a non-faulty node is useless in 3-D meshes if and only if it has three useless or faulty neighbors in the $+X$, $+Y$, and $+Z$ directions. Similarly, a non-faulty node is can't-reach if and only if it has three can't-reach or faulty neighbors in the $-X$, $-Y$, and $-Z$ directions. The corresponding labeling scheme is shown in Algorithm 4.

Algorithm 4: Labeling procedure of MCC in 3-D meshes

1. Initially, label all faulty nodes as *faulty* and all non-faulty nodes as *safe*.
2. If node u is safe, but its $+X$ neighbor, $+Y$ neighbor, and $+Z$ neighbor are faulty or useless, u is labeled *useless*.

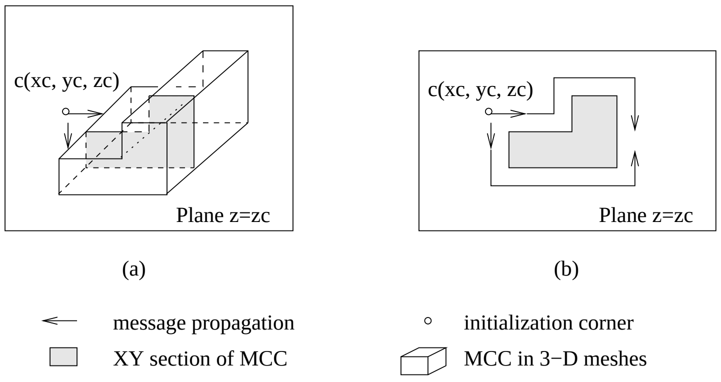

Figure 8. Identification of an XY section with the initialization $(+Y - X)$-corner c.

3. If node u is safe, but its $-X$ neighbor, $-Y$ neighbor, and $-Z$ neighbor are faulty or can't-reach, u is labeled *can't-reach*.
4. The nodes are iteratively labeled until there is no new useless or can't-reach node.
5. All faulty, useless, and can't-reach nodes are also called *unsafe* nodes.

Figure 7 (b) shows two sample MCCs in 3-D meshes. First, $(5,5,6)$, $(6,5,5)$, $(5,6,5)$, $(6,7,5)$, $(7,6,5)$, $(5,4,7)$, $(4,5,7)$, and $(7,8,4)$ are faulty nodes. Then, $(5,5,5)$ becomes useless and $(5,5,7)$ becomes can't-reach according to our labeling process in Algorithm 4. One MCC contains only one faulty node $(7,8,4)$ and the other MCC contains all the other faulty, useless, and can't-reach nodes. Usually, a 2-D section of the MCC parallel to plane $x = 0$, plane $y = 0$, or plane $z = 0$ is not a *convex polygon*. A convex polygon has been defined in [20] as a polygon P for which a line segment connecting any two points in P lies entirely within P. A non-convex section of the second MCC on the plane $z = 5$ with a hole at $(6,6,5)$ is shown in Figure 7 (b).

In the following subsection, we will introduce a process (see Algorithm 5) to collect the shape information of each MCC and distribute to a limited number of nodes along so-called boundaries for our Manhattan routing.

4.2 Identification process

The identification process for an MCC in 3-D meshes is based on the one for the MCC in 2-D meshes. It starts from the identification of each 2-D section on the XY plane, YZ plane, and XZ plane simultaneously. Simply, we call these sections XY sections, YZ sections, and XZ sections. For each section, for example, an XY section, a two-head-on message identification process in Algorithm 2 is activated at its corner c, say one with the minimum coordinate along the X dimension of those which have the maximum coordinate along the Y dimension (see Figure 8). Such a corner is also called the $(+Y-X)$-corner of this XY section.

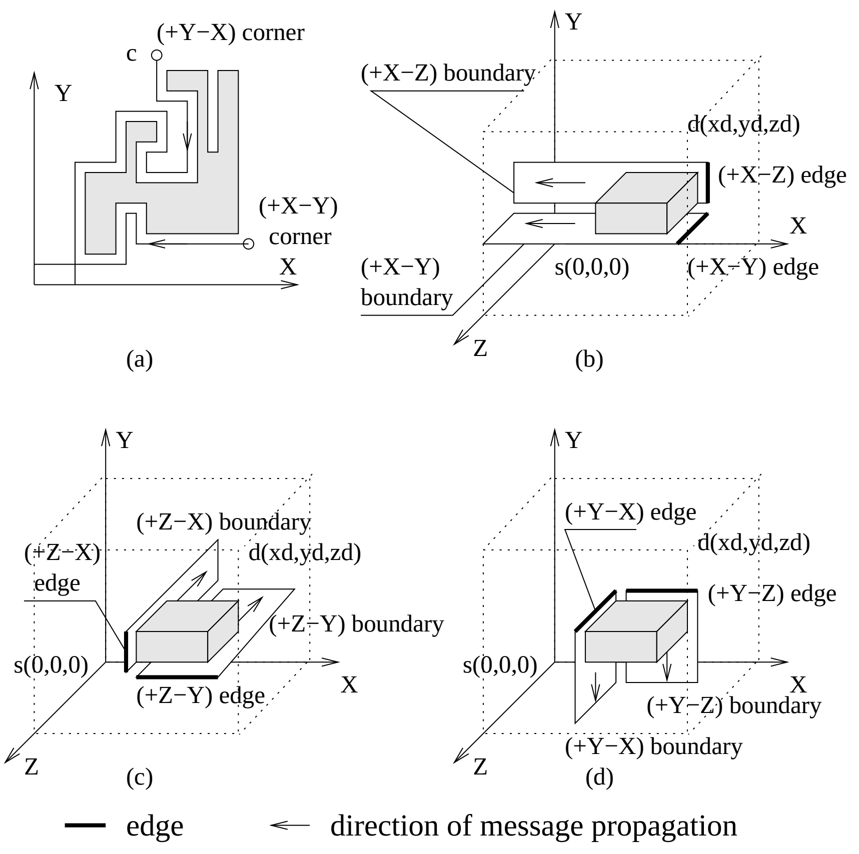

Figure 9. Samples of corners, edges, and boundaries in 3-D meshes.

The corner of a section uses the previous definition for the one of an orthogonal fault region in 2-D meshes. This XY section may have several corners with the maximum coordinate along the X dimension and the one with the minimum coordinate along the Y dimension is called the $(+X - Y)$-corner of this section. Respectively, we have $(+X - Z)$- and $(+Z - X)$-corners of a XZ section, and $(+Y - Z)$- and $(+Z - Y)$-corners of a YZ section. Each YZ / XZ section will be identified by a similar process initiated from its $(+Z - Y)$- / $(+X - Z)$-corner. It is noted that these two identification messages may meet at any edge node of the section, not necessarily a corner node (see Figure 8 (b)).

After section identification, six kinds of edges of each MCC are identified for the boundary construction: $(+Y - X)$-edge, $(+Y - Z)$-edge, $(+X - Y)$-edge, $(+X - Z)$-edge, $(+Z - Y)$-edge and $(+Z - X)$-edge (see Figure 9). Each of these edges are defined by all of its edge nodes. Each edge node is the corresponding corner in its 2-D section. The identification process of edge nodes is to find the path linking edge nodes in two neighboring 2-D sections. The edge node next to the other in the routing direction, $+X$, $+Y$, or $+Z$ direction, is called a succeeding edge node. The other one is called a preceding edge node. By connecting all the links together, the entire edge can be constructed. Such a process has three phases. In phase one, a message will be initiated at an edge node and route around its 2-D section to find a path to the neighboring

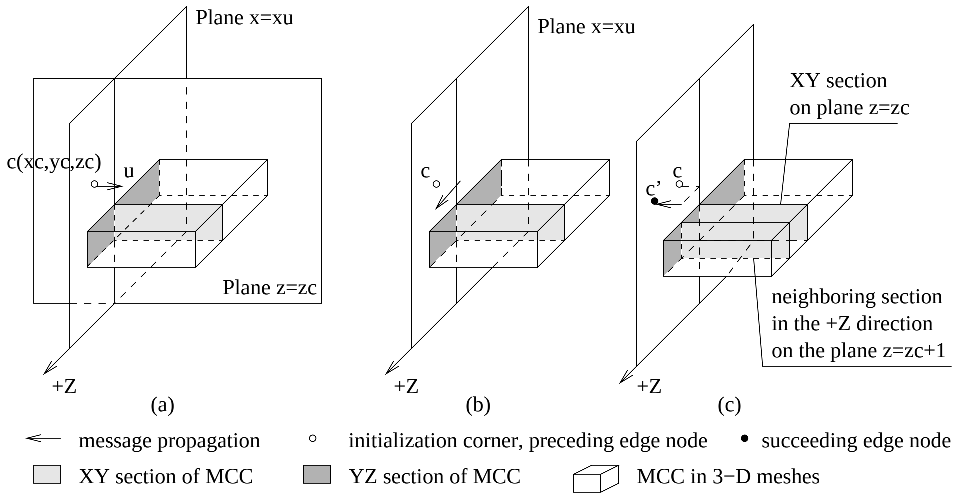

Figure 10. Identification of edge from node $c(x_c, y_c, z_c)$. (a) Phase one. (b) Phase two. (c) Phase three.

section, in a certain direction, say the $+Z$ direction in an $(+Y-X)$ edge for the $(+X/+Y/+Z)$-routing. In phase two, this message will be propagated along that path to the neighboring section. In phase three, it will route around that neighboring section to reach the corresponding corner. Once this message reaches that corner, those two corners in neighboring sections are identified as preceding and succeeding edge nodes and the path between them will be used for future edge construction.

For example, the $(+Y-X)$-edge of an MCC is defined by the $(+Y-X)$-corners of all its XY sections. The identification process for this $(+Y-X)$-edge starts from each $(+Y-X)$-corner. In phase one, from a $(+Y-X)$-corner $c(x_c, y_c, z_c)$ in the XY section $z = z_c$ in Figure 10 (a), a message will be sent to route around this section. When such a message passes through a node $u(x_u, y_u, z_c)$ with an unsafe neighbor in the $-Y$ direction, the identified information of the YZ section on the plane $x = x_u$ is used to find a neighboring section on plane $z = z_c + 1$. A neighboring section exists if z_c is not the minimum coordinate in the $+Z$ dimension in that YZ section. In phase two, the neighboring section is found and the message will go around the corresponding YZ section to the neighboring XY section (see Figure 10 (b)). In phase three, once the message arrives at the neighboring XY section, it will go around that section to reach its corresponding $(+Y-X)$ corner c'. At node c', c is identified as its preceding edge node and the information of the path to node c (see the dash link in Figure 10 (c)) is saved for future information propagation. It is noted that the message propagation may require several hops to detour the irregular fault region in each phase (see Figure 11).

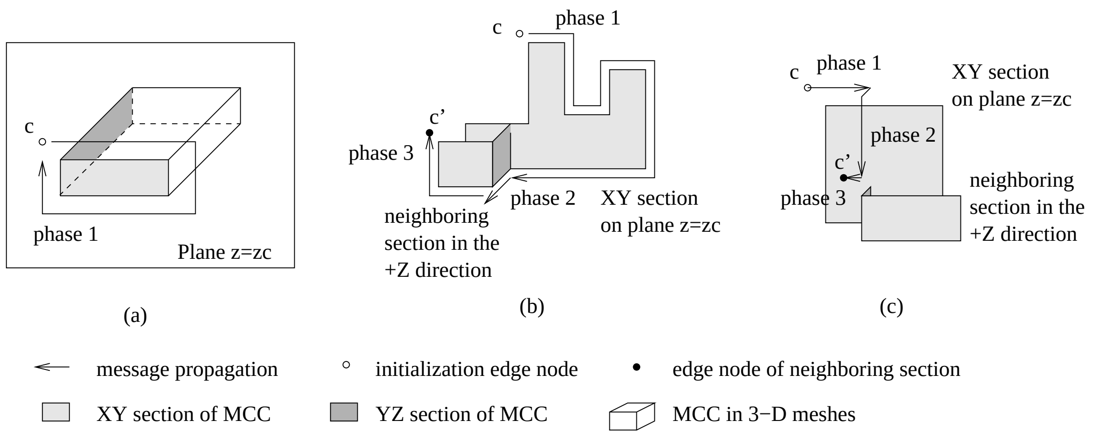

Figure 11. Some samples of identification of edge node c. (a) Identification of starting point of edge. (b) Complex case of phase one in finding neighboring section. (c) Complex case of phase two in reaching the neighboring section.

4.3 Edge construction

In phase one in the above identification process of an edge node, if the neighboring section is not found, that starting edge node without a succeeding edge node is identified as the starting point of the entire edge, and the corresponding section is identified as a surface of this MCC (see Figure 11 (a)). From that identified starting edge node u, a collection process is activated to collect all the links between preceding and succeeding edge nodes and form the entire edge. A message is sent along the paths linking each edge node and its preceding edge node. Such a propagation will continue until it reaches the end node (an edge node without any preceding node). At each edge node v it passes through, the section information is attached to the message. With the previously attached information, the area of MCC from the current node v to the starting point u, $M(v)$, is determined. The information of forbidden region $Q(v)$ and the critical region $Q'(v)$ can also be formed at node v. Figure 12 shows some samples of the determined information at edge nodes. It is noted that an edge node may have more than one preceding edge node (see node c'' in Figure 12 (a)). In that case, the collection process needs a multicasting to all the preceding nodes. It is also noted that each edge node cannot have more than one succeeding edge node due to the exclusion of the corresponding edge node in neighbor section.

4.4 Boundary construction

At each edge node u, say along a $(+Y-X)$-edge, after $M(u)$ is formed, the information of $M(u)$, $Q_Y(u)$, and $Q'_Y(u)$ will be propagated along the boundary, also called $(+Y-X)$-boundary, to block the routing from entering the region $Q_Y(u)$ in the $+X$ dimension if the destination is inside the critical

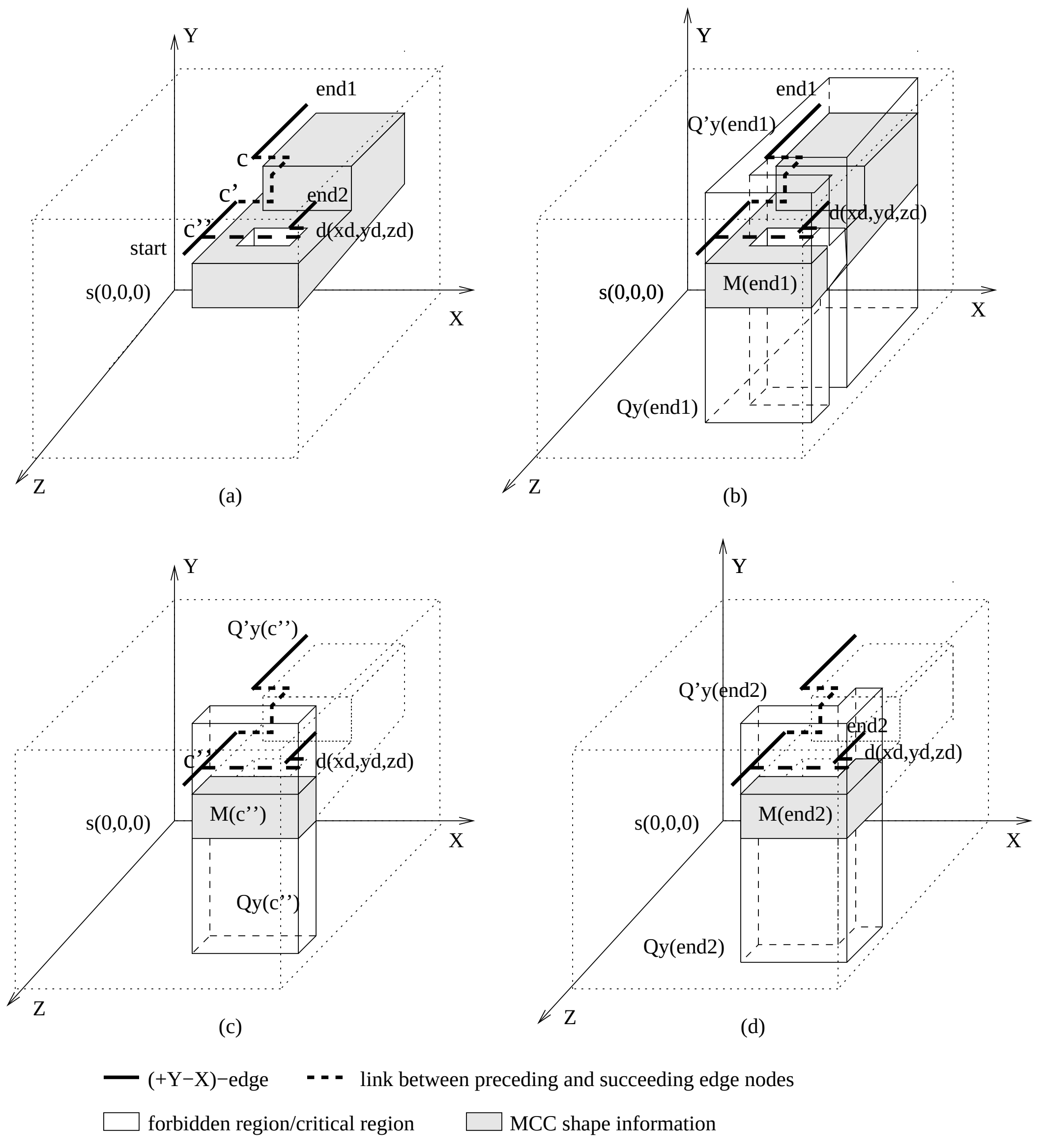

Figure 12. (a) Sample of constructed $(+Y-X)$-edge. (b) Information (M, Q_y, and Q'_y) formed at end edge node $end1$. (c) Information formed at an edge node c''. (d) Information formed at end edge node $end2$.

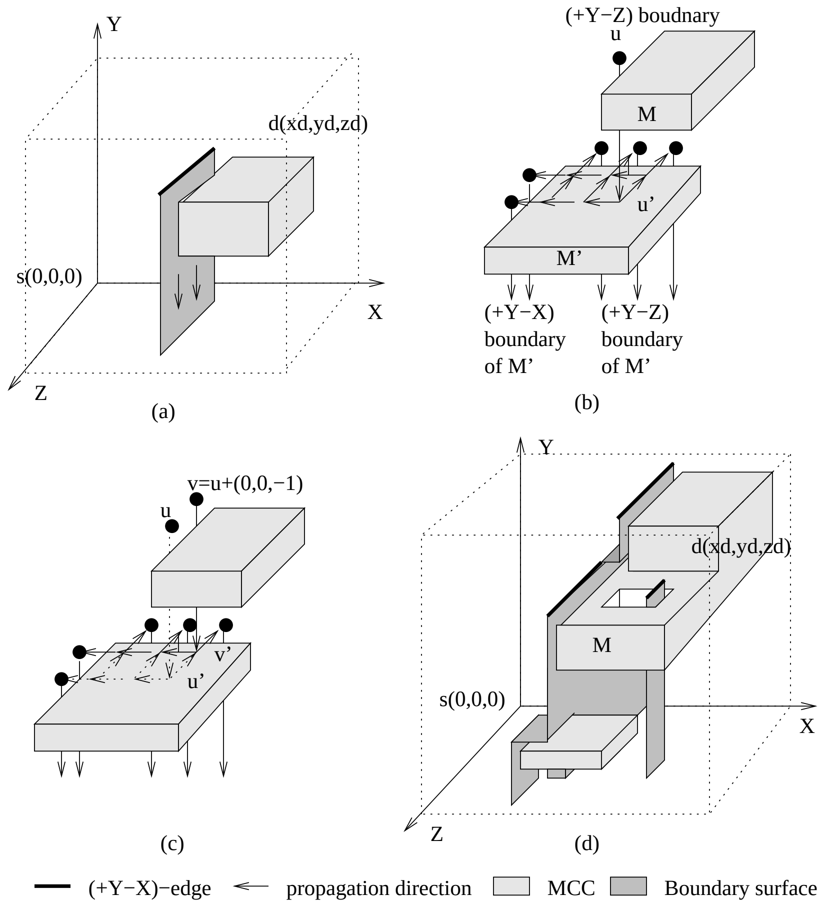

Figure 13. (a) Construction of a $(+Y-X)$-boundary. (b) Propagation of $M(u)$ at the intersection u' to cover the "nose" part. (c) $M(u)$ overwritten by $M(v)$. (d) A complete $(+Y-X)$-boundary constructed for MCC M.

Algorithm 5: Identification and boundary construction of an MCC in 3-D meshes

```
1. Identification of each 2-D section by using the identification process in Algorithm 2.

2. Identification of each edge: (a) a message is sent along each $XY$, $YZ$, or $XZ$ section from its starting corner to
   find the path to its neighboring section; (b) the message reaches the neighboring section along this path; (c) an
   identification process in Algorithm 2 is applied to reach the corresponding corner in this neighboring section,
   the succeeding edge node.

3. Edge construction: If no succeeding edge node is found, the edge node itself will be identified as the starting
   node of the entire edge. From this node, a message will propagate along the links from each edge to its preceding
   node(s) and such a propagation will continue until it reaches the end (an edge node without any preceding edge
   node). At each edge node, the 2-D section information will be attached to the message. With the information
   previously attached, the concerning MCC part $M(c)$ can be determined. The forbidden region $Q(c)$ and the
   critical region $Q'(c)$ can also be formed.

4. Boundary construction: After $M(u)$, $Q(u)$, and $Q'(u)$ is formed at an edge node $u$, say along the $(+Y-X)$-
   edge, the information $M(u)$, $Q_Y(u)$, $Q'_Y(u)$ will be propagated along its $(+Y-X)$-boundary in the $-Y$
   direction. If it intersects with another MCC $M$, it will join the boundary of the "nose" part of $M$ and merge its
   forbidden region $Q_Y$ into $Q_Y(u)$.
```

region $Q'_Y(u)$. Initially, a message carrying the information is sent from node u along the $-Y$ dimension (see Figure 13 (a)). Once this message intersects with another MCC M' at node $u'(x_{u'}, y_{u'}, z_{u'})$, it will propagate along the surface of M' in the $-Z$ and $-X$ directions to join its $(+Y-X)$ boundary propagation and $(+Y-Z)$ boundary propagation. From each joint point, the forbidden region Q_Y of M' will merge into $Q_Y(u)$: $Q_Y(u) = Q_Y \cup Q_Y(u)$ (see Figure 13 (b)). To avoid propagation of redundant information along the boundaries of the intersected MCC, we have the following superseding rule:

- **Superseding rule**: A propagation of an edge node overwrites the propagation of its succeeding edge node because the MCC information of the former one contains that of the latter one.

Figure 13 (c) shows that the propagation of an edge node u is overwritten by the propagation of its preceding edge node on the surface of the intersected MCC M'. A sample of a complete $(+Y-X)$ boundary is shown in Figure 13 (d). The whole procedure to collect and distribute MCC information is shown in Algorithm 5.

5 Sufficient and necessary condition for the existence of a Manhattan distance path in 3-D meshes

In this section, a sufficient and necessary condition with boundary information is presented to ensure the existence of a Manhattan distance path in 3-D meshes.

After the boundary construction, a boundary node will have the region information $M(c)$, the forbidden region information $Q(c)$ ($Q_X(c)$, $Q_Y(c)$, or $Q_Z(c)$), and the critical region information $Q'(c)$ ($Q'_X(c)$,

$Q'_Y(c)$, or $Q'_Z(c)$). When a routing message arrives and its destination d is inside $Q'(c)$, the boundary line can be used as a part of path in the Manhattan routing to route around $M(c)$. Thus, we have the following sufficient and necessary condition for the existence of a Manhattan distance path in 3-D meshes:

Lemma 2: *A routing does not have a Manhattan distance path if and only if there exists an MCC for which (a) $d \in Q'_X$ and $s \in Q_X$, (b) $d \in Q'_Y$ and $s \in Q_Y$, or (c) $d \in Q'_Z$ and $s \in Q_Z$.*

Proof: Along the path in $+X/+Y/+Z$ routing from s to d, if a Manhattan distance path exists, any intermediate node u should have a length $D(s,u)$ path to s (condition (a)). Among all the nodes in [$0 : x_d$, $0 : y_d$, $0 : z_d$] meeting such a satisfaction, only a node u which has a length $D(u,d)$ path to d can be selected to form the Manhattan distance path from s to d (condition (b)). Assume $M(c_x)$ is the closest MCC that $d \in Q'_x(c_x)$. Respectively, $M(c_y)/M(c_z)$ is the closest MCC that $d \in Q'_y(c_y)/Q'_z(c_z)$. Define the region of the Manhattan distance paths ($RMDP$) that includes every node meeting the satisfaction of both conditions (a) and (b). We have its region: [$0 : x_d$, $0 : y_d$, $0 : z_d$]$-Q_X(c_x) - Q_Y(c_y) - Q_Z(c_z)$. Based on the construction of boundaries for $Q_X(c_x)$, $Q_Y(c_y)$, and $Q_Z(c_z)$, a Manhattan distance path from d to s (i.e., from s to d) exists iff $s \in RMDP$; that is, $s \notin Q_X(c_x)$, $Q_Y(c_y)$, and $Q_Z(c_z)$. ■

Theorem 2: *A routing does not have a Manhattan distance path if and only if there exists an MCC for which (a) $d \in Q'_X$ and neither its $(+X - Y)$-boundary nor its $(+X - Z)$-boundary intersects with the surface [$0 : 0$, $0 : y_d$, $0 : z_d$], (b) $d \in Q'_Y$ and neither its $(+Y - X)$-boundary nor its $(+Y - Z)$-boundary intersects with the surface [$0 : x_d$, $0 : 0$, $0 : z_d$], or (c) $d \in Q'_Z$ and neither its $(+Z - X)$-boundary nor its $(+Z - Y)$-boundary intersects with the surface [$0 : x_d$, $0 : y_d$, $0 : 0$].*

Proof: When we find an MCC and its $d \in Q'_X$ ($/Q'_Y/Q'_Z$), $s \in Q_X$ ($/Q_Y/Q_Z$) iff the $(+X - Y)$-boundary ($/(+Y - Z)$-$/(+Z - X)$-boundary) and $(+X - Z)$-boundary ($/(+Y - X)$-$/(+Z - Y)$-boundary) intersect with the surface [$0 : 0$, $0 : y_d$, $0 : z_d$](/[$0 : x_d$, $0 : 0$, $0 : z_d$]/[$0 : x_d$, $0 : y_d$, $0 : 0$]). With Lemma 2, the statement is easy to prove. ■

6 Boundary-information-based routing under the MCC model in 3-D meshes

Based on Theorem 2, Wu's routing in [19] in 3-D meshes is extended to a routing under the MCC model (see Algorithm 6). Such a routing can find a Manhattan distance path from the source and destination nodes whenever this path exists.

Similar to the routing in 2-D meshes, the feasibility check is first activated at the source s to make sure that a Manhattan distance path exists. Otherwise, the routing will stop. First, the source node s will play the role of the destination in Theorem 2. Three detection messages will be sent from s along the $+X$, $+Y$, and $+Z$ directions. If any message, say the one along the $+Y$ direction (see Figure 14 (c)), intersects another MCC, it will join the $(-Y + X)$- and $(-Y + Z)$ boundaries as boundary construction, i.e., the $(+Y - X)$- and $(+Y - Z)$ boundaries in the 3-D meshes after 180 degree rotation (see Figure 14 (b)). If any copy of

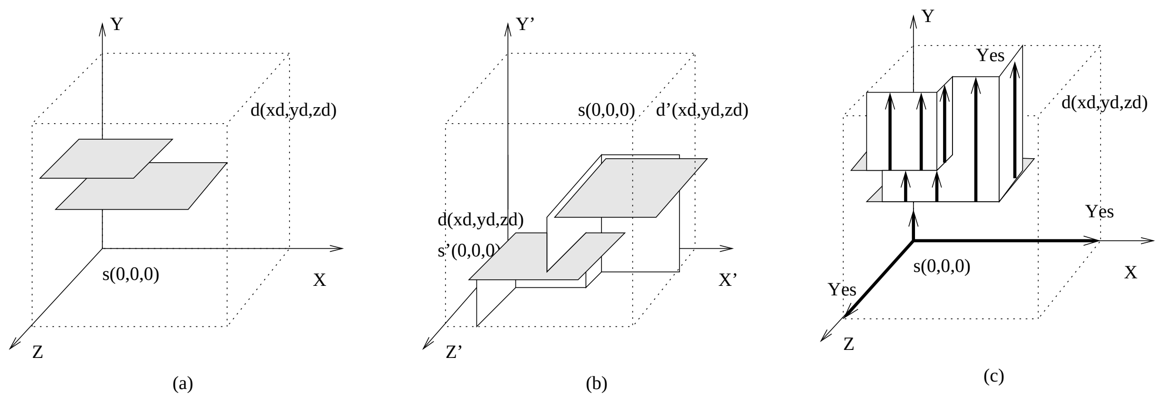

Figure 14. Sample of detection message propagation. (a) Fault configuration. (b) Configuration after 180 degree self-rotation and the corresponding $(+Y-X)$- and $(+Y-Z)$-boundaries in the meshes. (c) Propagation of detection message along the $(+Y-X)$- and $(+Y-Z)$-boundaries in (b).

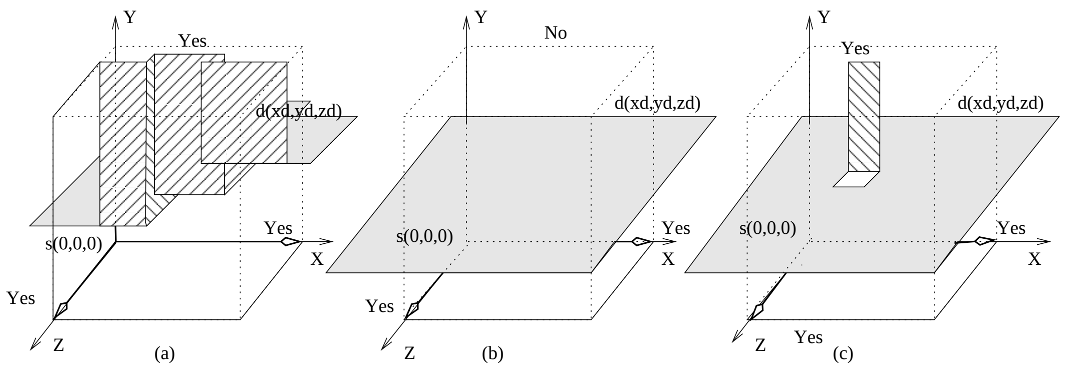

Figure 15. Samples of feasibility check.

Algorithm 6: Routing from $s(0,0,0)$ to $d(x_d, y_d, z_d)$ ($x_d, y_d, z_d \geq 0$)

1. Feasibility check: At source s, send detection messages along the $+X$, $+Y$ and $+Z$ directions. When a message, say the one along the $+X$ direction, intersects another MCC, it will join the $(-X+Y)$- and $(-X+Z)$-boundaries as boundary construction. The source node will check if these three detection messages can reach the surfaces [$x_d : x_d$, $0 : y_d$, $0 : z_d$], [$0 : x_d$, $y_d : y_d$, $0 : z_d$] and [$0 : x_d$, $0 : y_d$, $z_d : z_d$] respectively. If any one of these three surfaces cannot be reached, stop the routing since there is no Manhattan distance path.
2. Routing decision and message sending at the current node u, including the source s:
   (a) Add all the preferred directions into the set of candidates of forwarding directions F and find all the recorded MCCs.
   (b) For each MCC found in the above step, exclude direction from F if the destination is in the critical region and the neighbor of u along this direction is inside the forbidden region.
   (c) Apply any fully adaptive and minimal routing process to select a forwarding direction from set F.
   (d) Forward the routing message along the selected forwarding direction to the next node.

this message reaches the surface [$0 : x_d$, $y_d : y_d$, $0 : z_d$], it will return "YES" back to s. If s receives all three "YES" returns, based on the sufficient and necessary condition in Theorem 2, a Manhattan routing from d to s exists; that is, a Manhattan routing from s to d exists. Figure 15 shows some samples of check results.

At each node, including the source node s, the routing process basically has three preferred directions: the $+X$, $+Y$, and $+Z$ directions. The boundary information of an MCC with the destination in the critical region will help the routing process avoid entering the forbidden region by excluding the corresponding preferred direction from the candidates for the forwarding direction. After that, any fully adaptive and minimal routing process could be applied to pick up the forwarding direction and forward the routing message along this direction to the corresponding neighbor. Figure 16 shows some samples of routing under our MCC model in 3-D meshes.

7 Monte Carlo Simulation Results

We developed a simulation to test the effect of our new MCC model on the performance of routing in 3-D meshes, in terms of the percentage of successful Manhattan routing. The cost for construction of MCC model was also tested, in terms of (a) the number of unsafe nodes whose communications are disabled, and (b) the number of rounds of information exchanges and updates needed in a synchronous round-based system (i.e, the speed that the construction process converges). To show that our new information model is cost-effective, the results are compared with those under the cuboid fault block (CFB) model [19], which are the best results known to date.

The simulation is developed in the Monte Carlo estimate procedure. That is, we simply take many

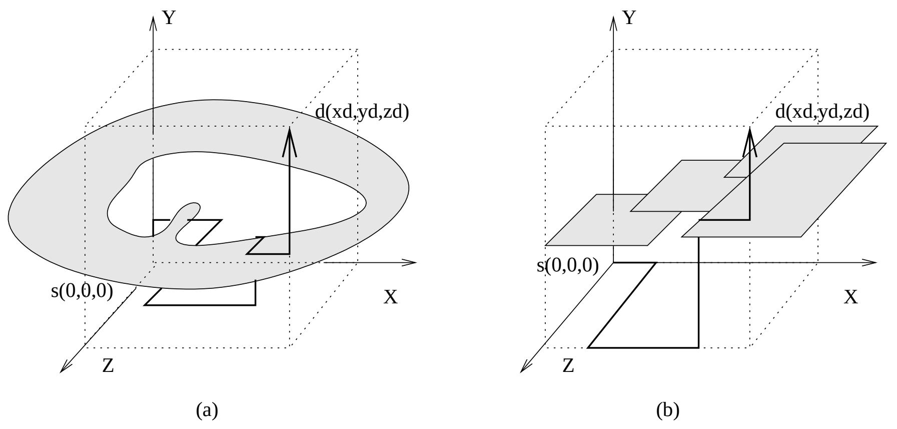

Figure 16. Samples of routing under the MCC model.

samples, independently, and average them to get results close to the true value. In the simulation, we take many samples ($> 10,000$) of a 30×30×30 3-D mesh. In each case, a certain number of faults are uniformly distributed. After this, the MCC information and CFB information are constructed. And then, many routing cases with a pair of the source and the destination randomly generated for each are tested in different routings. We only test those cases when a Manhattan distance path exists between the source and the destination. When the number of faults is larger than 500, applying the CFB model may disable all the non-faulty nodes in the networks and furthermore disable all their communication while many nodes and their data transmission are kept enabled under our MCC model in the same fault configuration. Therefore, only the results when the number of faults is no more than 500 are compared in a fair way in Figure 17. It is noted that all the schemes presented in this paper can also be applied in an asynchronous system; however, to make the discussion simple, we do not pursue the relaxation here.

Compared with the CFB model, the MCC model has much fewer unsafe nodes and can enable much more end-to-end communication in the entire network, especially when the number of faults is larger than 400. As shown in Figure 17 (b), the construction process of our MCC model also converges very fast in terms of the number of rounds needed. It is noted that in our random fault generator, the faults will be distributed sparsely when only a small amount of faults are generated. When the number of faults is less than 10, both MCCs and CFBs contain faults only; that is, no unsafe node is disabled and the costs under two different models are the same. Even when the number of faults reaches 100, few unsafe nodes are disabled in both models. Therefore, the cost is nearly the same for our new MCC model and the CFB model when the number of faults is limited. However, the cost of MCC model is always less than that of the CFB model.

Our new MCC routing proposed in this paper can find a Manhattan distance path from s to d whenever it exists. However, it needs a broadcast along surfaces in the feasible check process. Wu presented a simple feasibility check for the Manhattan routing in [19], which only requires message propagations along three rays. However, if the existence of a Manhattan distance path is ensured in the check process, then a

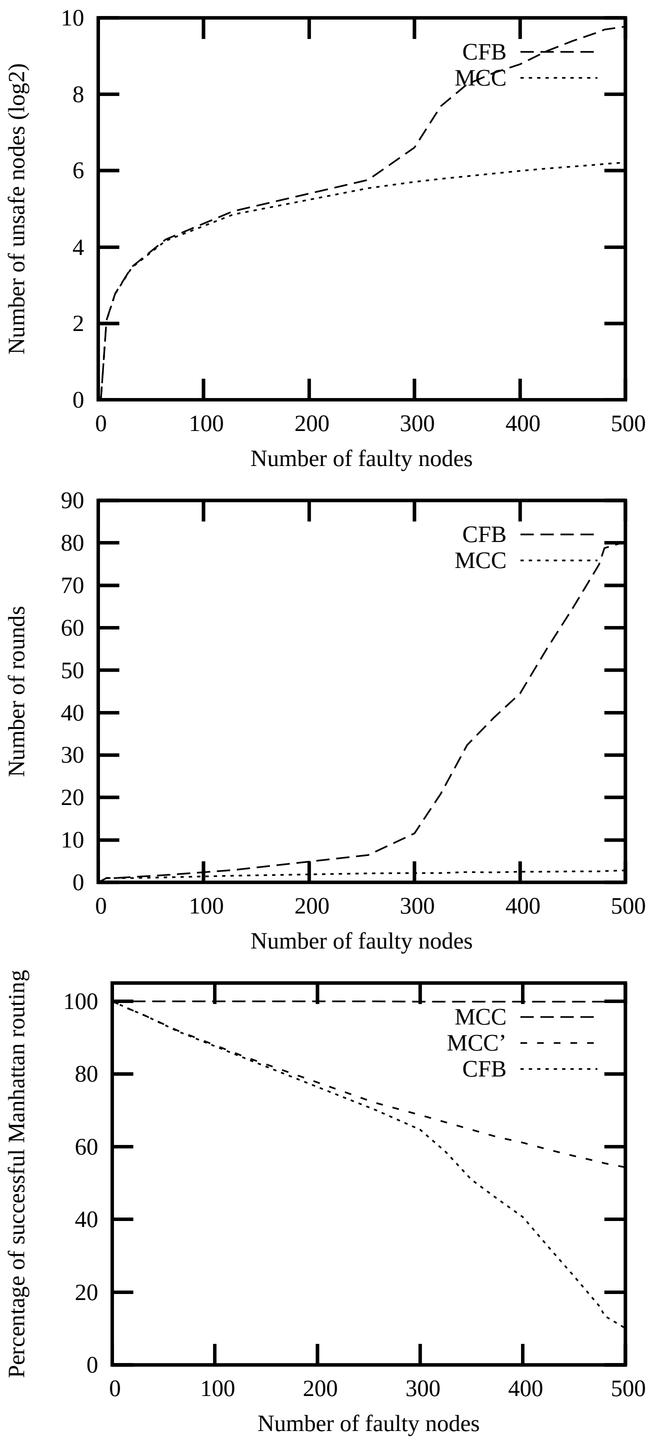

Figure 17. (a) Number of unsafe nodes (lg), (b) number of rounds of information exchanges and updates in labeling process, (c) percentage of successful Manhattan routing.

Manhattan routing can be conducted, but not the other way around. To reduce the cost for the feasibility check, we replace our check process in Algorithm 6 by that in [19]. The corresponding routing, denoted by MCC', is a simple version of our MCC routing. In the routing [19] using cuboid fault block (denoted by CFB), the proposed Manhattan routing under the MCC model (denoted by MCC), and its simplified version MCC' routing, whenever the feasibility check is passed, a Manhattan routing can be guaranteed. Thus, in these routings, the success of Manhattan routing depends on the passing rate in the check process. Figure 17 (c) shows the percentage of passing cases, i.e., the rate of successful Manhattan routing. The Monte Carlo simulation results show that our new MCC routing can always find a Manhattan distance path whenever it exists (rate of successful Manhattan routing $= 100\%$). In MCC' routing, the simple check process is not able to identify every existing Manhattan distance path. Thus, MCC' routing will have less success than MCC routing. However, by using the MCC information, the rate is still higher than that of CFB routing. These results show the significance of our MCC information model in 3-D meshes.

8 Conclusion

In this paper, we have proposed the minimal fault region MCC for the Manhattan routing, also called the minimal routing, in 3-D meshes by considering the positions of the source and destination. Using a non-faulty node contained in an MCC will definitely make the routing non-minimal. If no Manhattan distance path exists under the MCC model, there will be absolutely no Manhattan routing. By using our boundary information, we have provided a fully distributed process in both 2-D meshes and 3-D meshes to collect and distribute the MCC information. Based on this boundary information, our routing will guarantee a Manhattan distance path for the data communication in a system without using global information. Our estimates in the Monte Carlo method have shown the improvement under our MCC model in 3-D meshes by comparing with the best currently known approaches. In our future work, we will study the performance of our new routing. The maximum achieved throughput among the safe nodes that can communicate under the MCC model will be analyzed and tested. We will also extend our results to dynamic networks in which any of the components can become faulty during the routing process. As a result, the minimal fault region can change its shape dynamically and the corresponding boundaries will be adjusted frequently. Next, our results will be extended to higher dimension networks.

9 Acknowledgments

The authors wish to thank the anonymous reviewers for the many suggestions that helped improving the presentation quality of the article. The work was supported in part by NSF grants ANI 0073736, CCR 0329741, CNS 0422762, CNS 0434533, EIA 0130806, CNS 0531410, and CNS 0626240.

References

[1] Definition of manhattan distance. Available at http://www.nist.gov/dads/HTML/manhattanDistance.html.

[2] Definition of monte carlo method. Available at http://en.wikipedia.org/wiki/Monte_carlo_method.

[3] F. Allen et al. Blue gene: A vision for protein science using a Petaflop supercomputer. *IBM Systems Journal*. 40, 2001, 310-327.

[4] R. V. Boppana and S. Chalasani. Fault tolerant wormhole routing algorithms for mesh networks. *IEEE Transactions on Computers*. Vol. 44, No. 7, July 1995, 848-864.

[5] 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.

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

[7] A. A. Chien and J. H. Kim. Planar-adaptive routing: Low-cost adaptive networks for multiprocessors. *Journal of ACM*. 42, (1), January 1995, 91-123.

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

[9] R. L. Hadas and E. Brandt. Origin-based fault-tolerant routing in the mesh. *Future Generation Computer Systems*. Vol. 11, No. 6, October, 1995, 603-615.

[10] Z. Jiang and J. Wu. A limited-global-information model for dynamic routing in 2-D meshes. *Proc. of* 3^{rd} *Workshop on Parallel and Distributed Scientific and Engineering Computing with Applications (PDSECA-02), in conjunction with IPDPS'02.* 2002, CD-ROM.

[11] R. K. Koeninger, M. Furtney, and M. Walker. A shared memory MPP from Cray research. *Digital Technical Journal*. Vol. 6, No. 2, Spring 1994, 8-21.

[12] 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*. January 1995, 102-111.

[13] L. Sheng and J. Wu. Maximum-shortest-path (msp) is not optimal for a general n $\times$ n torus. *IEEE Transactions on Reliability*. Vol. 52, No. 1, 2003, 22-25.

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

[15] D. Wang. A rectilinear-monotone polygonal fault block model for fault-tolerant minimal routing in meshes. *IEEE Transactions on Computers.* Vol. 52, No. 3, March 2003.

[16] 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), February 2000, 149-159.

[17] J. Wu. A fault-tolerant adaptive and minimal routing scheme in n-D meshes. *The Computer Journal*. Vol. 45, No. 3, 2002, 349-363.

[18] J. Wu. Maximum-shortest-path (msp): An optimal routing policy for mesh-connected multicomputers. *IEEE Transactions on Reliability*. Vol. 48, No. 3, 1999, 247-255.

[19] J. Wu. A simple fault-tolerant adaptive and minimal routing approach in 3-D meshes. *Journal of Computer Science and Technology*. Vol. 18, No. 1, 2003, 1-13.

[20] J. Wu and Z. Jiang. On constructing the minimum orthogonal convex polygon in 2-D faulty meshes. *Proc. of International Parallel and Distributed Processing Symposium (IPDPS)*. 2004, (CD-ROM).

[21] D. Xiang. Fault-tolerant routing in hypercube multicomputers using local safety information. *IEEE Transactions on Parallel and Distributed Systems*. Vol. 12, No. 9, 2001, 942-951.

[22] D. Xiang, A. Chen, and J. Wu. Reliable broadcasting in wormhole-routed hypercube-connected networks using local safety information. *IEEE Transactions on Reliability*. Vol. 52, No. 2, 2003, 245-256.