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

Zhen Jiang, *Member, IEEE*, Jie Wu, *Senior Member, IEEE*, and Dajin Wang, *Member, IEEE*

***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 source & destination when the new faulty components are constructed. Specifically, all faulty nodes will be contained in some disjoint fault-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 & distribute MCC information to a limited number of nodes along so-called boundaries. Moreover, a sufficient & 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 information of boundary in routing-decisions at the intermediate nodes. The results of our Monte-Carlo-estimate show substantial improvement of the new fault-information model in the percentage of successful Manhattan-routing conducted in 3-D meshes.**

***Index Terms*—Adaptive routing, distributed algorithms, fault-tolerant routing, minimal routing, 3-D meshes.**

ACRONYM[1]

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

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) & (x, y'), and respectively we have the one along the X dimension $[x : x', y : y]$
$M(c)$	a 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 2-D section of a 3-D MCC 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 a 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

Manuscript received January 29, 2006; revised September 20, 2006; January 25, 2007; May 10, 2007; and July 24, 2007; accepted July 24, 2007. This work was supported in part by NSF Grants ANI 0073736, CCR 0329741, CNS 0422762, CNS 0434533, EIA 0130806, CNS 0531410, and CNS 0626240. Associate Editor: G. Levitin.

Z. Jiang is with the Computer Science Department, West Chester University, West Chester, PA 19383 USA (e-mail: zjiang@wcupa.edu).

J. Wu is with the Computer Science & Engineering Department, Florida Atlantic University, Boca Raton, FL 33431 USA (e-mail: jie@cse.fau.edu).

D. Wang is with the Computer Science Department, Montclair State University, Upper Montclair, NJ 07043 USA (e-mail: wang@pegasus.montclair.edu).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TR.2007.909768

[1]The singular and plural of an acronym are always spelled the same.

I. INTRODUCTION

IN a multi-computer-system, a collection of processors, or nodes, work together to solve problems in large-scale applications. These nodes communicate data, and coordinate their efforts by sending & receiving packets through the underlying

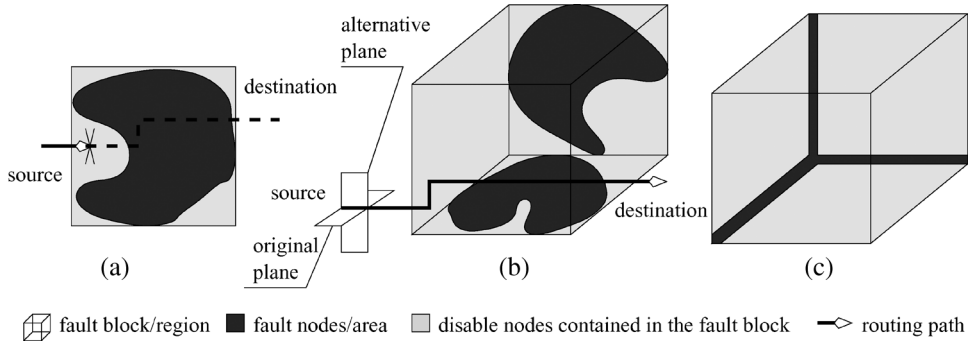

Fig. 1. (a) Routing blocked from entering a rectangular fault-block in 2-D meshes; (b) the use of alternative plane in Manhattan-routing after the block in the original plane; (c) a sample of cuboid fault-block that contains too many non-faulty nodes.

network of communication. Thus, the performance of such a multi-computer-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 multi-computers. Basically, routing is the process of transmitting data from one node, called the source, to another node, called the destination. 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 topologies of networks for multi-computer-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 multi-computers were built based on the 3-D meshes [3], [11]. As the number of nodes in a mesh-connected multi-computer-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]–[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 rectangle-shaped 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 Fig. 1(a)). When the rectangular fault-block is extended to a cuboid block in 3-D meshes, most routings blocked by the rectangle-shaped section of this block on one plane can find a Manhattan-distance-path along an alternative plane (see Fig. 1(b)). That is, some non-faulty nodes inside the cuboid fault-block can be used in Manhattan-routing, and are unnecessarily disabled to form the cuboid fault-region. In the worst case, a few nodes can disable the entire mesh, and further, block any communication (see Fig. 1(c)).

In this paper, we focus on the minimal fault-region for Manhattan-routing, in which the non-faulty nodes contained are reduced as much as possible. Because the global information is not suitable for complex & large-scale 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 MCC model for Manhattan-routing in 3-D meshes. A node will be included in a MCC of 3-D meshes iff 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 source & destination when the MCC is constructed.
- 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 information of boundaries 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 method of Monte-Carlo-estimate [2] to determine the number of non-faulty nodes included in MCC 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 model of rectangular fault-blocks. After each non-faulty node in Wang's MCC is labeled, the shape of the fault-region is identified in a new distributed process. Then, the identified information of this MCC will be propagated to form 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 & necessary condition for the ex-

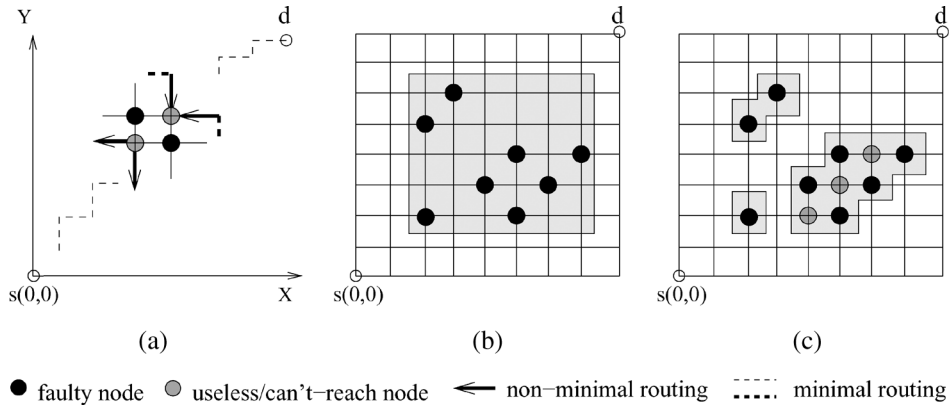

Fig. 2. (a) Definition of useless & can't-reach nodes; (b) sample of rectangular fault-block; (c) the corresponding MCC.

istence of a Manhattan-distance-path, which has been rewritten without using global information, is used to ensure the Manhattan-routing at the source. In phase two, the routing at each intermediate node, including the source, will forward the packet to the next node along the path. It uses the information of boundaries to keep the route minimal while not missing any Manhattan-distance-path.

Second, the above 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 the MCC. Via the propagation of information along 2-dimensional surfaces, which is implemented through the transmission of messages between two neighboring nodes along one of those three dimensions X, Y, and Z, the shape of MCC is identified, and the corresponding boundaries are constructed. With the information of these boundaries, 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 II introduces some necessary notations & preliminaries. Section III provides our construction for the boundaries of MCC in 2-D meshes, and the corresponding boundary-information-based routing. In the same section, Wang's sufficient & necessary condition for the existence of a Manhattan-distance-path is rewritten without using global information. Section IV presents the MCC model in 3-D meshes, and extends the corresponding processes in 2-D meshes to 3-D meshes. A labeling scheme for non-faulty nodes in each MCC, and the construction of boundaries for a MCC, are presented. Section V provides a sufficient & necessary condition for the existence of a Manhattan-distance-path in 3-D meshes. In Section VI, our Manhattan-routing in 3-D meshes is provided. Its improvement compared with the best existing routing is shown in the results of our Monte-Carlo-estimate in Section VII. Section VIII concludes this paper, and provides directions for future research.

II. 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 $(u_1, u_2, \ldots, u_n)$, where $0 \leq u_i \leq k-1$. Two nodes $(v_1, v_2, \ldots, v_n)$ & $(u_1, u_2, \ldots, u_n)$ are neighbors if their addresses differ in one & 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 & v, $D(u,v)$, is equal to $|x_v - x_u| + |y_v - y_u|$. Assume node s is the source, u is the current node, and d is the destination. 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 s to destination 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 in 2-D meshes is also called $(+X/+Y)$-routing. In this paper, the Manhattan-routing & $(+X/+Y)$-routing are used alternatively. Similarly, in a 3-D mesh, (0,0,0) is the source, $u(x_u, y_u, z_u)$ is the current node, $d(x_d, y_d, z_d)$ is the destination where $x_d, y_d, z_d \geq 0$, and the Manhattan-distance between two nodes u & v, $D(u,v)$, is equal to $|x_v - x_u| + |y_v - y_u| + |z_v - z_u|$. (x_u+1, y_u, z_u), (x_u-1, y_u, z_u), (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* & *can't-reach* nodes (see Fig. 2(a)): A node labeled useless is a node such that when 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 move in the $-X$ or $-Y$ direction must be taken, making the routing non-minimal. The labels of faulty, useless, and can't-reach nodes can be determined through a labeling procedure. All faulty, useless, and can't-reach nodes are also called unsafe nodes. The

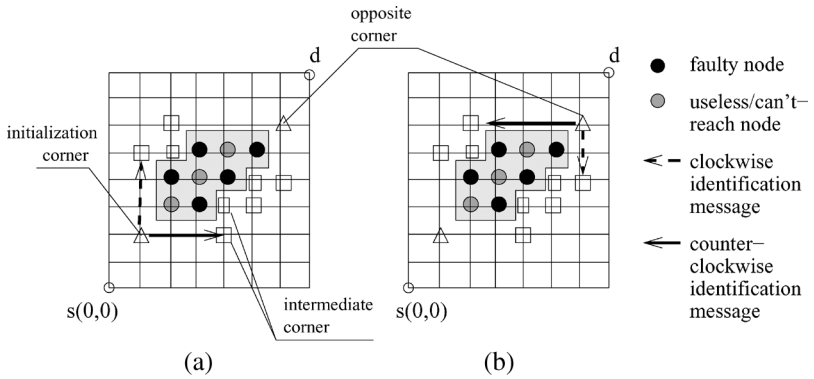

Fig. 3. (a) Identification activated at the initialization-corner; (b) identified information re-sending.

labeling procedure is given in Algorithm 1, and it can quickly identify the non-faulty nodes in MCC. Each active node collects its neighbors' status, and updates its own status. Only those affected nodes update their status. Eventually, neighboring unsafe nodes form a MCC. Fig. 2(a) shows the idea of the definition of useless & can't-reach nodes. Fig. 2(c) shows some samples of MCC for the routing from (0,0) to (x_d, y_d) where x_d, $y_d \geq 0$.

Algorithm 1: Labeling procedure of MCC for the routing from (0,0) to (x_d, y_d) where 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 & $+Y$-neighbor are faulty or useless, then u is labeled *useless*.
3) If node u is safe, but its $(-X)$-neighbor & $(-Y)$-neighbor are faulty or can't-reach, then 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.

III. 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, but also such a routing can be achieved successfully by routing-decisions at intermediate nodes along the path. The new routing provided in this section can find a Manhattan-distance-path from the source to the destination whenever such a path exists.

A. Corner & Boundary of MCC in 2-D Meshes

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

From that initialization-corner, two messages, one clockwise and one counter-clockwise, are initiated. They will be sent to its two edge-neighbors. Each message is used to identify partial region of MCC. 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 $u(x_u, y_u)$, the address of node (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 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 then 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. Fig. 3 shows a sample of the identification for a MCC in 2-D meshes.

A MCC has only one pair of initialization-corner $c(x_c, y_c)$ & opposite-corner $c'(x_{c'}, y_{c'})$. If these two messages cannot meet at that opposite corner in the process of identification, 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 boundaries. If only one message is received at the initialization-corner, the other has been discarded during the propagation, and this message should also be discarded. Usually, a TTL (time-to-live) is associated with each message, and the corresponding message will be discarded when the time expires.

In 2-D meshes, a MCC with initialization-corner c is denoted by $M(c)$. The $(+X/+Y)$-routing 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 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, two

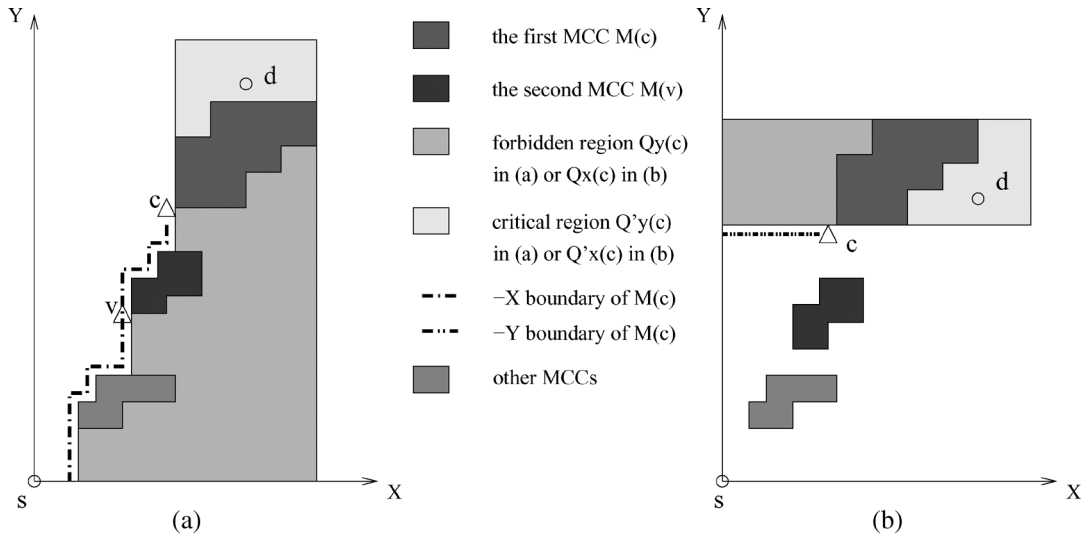

Fig. 4. Samples of boundary-construction under the MCC model in 2-D meshes.

boundaries will be constructed when both two messages of identification are received at node c. In such a construction, two messages will be initiated. One message carrying the information $M(c)$, $Q_Y(c)$, and $Q'_Y(c)$ will propagate to all the nodes along the line $x = x_c$ until it reaches the edge of this 2-D mesh. These nodes form the so-called $(-X)$-boundary. When this boundary 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 Fig. 4(a)). Similarly, another propagation, carrying $M(c)$, $Q_X(c)$, and $Q'_X(c)$, will construct the $(-Y)$-boundary. It will go along line $y = y_c$, and make a turn in the $-Y$ direction if necessary (see Fig. 4(b)). The whole procedure is shown in Algorithm 2.

Algorithm 2: Identification & boundary-construction for a 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 of MCC $M(c)$: (a) From node c, two messages, one clockwise and one counter-clockwise, are sent along the edge-nodes of $M(c)$ until they reach node c'. (b) Information of the partial region of $M(c)$, including the address 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 these two messages are both received. Meanwhile, the forbidden-regions ($Q_X(c)$ & $Q_Y(c)$), and critical-regions ($Q'_X(c)$ & $Q'_Y(c)$), are identified.
3) Construction of $(-X)$- / $(-Y)$-boundary of $M(c)$: A boundary-construction is activated at node c after it receives two messages in the above process of identification. The information of $M(c)$, $Q_Y(c)$ / $Q_X(c)$, and $Q'_Y(c)/Q'_X(c)$ is propagated along the 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 $M(c)$, $Q_Y(c)/Q_X(c)$.

B. Sufficient & Necessary Condition for the Existence of a Manhattan-Distance-Path in 2-D Meshes

The MCC model includes many fewer non-faulty nodes in its fault-region than the conventional model in 2-D meshes. Many non-faulty nodes that would have been included in rectangular fault-blocks now can become candidate nodes of routing. As a matter of fact, MCC is the "ultimate" minimal fault-region; that is, no non-faulty node contained in a MCC can 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 & 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 a MCC $M(c)$ that a) $s \in Q_X(c) \wedge d \in Q'_X(c)$, or b) $s \in Q_Y(c) \wedge d \in Q'_Y(c)$.

Theorem 1: A routing does not have a Manhattan-distance-path iff there exists a 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 $[0 : 0, 0 : y_d]$.

C. Boundary-Information-Based Routing Under the MCC Model in 2-D Meshes

Wu proposed a minimal, adaptive routing in n-D meshes in [17]. It can easily be extended to a routing in 2-D meshes under

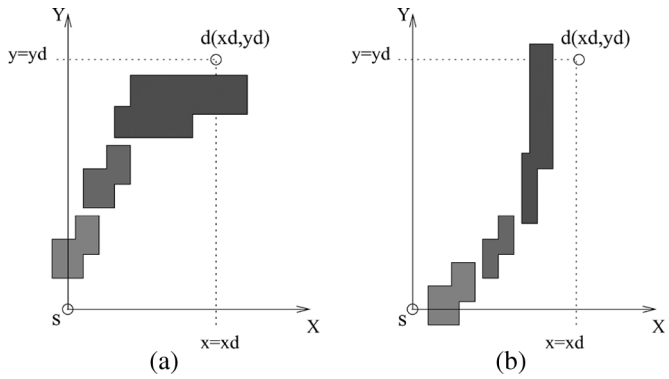

Fig. 5. (a) Type-I sequence of MCC; (b) type-II sequence of MCC.

the MCC model (see Algorithm 3). In this routing, at the source s, a check is first activated to make sure that the Manhattan-routing exists. Otherwise, the routing will stop. First, after a self-rotation of 180°, the source s will play the role of the destination in Theorem 1. It sends two messages of detection: one along the $(-X)$-boundary from s (i.e., the $(+X)$-boundary in the original mesh before the rotation), and one along the $(-Y)$-boundary from s (i.e., the $(+Y)$-boundary in the original mesh). The first one is to check if a type-I sequence [15] (see Fig. 5(a)) 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 (see Fig. 5(b)) exists or not. If the source s knows both segments can be reached, based on the sufficient & necessary condition in Theorem 1, a Manhattan-routing from d to s exists. That is, a Manhattan-routing from s to d exists.

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

1) Check of feasibility: At source s, send two messages of detection (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 because there is no Manhattan-distance-path.
2) Routing-decision 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 each recorded MCC.
   b) For each $M(c)$ found, remove $+Y/+X$ direction from F if $d \in Q_X(c)/Q_Y(c)$.
   c) Apply any fully-adaptive, minimal routing to select a forwarding direction from set F.
   d) Forward the packet along the selected direction to the next node.

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

IV. MCC Model in 3-D Meshes

In this section, we present our solution for constructing MCC, and propagating the corresponding 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. After that, the information of 2-D sections is collected along the edges in the edge-construction. With this information collected, the region is identified as a 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.

A. Labeling Process

A useless node u in a MCC in 2-D meshes has two useless or faulty neighbors, one in each of 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, one in each of the $+X$, and $+Y$ directions, the routing can still route around the fault-region in the $+Z$ direction. Therefore, a non-faulty node is useless in 3-D meshes iff it has three useless or faulty neighbors, one in each of the $+X$, $+Y$, and $+Z$ directions. Similarly, a non-faulty node is can't-reach iff it has three can't-reach or faulty neighbors, one in each of 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, then u is labeled *useless*.
3) If node u is safe, but its $(-X)$-neighbor, $(-Y)$-neighbor, and $(-Z)$-neighbor are faulty or can't-reach, then 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.

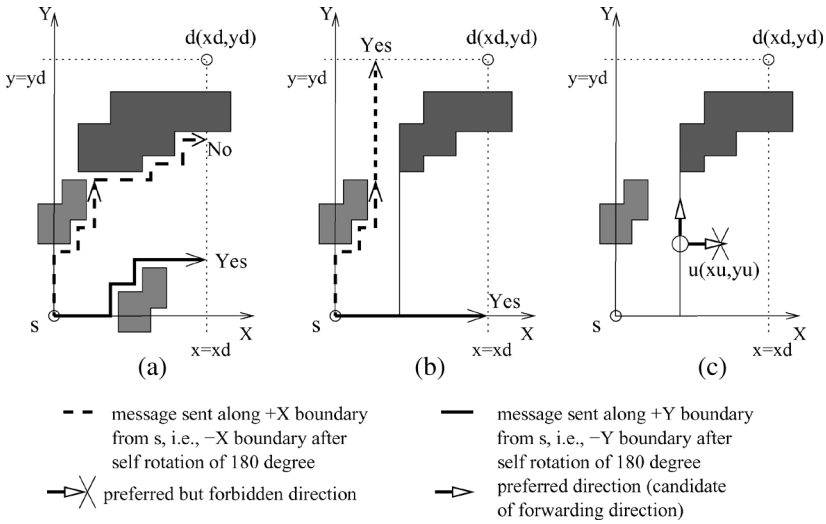

Fig. 6. (a) Check in a case without any Manhattan-distance-path; (b) check to ensure the existence of a Manhattan-distance-path; (c) routing-decision to construct a Manhattan-distance-path.

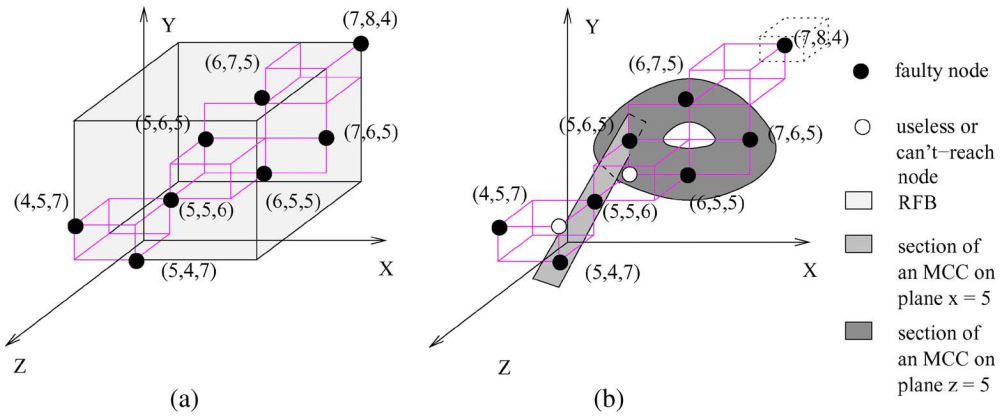

Fig. 7. (a) Sample of cuboid fault-block in 3-D meshes; (b) the corresponding MCC.

Fig. 7(b) shows two sample MCC 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 Fig. 7(b).

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

B. Edge-Identification

The identification of the shape of a MCC in 3-D meshes 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. The process is based on the identification of the shape for the MCC in 2-D meshes. For each section, for example an XY section, an identification 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 Fig. 8). Such a corner is also called the $(+Y - X)$-corner of this XY section. 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 $(+X - Z)$- & $(+Z - X)$-corners of a XZ section, and $(+Y - Z)$- & $(+Z - Y)$-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 messages in the identification of a 2-D section may meet at any edge-node, not necessarily a corner-node (see Fig. 8(b)).

After the identification of 2-D sections, 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,

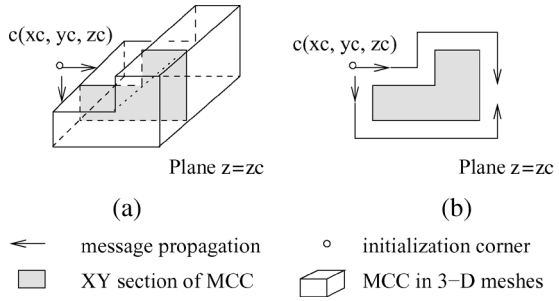

Fig. 8. Identification of an XY section with the initialization-corner, $(+Y-X)$-corner c.

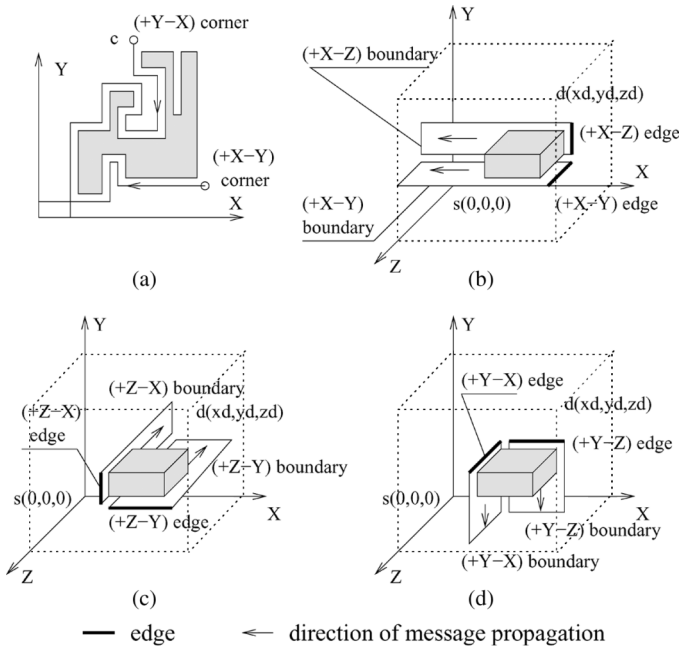

Fig. 9. Samples of corners, edges, and boundaries in 3-D meshes.

$(+Z-Y)$-edge, and $(+Z-X)$-edge (see Fig. 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 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 the succeeding edge-node. The other one is called its 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 routed around its 2-D section to find a path to the neighboring 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. When 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 a MCC is defined by the $(+Y-X)$-corners of all its XY sections. The identification of 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 Fig. 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 Fig. 10(b)). In phase three, when 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 Fig. 10(c)) is saved for future propagation of information. It is noted that the propagation of a message may require several hops to detour the irregular fault-region in each phase (see Fig. 11).

C. Edge-Construction

In phase one in the above identification 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 Fig. 11(a)). From that starting edge-node u, a collection-process is activated to collect all the links between preceding & succeeding edge-nodes, and form the entire edge. A message is sent along the path linking each edge-node to its preceding edge-node. Such a propagation will continue until it reaches the ending node, i.e., 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 critical-region $Q'(v)$ can also be formed at node v. Fig. 12 shows some samples of the determined information at edge-nodes. It is noted that an edge-node (see node c'' in Fig. 12(b)) may have more than one preceding edge-node. In that case, the collection process needs a multi-casting 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 the neighboring section.

D. 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-region $Q'_Y(u)$. Initially, a message carrying the information is sent from node u along the $-Y$ dimension (see Fig. 13(a)). When 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 $-Z$ & $-X$ directions to join its $(+Y-X)$-boundary & $(+Y-Z)$-boundary. 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 Fig. 13(b)). To avoid

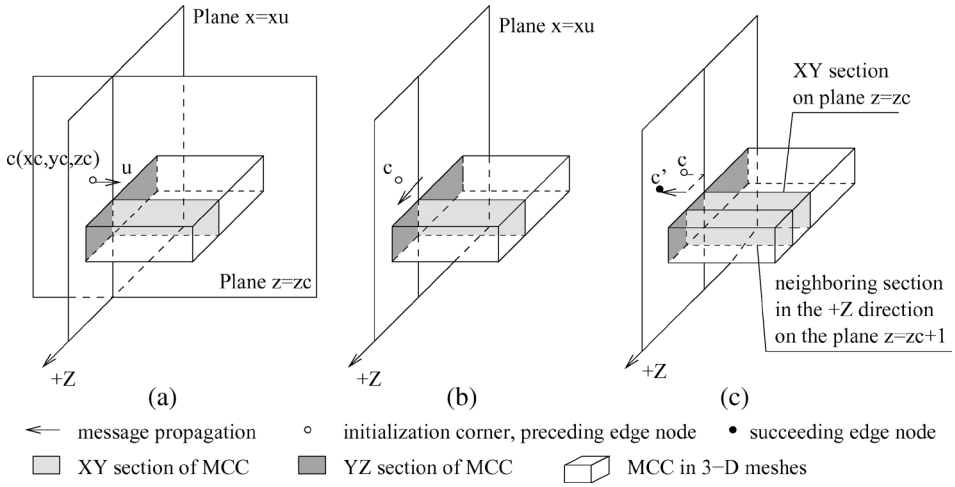

Fig. 10. Identification of edge from node $c(x_c, y_c, z_c)$: (a) phase one; (b) phase two; (c) phase three.

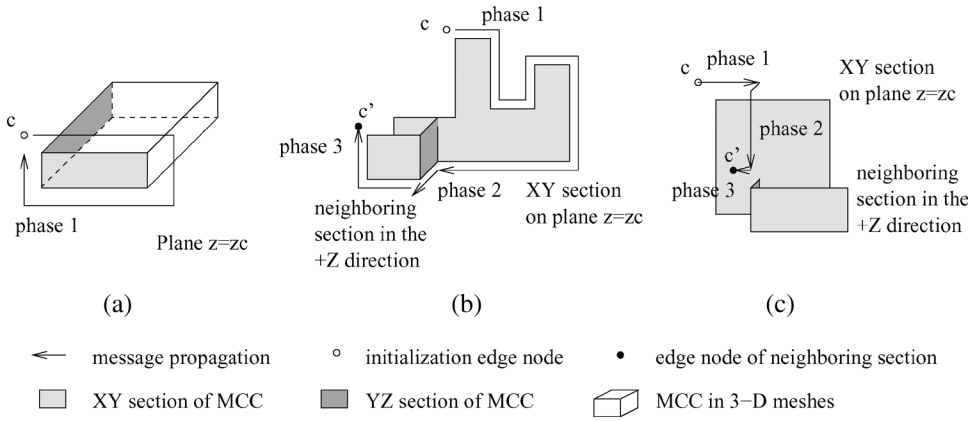

Fig. 11. Some samples of identification of edge-node c: (a) identification of starting point of edge; (b) complex case of phase one in finding neighboring section; (c) complex case of phase two in reaching the neighboring section.

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.

Fig. 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 Fig. 13(d). The whole procedure to collect & distribute MCC information is shown in Algorithm 5.

Algorithm 5: Identification & boundary-construction of a MCC in 3-D meshes.

---

1) Identification of each 2-D section by using the process of identification in Algorithm 2.
2) Edge-identification: 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) the message routes around that neighboring section to reach the corresponding corner, 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-node to its preceding edge-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 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 $M(u)$, $Q_Y(u)$, and $Q'_Y(u)$ 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 the corresponding forbidden-region Q_Y into $Q_Y(u)$.

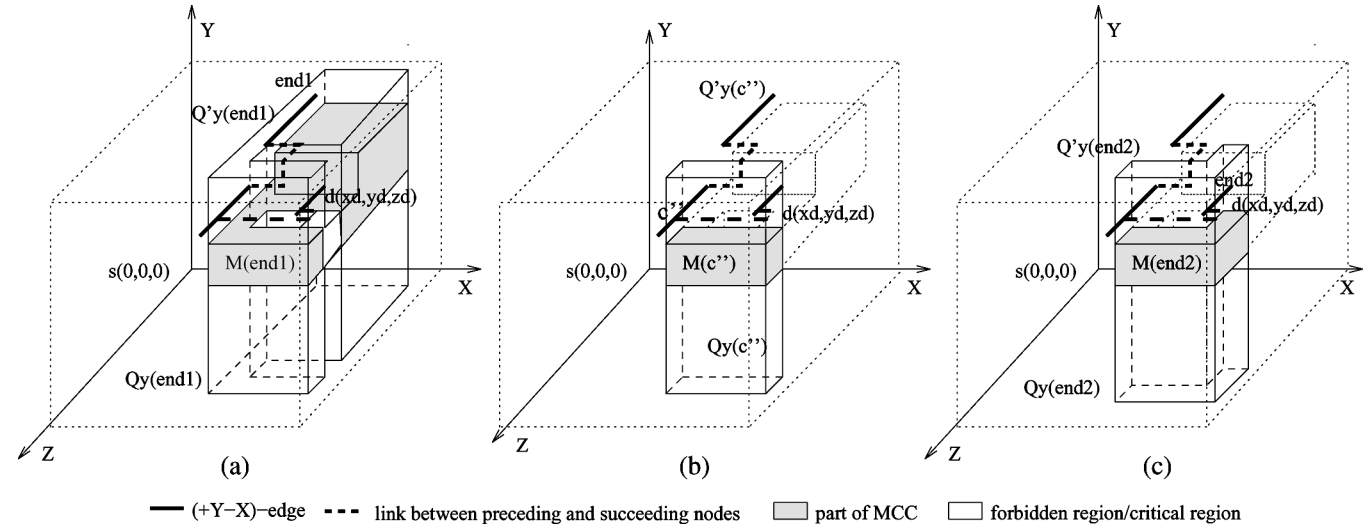

Fig. 12. Sample of constructed $(+Y-X)$-edge: (a) information (M, Q_y, and Q'_y) formed at ending edge-node *end1*; (b) information formed at an edge-node c''; (c) information formed at ending edge-node *end2*.

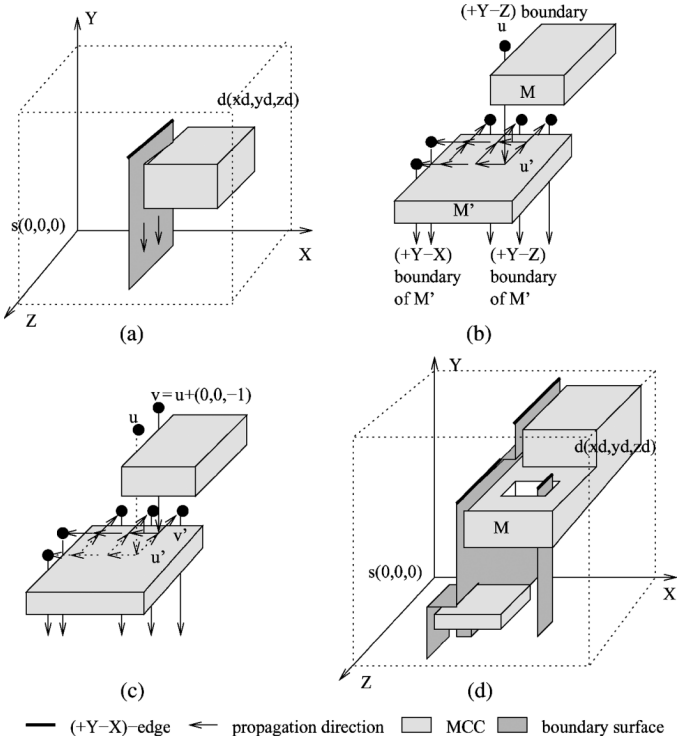

Fig. 13. (a) Construction of a $(+Y-X)$-boundary; (b) propagation of $M(u)$ at the intersection u' to cover the "nose"-part; (c) $M(u)$ overwritten by $M(v)$; (d) a complete $(+Y-X)$-boundary constructed for MCC M.

V. Sufficient & Necessary Condition for the Existence of a Manhattan-Distance-Path in 3-D Meshes

In this section, a sufficient & 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 information of fault-region $M(c)$, the forbidden-region $Q(c)$ ($Q_X(c)$, $Q_Y(c)$, or $Q_Z(c)$), and the critical-region $Q'(c)$ ($Q'_X(c)$, $Q'_Y(c)$, 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 a path in the Manhattan-routing to route around $M(c)$. Thus, we have the following sufficient & 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 iff there exists a MCC for which a) $d \in Q'_X \wedge s \in Q_X$; b) $d \in Q'_Y \wedge s \in Q_Y$; or c) $d \in Q'_Z \wedge s \in Q_Z$.

Theorem 2: A routing does not have a Manhattan-distance-path iff there exists a 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 $[0:x_d,0:y_d,0:0]$.

VI. 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 whenever this path exists.

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) Check of feasibility: At source s, send messages of detection along the $+X$, $+Y$, and $+Z$ directions. When a message, say the one along the $+X$ direction, intersects another MCC, it will join the $(-X+Y)$- & $(-X+Z)$-boundaries as boundary-construction. The source will check if these three 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.

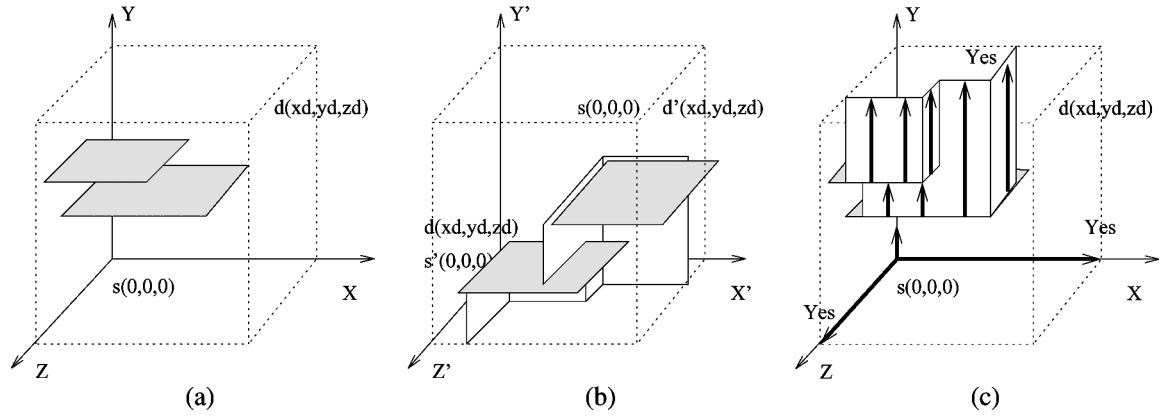

Fig. 14. Sample of propagation of detection-message: (a) configuration of faults; (b) configuration after a self-rotation of 180°, and the corresponding $(+Y-X)$- & $(+Y-Z)$-boundaries in the meshes; (c) propagation of detection message along the $(+Y-X)$- & $(+Y-Z)$-boundaries in (b).

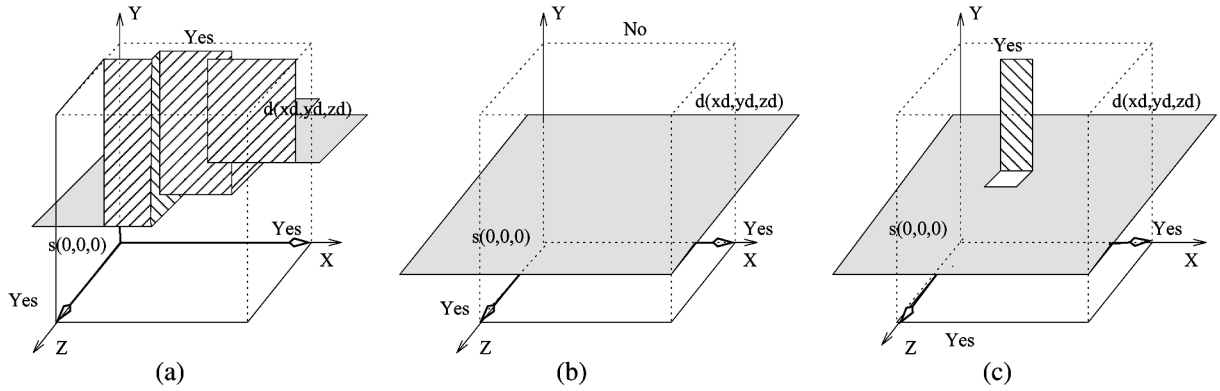

Fig. 15. Samples of check of feasibility.

2) Routing-decision 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 each recorded MCC.
   b) b) For each MCC found in the above step, remove 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 & minimal routing to select a forwarding-direction from set F.
   d) Forward the packet along the selected direction to the next node.

Similar to the routing in 2-D meshes, a 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 s will play the role of the destination in Theorem 2. Three messages of detection will be sent from s along the $+X$, $+Y$, and $+Z$ directions. If any message, say the one along the $+Y$ direction (see Fig. 14(c)), intersects another MCC, it will join the $(-Y+X)$- & $(-Y+Z)$-boundaries as boundary construction, i.e., the $(+Y-X)$- & $(+Y-Z)$-boundaries in the 3-D meshes after a self-rotation of 180° (see Fig. 14(b)). If any copy of 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 & necessary condition in Theorem 2, a Manhattan-routing from d to s exists; that is, a Manhattan-routing from s to d exists. Fig. 15 shows some samples of this check.

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

VII. Results of Monte-Carlo-Estimate

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 & updates needed in a synchronous-round-based system (i.e, the speed that the construction 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.

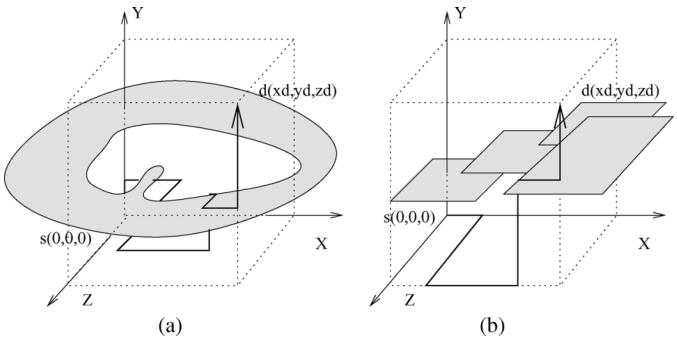

Fig. 16. Samples of routing under the MCC model.

The simulation is developed in the method of Monte-Carlo-estimate. That is, we simply take many 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 $\times$ 30 $\times$ 30 3-D mesh. In each case, a certain number of faults are uniformly distributed. After this, the MCC, and CFB are constructed. And then, many routing-cases with a pair of source & destination randomly generated for each are tested in different routings. We only test those cases when a Manhattan-distance-path exists between source & 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 configuration of faults. Therefore, only the results when the number of faults is no more than 500 are compared in a fair way in Fig. 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 Fig. 17(b), the construction of our MCC model also converges very fast in terms of the number of rounds needed. It is noted that, in our generator of random-faults, the faults will be distributed sparsely when only a small amount of faults are generated. When the number of faults is less than 10, MCC, and CFB 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 MCC and CFB when the number of faults is limited. However, the cost of the 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 check of feasibility. Wu presented a simple check of feasibility for the Manhattan-routing in [19], which only requires propagations of information along three rays. However, if the existence of a Manhattan-distance-path is ensured in the process of a check, then a Manhattan-routing can be conducted, but not the other way around. To reduce the cost for the check, we replace our check in Algorithm~6 by that in [19]. The corresponding

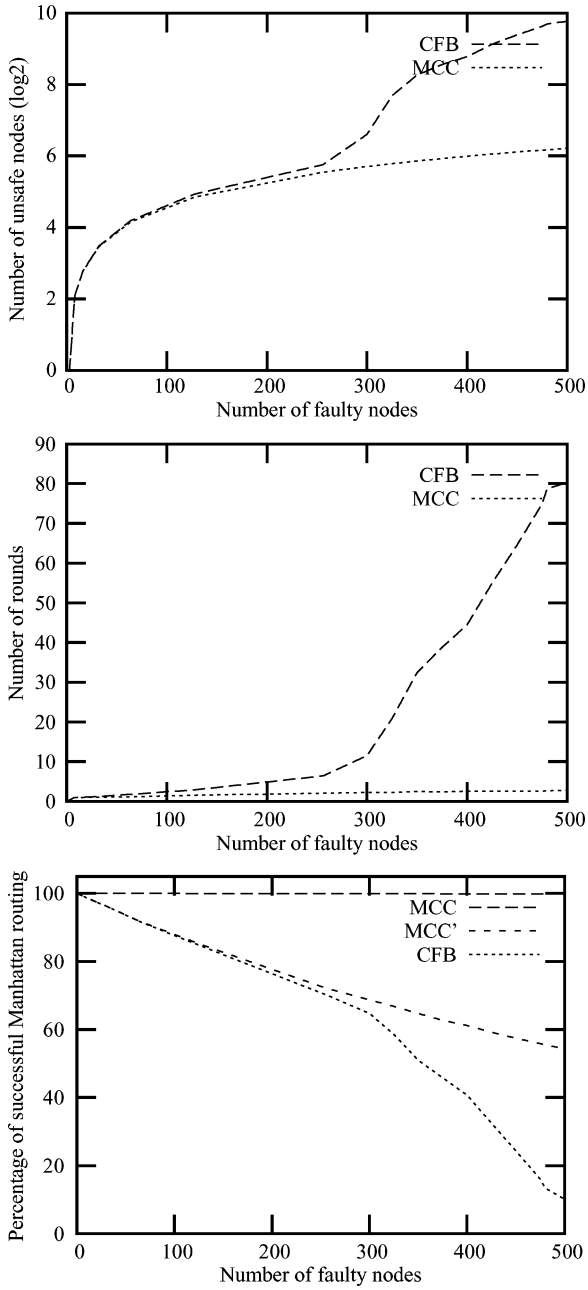

Fig. 17. (a) Number of unsafe nodes (lg); (b) number of rounds of information exchanges & updates in labeling process; (c) percentage of successful Manhattan-routing.

routing, denoted by MCC', is a simple version of our MCC routing. In the routing using a cuboid fault-block, which is denoted by CFB [19], the proposed Manhattan-routing under the MCC model which is denoted by MCC, and its simplified version MCC' routing, whenever the check of feasibility is passed, a Manhattan-routing can be guaranteed. Thus, in these routings, the success of Manhattan-routing depends on the passing rate of the check. Fig. 17(c) shows the percentage of passing cases, i.e., the rate of successful Manhattan-routing. The results of Monte-Carlo-simulation show that our new MCC routing can always find a Manhattan-distance-path whenever it exists, i.e., rate of successful $\mathrm{Manhattan{-}routing} = 100\%$. 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 value of applying our MCC information model in 3-D meshes.

VIII. 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 source & destination. Using a non-faulty node contained in a 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. We have provided a fully-distributed process in 2-D & 3-D meshes to collect & distribute the MCC information along the boundaries. Based on the boundary-information, our routing will guarantee a Manhattan-distance-path for the 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 & tested. We will also extend our results to dynamic networks in which any of the components can become faulty during the routing. 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.

Appendix

Here we give the proof of the theorems given in the main body of this paper.

Proof of Lemma 1: A sequence of MCC $(M_1, M_2, \ldots, M_n)$, 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 & M_{i+1}, $1 \leq i \leq n-1$,

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

and

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

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

Similarly, a sequence of MCC $(M_1, M_2, \ldots, M_n)$, 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 & M_{i+1}, $1 \leq i \leq n-1$,

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

and

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

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

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

Proof of Lemma 2: 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) & 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 $Q_Z(c_z)$. ■

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

Acknowledgment

The authors would like to thank the anonymous reviewers for their many suggestions that helped improving the presentation quality of the article.

References

[1] Manhattan-Distance (Definition) [Online]. Available: http://www.nist.gov/dads/HTML/manhattanDistance.html.
[2] Monte Carlo Method (Definition) [Online]. Available: 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*, vol. 40, no. 2, pp. 310–327, 2001.
[4] R. V. Boppana and S. Chalasani, "Fault tolerant wormhole routing algorithms for mesh networks," *IEEE Trans. Computers*, vol. 44, no. 7, pp. 848–864, July 1995.
[5] Y. M. Boura and C. R. Das, "Fault-tolerant routing in mesh networks," in *Proc. of 1995 International Conference on Parallel Processing*, August 1995, vol. 1, pp. 106–109.
[6] S. Chalasani and R. V. Boppana, "Communication in multicomputers with nonconvex faults," *IEEE Trans. Computers*, vol. 46, no. 5, pp. 616–622, May 1997.
[7] A. A. Chien and J. H. Kim, "Planar-adaptive routing: Low-cost adaptive networks for multiprocessors," *Journal of ACM*, vol. 42, no. 1, pp. 91–123, January 1995.

[8] W. J. Dally, "The J-machine: System support for actors," in *Towards Open Information Science*, Hewitt, Carl, and Agha, Eds. : MIT Press, 1992.
[9] R. L. Hadas and E. Brandt, "Origin-based fault-tolerant routing in the mesh," *Future Generation Computer Systems*, vol. 11, no. 6, pp. 603–615, October 1995.
[10] Z. Jiang and J. Wu, "A limited-global-information model for dynamic routing in 2-D meshes," in *Proc. of 3rd 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, pp. 8–21, Spring, 1994.
[12] R. Libeskind-Hadas and E. Brandt, "Origin-based fault-tolerant routing in the mesh," in *Proc. of the 1st International Symposium on High Performance Computer Architecture*, January 1995, pp. 102–111.
[13] L. Sheng and J. Wu, "Maximum-shortest-path (msp) is not optimal for a general n × n torus," *IEEE Trans. Reliability*, vol. 52, no. 1, pp. 22–25, 2003.
[14] C. C. Su and K. G. Shin, "Adaptive fault-tolerant deadlock-free routing in meshes and hypercubes," *IEEE Trans. Computers*, vol. 45, no. 6, pp. 672–683, June 1996.
[15] D. Wang, "A rectilinear-monotone polygonal fault block model for fault-tolerant minimal routing in meshes," *IEEE Trans. Computers*, vol. 52, no. 3, pp. 310–320, March 2003.
[16] J. Wu, "Fault-tolerant adaptive and minimal routing in mesh-connected multicomputers using extended safety levels," *IEEE Trans. Parallel and Distributed Systems*, vol. 11, no. 2, pp. 149–159, February 2000.
[17] J. Wu, "A fault-tolerant adaptive and minimal routing scheme in n-D meshes," *The Computer Journal*, vol. 45, no. 3, pp. 349–363, 2002.
[18] J. Wu, "Maximum-shortest-path (msp): An optimal routing policy for mesh-connected multicomputers," *IEEE Trans. Reliability*, vol. 48, no. 3, pp. 247–255, 1999.
[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, pp. 1–13, 2003.
[20] J. Wu and Z. Jiang, "On constructing the minimum orthogonal convex polygon in 2-D faulty meshes," in *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 Trans. Parallel and Distributed Systems*, vol. 12, no. 9, pp. 942–951, 2001.
[22] D. Xiang, A. Chen, and J. Wu, "Reliable broadcasting in wormhole-routed hypercube-connected networks using local safety information," *IEEE Trans. Reliability*, vol. 52, no. 2, pp. 245–256, 2003.

Zhen Jiang received the BS degree in Computer Engineering from Shanghai Jiaotong University, P. R. China, in 1992; the Master degree in Computer Science from Nanjing University, P.R. China, in 1998; and the Ph.D. degree in Computer Engineering from Florida Atlantic University in 2002. He is currently an associate professor in the department of Computer Science at West Chester University of Pennsylvania, and a faculty member of the Information Security Center at West Chester University. His research interests are in the area of computer and network security, fault tolerant systems, and wireless sensor networks. He is a member of the IEEE.

Jie Wu is a distinguished professor in the Department of Computer Science and Engineering at Florida Atlantic University, and a program director at the US National Science Foundation. He has published more than 300 papers in various journals and conference proceedings. His research interests include mobile computing, routing protocols, fault-tolerant computing, and interconnection networks. He was a co-guest editor of a special issue in Computer on ad hoc networks. He was also an editor of several special issues of the Journal of Parallel and Distributing Computing (JPDC), and IEEE TRANSACTIONS ON PARALLEL AND DISTRIBUTED SYSTEMS (TPDS). He is the author of the text Distributed System Design (CRC, 1998), and is the editor of the text Handbook on Theoretical and Algorithmic Aspects of Sensor, Ad Hoc Wireless, and Peer-to-Peer Networks (Auerbach, 2005). He served as an associate editor of the IEEE TRANSACTIONS ON PARALLEL AND DISTRIBUTED SYSTEMS, and is currently on the editorial board of several international journals. He is a recipient of the 1996-1997, 2001-2002, and 2006-2007 Researcher of the Year Awards at Florida Atlantic University. He served as an IEEE Computer Society distinguished visitor, and is the chairman of IEEE Technical Committee on Distributed Processing (TCDP). He is a senior member of the IEEE.

Dajin Wang received the BEng degree from Shanhai University of Science and Technology, China, in 1982; and the MS degree, and the PhD degree, both in computer science, from the Stevens Institute of Technology, Hoboken, New Jersey, in 1986, and 1990, respectively. Since then, he has been with the faculty of the Department of Computer Science at Montclair University, Upper Montclair, New Jersey, where he is currently a professor. He has also been a visiting research professor at the State Key Laboratory for Novel Software Technology at Nanjing University, China, and has held the position of visiting scholar at Hong Kong Polytechnic University. His research interests include fault-tolerant computing, algorithmic robotics, and parallel processing. He is a member of the IEEE.