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

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

Jie Wu
Dept. of Computer Sci. & Eng.
Florida Atlantic University
Boca Raton, FL 33431
jie@cse.fau.edu

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

Abstract

In this paper we rewrite Wang's Minimal-Connected-Component (MCC) model [7] in 2-D meshes without using global information so that not only the existence of a minimal path can be ensured at the source, but also such a path can be formed by routing decisions at intermediate nodes along the path. We extend this MCC model and the corresponding routing in 2-D meshes to 3-D meshes. It is based on our early work on fault tolerant adaptive and minimal routing [9] and the boundary information model [8] in 3-D meshes. We study fault tolerant adaptive and minimal routing from the source and the destination and 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 sufficient and necessary condition is proposed for the existence of the minimal routing path in the presence of our faulty components. Based on such a condition, the corresponding routing will guarantee a minimal path whenever it exists.

Index Terms: Adaptive routing, fault information models, fault tolerance, minimal routing, 3-D meshes.

1 Introduction

The *mesh-connected topology* 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 [1, 5]. The performance of such a multicomputer system is highly dependent on the node-to-node communication cost. It is necessary to present a *minimal routing* (i.e., a shortest path routing) in mesh networks. We focus here on achieving fault tolerance using the inherent redundancy present in 3-D meshes, without adding spare nodes and/or links.

Most existing literature [2, 3, 6, 10] uses the simplest orthogonal convex region to model node faults (link faults can be treated as node faults by disabling the corresponding adjacent nodes). Wu provided a node labelling scheme in [8] that identifies nodes (faulty and non-faulty) that cause routing detours in 2-D meshes and such nodes are called unsafe nodes. Connected unsafe nodes form a rectangular region, also called a rectangular faulty block. In [8], the information of rectangular faulty blocks is distributed to a limited number of nodes at the boundary lines to prevent a routing message from entering a detour area. By using this so-called limited global fault information at boundary lines in 2-D meshes, Wu's fully adaptive routing proposed in [8] can easily find a minimal path. To reduce the number of non-faulty nodes contained in rectangular faulty blocks, Wang [7] proposed the minimal connected component (MCC) model as a refinement of the rectangular faulty block model by considering the relative locations of source and destination nodes. The original idea is that a node will be included in an MCC only if using it in a routing will definitely make the route non-minimal. It turns out that each MCC is of the rectilinear monotone polygonal shape and is the absolutely minimal fault region in 2-D meshes.

In this paper, we provide a boundary construction for MCCs in 2-D meshes through message exchanges among neighboring nodes. With the information of MCCs at the boundary lines, Wang's sufficient and necessary condition

*The work was supported in part by NSF grants ANI 0083836, CCR 9900646, CNS 0422762, CNS 0434533, and EIA 0130806.

Proceedings of the 2005 International Conference on Parallel Processing (ICPP'05)
0190-3918/05 $20.00 © 2005 **IEEE**

of the existence of the minimal routing can be re-written so that not only a minimal routing can be ensured at the source node but also a minimal path can be formed by routing decisions at intermediate nodes along the path. After that, the MCC model and its boundary construction will be extended to 3-D meshes. A sufficient and necessary condition is proposed for the existence of the minimal routing in 3-D meshes. Based on this condition, a new (fully) adaptive routing is provided to build a minimal path between source and destination nodes whenever such a minimal path exists. Extensive simulation has been done to determine the number of non-faulty nodes included in MCCs in 3-D meshes and the rate of success minimal routing under MCC model. The result obtained is compared with the best existing known result. Due to the space limit, these simulation results are not shown. Details can be found in [4].

The challenge here is not only to conduct a theoretical study on the MCC model in 3-D meshes and its corresponding sufficient and necessary condition for existence of minimal routing, but also to search for a practical and efficient implementation in a system where each node knows only the status of its neighbors. First, after each non-faulty node in an MCC is labelled, the whole 3-D fault region should be identified. Then, the identified information of this MCC will be propagated in two dimensions along some 2-D surfaces (also called boundaries) to avoid the routing entering the detour area. Finally, a new routing with two phases is proposed. In phase one, the boundary information of any MCC that may block the routing message is collected and used to build the assurance of minimal routing at the source node. The routing process at the source will be activated only if a minimal path exists. In phase two, the routing process at each intermediate node between source and destination nodes will forward the message to the next node along the path. It uses the boundary information to avoid sending the message to the detour area and to keep the routing path minimal. It is noted that all these are implemented through the message transmission (including information messages and routing messages) between two neighboring nodes along one of those three dimensions X, Y and Z. Throughout the paper, proofs to theorems are omitted. Details can be found in [4].

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 labelled as (x_u, y_u) and

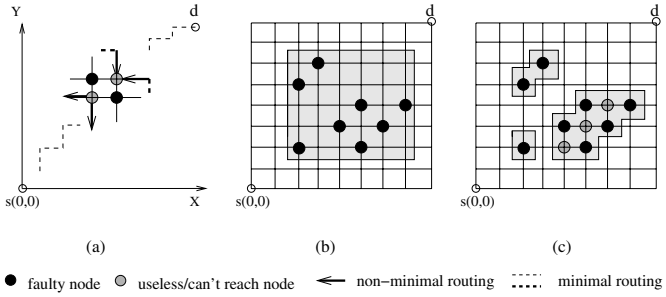

Figure 1. (a) Definition of useless and can't-reach nodes. (b) Sample of rectangular faulty block. (c) The corresponding MCCs.

the distance between two nodes u and v, $D(u, v)$, is equal to $\mid x_v - x_u \mid + \mid y_v - y_u \mid$. 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 $D(v, d) < D(u, d)$ where d is the destination; otherwise, it is called a *spare neighbor*. Respectively, the corresponding connecting directions are called *preferred direction* and *spare direction*. Without loss of generality, assume node $s(0, 0)$ is source node and node $d(x_d, y_d)$ $(x_d, y_d \geq 0)$ is the destination node. A routing process is *minimal* if the length of the routing path from source node s to destination node d is equal to $D(s, d)$. 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 distance between two nodes u and v, $D(u, v)$, is equal to $\mid x_v - x_u \mid + \mid y_v - y_u \mid + \mid z_v - z_u \mid$. $(x_u + 1, y_u, z_u)$, $(x_u - 1, y_u, z_u)$, $(x_u, y_u + 1, z_u)$, $(x_u, y_u - 1, z_u)$, $(x_u, y_u, z_u + 1)$ and $(x_u, y_u, z_u - 1)$ are $+X$, $-X$, $+Y$, $-Y$, $+Z$, and $-Z$ neighbors of node u.

The formation of MCC in 2-D meshes [7] is based on the notions of *useless* and *can't-reach* nodes (see in Figure 1 (a)): A node labelled useless is such a node that once a routing from $(0, 0)$ to (x_d, y_d) $(x_d, y_d \geq 0)$ enters it, the next move must take either $-X$ or $-Y$ direction, making the routing non-minimal. A node labelled can't-reach is such a node that for a routing to enter it, a $-X$ or $-Y$ direction move must be taken, making the routing non-minimal. The node status (non-faulty, faulty, useless, and can't-reach) can be determined through a labelling procedure. All faulty, useless, and can't-reach nodes are also called unsafe nodes. The labelling procedure is given in Algorithm 1. Only the affected nodes update their status. Figure 1 (a) shows the idea of the definition of useless and can't-reach nodes. Figure 1 (c) shows some samples of MCCs for the routing from

Proceedings of the 2005 International Conference on Parallel Processing (ICPP'05)
0190-3918/05 $20.00 © 2005 IEEE

Algorithm 1: Labelling 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 labelled *useless*.
3. If node u is safe, but its $-X$ neighbor and $-Y$ neighbor are faulty or can't-reach, u is labelled *can't-reach*.
4. The nodes are recursively labelled until there is no new useless or can't-reach node. All faulty, useless, and can't-reach nodes (other than safe nodes) are also called *unsafe* nodes.

$(0,0)$ to (x_d, y_d) $(x_d, y_d \geq 0)$.

3 Boundary information in 2-D meshes

In this section, we provide a distributed process to collect the information of each MCC and distribute it along the boundaries. Based on such information, the new routing process provided in this section can form a minimal path from the source to destination nodes whenever this path exists.

Corner and boundary of MCC . To collect the information of all MCCs for routing process, each MCC needs to identify its fault region. Any node inside fault region 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 nodes of the same MCC in different dimensions or a safe node with two unsafe node of the same MCC in different dimensions. After the labelling procedure, the identification process starts from an *initialization corner*. The initialization corner is a corner with two edge nodes of the same MCC in the $+X$ and $+Y$ dimensions. A safe node with two edge nodes of the same MCC in the $-X$ and $-Y$ dimensions is called its *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 these two edge neighbors. Such propagation will continue along the edges until the messages reach the opposite corner. When the clockwise message passes through any intermediate corner u, node information (x_u, y_u) will be attached to the message and 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 shape 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 2 shows a sample of the identification process.

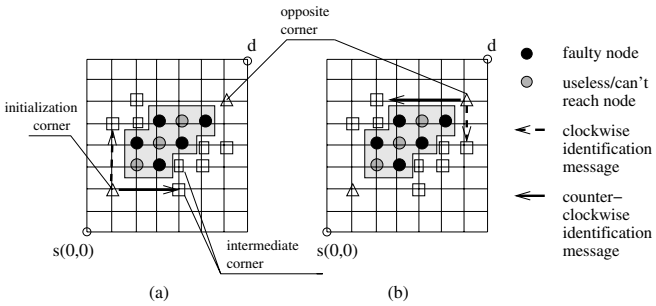

Figure 2. (a) Identification process activated at the initialization corner. (b) Identified information re-sending.

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 change of shape 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 the initialization corner c is noted by $M(c)$. A routing message should be forbidden to enter the region right below it if the destination is right above it. The first region is also called the *forbidden region*, noted by $Q_Y(c)$. The corresponding region right above $M(c)$ is called *critical region*, noted by $Q'_Y(c)$. Similarly, the routing message should avoid entering the forbidden region $Q_X(c)$ on the left side of $M(c)$ 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 Y boundary) will carry the information $M(c)$, $Q_Y(c)$, and $Q'_Y(c)$ and propagate to all the nodes along the boundary line $x = x_c$ in the $-Y$ direction 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 in Figure 3 (a)). Similarly, another boundary propagation (construction of

Proceedings of the 2005 International Conference on Parallel Processing (ICPP'05)
0190-3918/05 $20.00 © 2005 **IEEE**

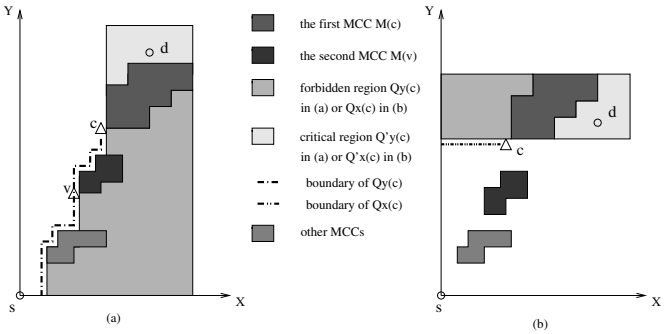

Figure 3. Samples of boundary construction under MCC model in 2-D meshes.

X boundary) carrying $M(c)$, $Q_X(c)$, and $Q'_X(c)$ will go along $y = y_c$ in the $-X$ direction and make a turn in the $-Y$ direction if necessary (see in Figure 3 (b)). The whole procedure is shown in Algorithm 2.

Algorithm 2: Identification process and boundary construction

1. Identification of edge nodes, the initialization corner c, intermediate corners, and the opposite corner c'.
2. Identification process of MCC $M(c)$: (a) From node c, two identification message (one clockwise and one counterclockwise) are sent along the edge nodes of $M(c)$ until they reach node c'. (b) Node information of all corners 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 shape information reaches back to node c. (d) The stable shape of $M(c)$ can be ensured at node c once two identification messages are both received. Meanwhile, the critical and forbidden 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 $M(c)$, $Q_X(c)$ / $Q_Y(c)$, and $Q'_X(c)$ / $Q'_Y(c)$ is propagated along the boundary line $y = y_c$ / $x = x_c$. When the propagation intersects another MCC ($M(v)$), it will go along the edges of $M(v)$ and join the same boundary of $M(v)$. Since then, the area information of forbidden region of $M(v)$ ($Q_X(v)$ / $Q_Y(v)$) will merge into that of $M(c)$: $Q_X(c)$ / $Q_Y(c)$.

Sufficient and necessary condition for the existence of minimal routing. 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 minimal routing. A routing that enters a non-faulty node in the MCC would force a step that violates the requirement for a minimal routing. If there exists no minimal routing under the MCC model, there will be absolutely no minimal routing. In [7], a sufficient and necessary condition was provided for the existence of minimal routing. This can be rewritten as the following:

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

Boundary-information-based routing. Wu proposed a minimal and adaptive routing in 3-D meshes in [9]. It can easily be extended to a routing in 2-D meshes under the MCC model (see in Algorithm 3). In this routing, at the source node s, a detection is activated to check if there exists a minimal path. First, node s will play the role as the destination in Theorem 1. It sends two detection messages, one along $+Y$ direction and one along $+X$ direction. If the first one cannot reach the segment $[0 : x_d, y_d : y_d]$, a minimal path is impossible (return "No"). If it intersects with another MCC, make a turn to the $+X$ direction, and then, turn back to the $+Y$ direction as soon as possible. Similarly, the second one is to check if the segment $[x_d : x_d, 0 : y_d]$ can be reached. If the source s knows both segments can be reached, based on Theorem 1, a minimal path exists from d to s (i.e., a minimal routing is feasible from s to d).

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 $+Y$ direction and the second along $+X$ direction) until they reach the line $x = x_d$ or line $y = y_d$. If the first / second message intersects with another MCC, make a turn to $+X$ / $+Y$ direction, and then, turn back to the $+Y$ / $+X$ direction as soon as possible. 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 minimal path.
2. Routing decision and message sending at the current node u, including node s, (a) add all the preferred directions into the set of candidates of forwarding directions F and find all the records of MCCs, (b) for each MCC found, exclude a direction from F if the destination is in the critical region and the neighbor of u along this direction is inside forbidden region, (c) apply any fully adaptive and minimal routing process to pick up a forwarding direction from set F, and (d) forward the routing message along the selected forwarding direction to the next node.

Proceedings of the 2005 International Conference on Parallel Processing (ICPP'05)
0190-3918/05 $20.00 © 2005 **IEEE**

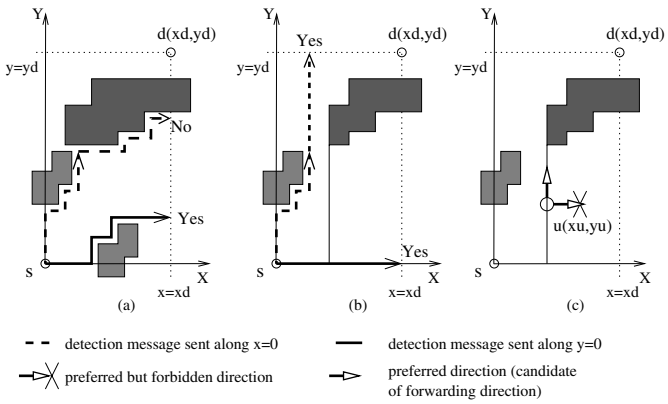

Figure 4. (a) Feasibility check for a case without minimal routing path. (b) Feasibility check to ensure the existence of minimal routing path. (c) Routing decisions in routing process to construct a minimal routing path.

After the detection, at each node along the routing path, including the source node s, the routing process basically has two preferred directions: $+X$ and $+Y$ 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 forwarding direction. 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. The procedure of feasibility check and routing decision can also be seen in the samples in Figure 4.

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 is identified in a labelling process. Then, each 2-D section of a 3-D fault region and its neighboring sections are identified in an identification process. After that, the information of 2-D sections of a fault region are collected along the edges of this fault region 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 are formed. Finally, in boundary construction, this information will be propagated along the boundary of this MCC to avoid the routing entering its forbidden region.

Labelling process. For a non-faulty node u in 3-D meshes, if it has only two useless or faulty neighbors in $+X$ and $+Y$ directions, the routing message can take the $+Z$ direction and route around the fault region. Therefore, a non-faulty node is useless in 3-D meshes if and only if it has three useless or faulty neighbors in $+X$, $+Y$, 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 $-X$, $-Y$, and $-Z$ directions. The corresponding labelling scheme is shown in Algorithm 4.

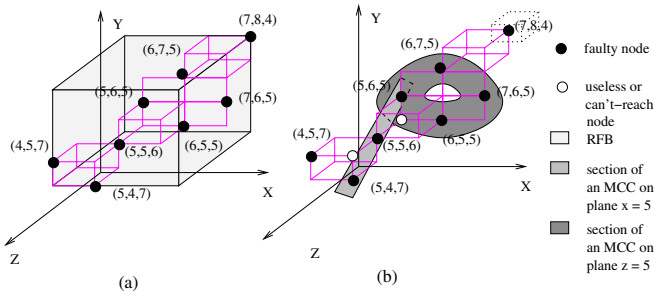

Figure 5. (a) Sample rectangular faulty block in 3-D meshes. (b) The corresponding MCCs.

Algorithm 4: Labelling 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 $+X$, $+Y$, and $+Z$ neighbors are faulty or useless, u is labelled *useless*.
3. If node u is safe, but its $-X$, $-Y$, and $-Z$ neighbors are faulty or can't-reach, u is labelled *can't-reach*.
4. The nodes are recursively labelled until there is no new useless or can't-reach node. All faulty, useless, and can't-reach nodes are also called *unsafe* nodes.

Figure 5 (b) shows two sample MCCs in 3-D meshes. One MCC contains only one faulty node $(7, 8, 4)$ and the other MCC contains all the other unsafe nodes, including the faulty nodes: $(5, 5, 6)$, $(6, 5, 5)$, $(5, 6, 5)$, $(6, 7, 5)$, $(7, 6, 5)$, $(5, 4, 7)$, and $(4, 5, 7)$. $(5, 5, 5)$ becomes useless and $(5, 5, 7)$ becomes can't-reach in our labelling process. 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 as a polygon P for which a line segment connecting any two points in P lies entirely within P. A section of the second MCC on the plane $z = 5$ shows a hole at $(6, 6, 5)$ in the MCC region.

Identification process. The identification process for an MCC in 3-D meshes is based on the one 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 2-D section, for example, a XY

Proceedings of the 2005 International Conference on Parallel Processing (ICPP'05)
0190-3918/05 $20.00 © 2005 **IEEE**

section, a two-head-on message identification process in algorithm 2 is activated at a corner c. This XY section may have several corners. The one with the minimum coordinate along the X dimension of those which have the maximum coordinate along Y dimension is called $(+Y-X)$-corner of this section. Respectively, we have $(+X-Y)$-corner of the same XY section, $(+X-Z)$- and $(+Z-X)$-corners of a XZ section, and $(+Y-Z)$- and $(+Z-Y)$-corners of a YZ section. Each XY / YZ / XZ section will be identified by a similar process initialize from its $(+Y-X)$- / $(+Z-Y)$- / $(+X-Z)$-corner. It is noted that these two identification messages may meet at any edge node of the 2-D section, not necessary a corner node.

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. Any of these edges is defined by all of its edge nodes and each edge node is the corresponding corner in its 2-D section. The identification process starts from each of these corners and has three phases. It is to find a path to link each pair of the preceding node and its succeeding node along the edge so that the whole edge can be formed. In phase one, a message will be initiated at the start corner and route around its 2-D section to find a path to the neighboring section. 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 its corresponding start corner. Once this message reaches that corner, those same type 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 XY sections of this MCC. In phase one, from a $(+Y-X)$-corner $c(x_c, y_c, z_c)$ in the XY section $z = z_c$ in Figure 6 (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 $-Y$ dimension, the identified information of the YZ section on the plane $x = x_u$ is used to find a neighboring section on plane $z = z_c + 1$. In phase two, the message will go around the corresponding YZ section to the neighboring XY section (seen in Figure 6 (a)). In phase three, once the message arrives at a node of the neighboring XY section, a two-head-on message propagation will be initiated to go around that section (one clockwise and one counter-clockwise) to reach its corresponding $(+Y-X)$-corner u' (seen in Figure 6 (b)). At node u', c is identified as its succeeding node along the edge and the information of the path to node c (see in Figure 6 (c)) is saved for future information propagation.

Edge construction. If the neighboring section cannot be found in phase one in the above identification process, that start corner is identified as one end of this edge and the corresponding section is identified as the surface of this MCC (see corner c' in Figure 6 (a)). In phase three of the above identification process, a concave region of the MCC containing the start corner can be identified if there is another 2-D section in the same plane. For example, in Figure 6 (b), at node u'', the second section is found. Once such a concave region is found, the start corner (corner c'' in the example in Figure 6 (b)) will be identified as one end of another edge (an inner edge of this MCC towards a concave area).

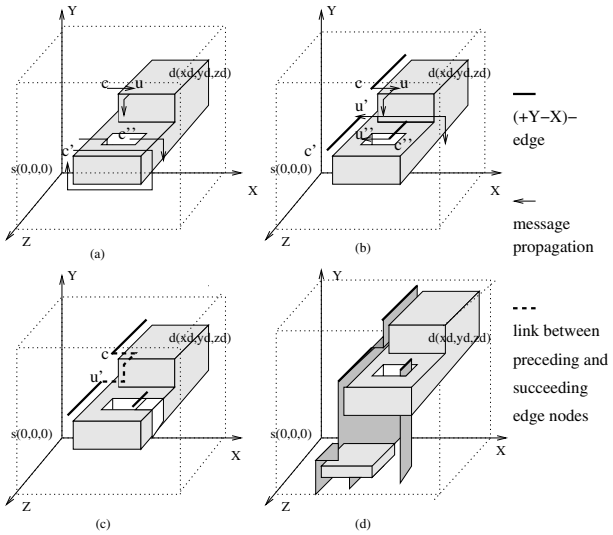

Figure 6. (a) Section identification. (b) Edge Identification. (c) Edge construction. (d) Boundary construction.

Starting from an identified end node u , the entire edge will form by collecting all the links between preceding and succeeding nodes along this edge. From u, a message is sent along the path to its succeeding node and such propagation will continue until it reaches the other end (an edge node without any succeeding node). At each edge node v it passes through, the section information is collected and the previously saved information is used to form a part of this MCC, $M(v)$. This $M(v)$ includes the MCC area from the current node v to the end node u. With the information of $M(v)$, the information of forbidden region $Q(v)$ and the critical region $Q'(v)$ can also be formed at node v (see in the example in Figure 6 (c)).

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 detour area $Q_Y(u)$ in the $+X$ dimension if the destination is inside the critical region $Q'_Y(u)$. Initially, a message carrying the information is

Proceedings of the 2005 International Conference on Parallel Processing (ICPP'05)
0190-3918/05 $20.00 © 2005 **IEEE**

sent from node u along the $-Y$ dimension. Once this message intersects with another MCC at node u', the boundary of node u will make a $-X$ turn and route around the XY section of the latter MCC until it joins the boundary of the latter MCC from v. Meanwhile, the forbidden region of node v ($Q_Y(v)$) will be merged into $Q_Y(u)$ ($Q_Y(u) = Q_Y(v) \cup Q_Y(u)$). If node u is the $(-Z)$-most edge node (the edge node without any succeeding node), at node u', a copy of the message will be sent to node v. From node v, it will go along that $(+Y - X)$-edge and reach the $(-Z)$-most edge node of the latter MCC. At each edge node v' it passes through, a boundary is constructed for node u with the information of $M(u)$, $Q'_Y(u)$, and $Q_Y(u) \cup Q_Y(v')$. A sample of boundary construction is shown in Figure 6 (d). The whole procedure is shown in Algorithm 5.

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

1. Identifi cation of each 2-D section (see in Algorithm 2).
2. Identifi cation of each edge: (a) a message is sent along each XY, YZ, or XZ section from its start corner to fi nd the neighboring section; (b) the message reaches the neighboring section in one hop; (c) an identifi cation process in algorithm 2 is applied to reach the corresponding corner in this neighboring section, the preceding node of that start corner.
3. Edge construction: If no neighboring section is found or there is another section in the same plane, the start corner will be identifi ed as an end node without a preceding node. From this end node, a message is sent to the succeeding node along the edge and the propagation will continue until it reaches the other end edge node. It will collect the section information at each edge node c and the previously saved information will be used to form the concerning MCC part $M(c)$, the forbidden region $Q(c)$, and the critical region $Q'(c)$.
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 will be propagated along its $(+Y - X)$-boundary in the $-Y$ direction. If it intersects with another MCC, $M(v)$, it will join the boundary of the "hose" part of $M(v)$ and merge $Q_Y(v)$ into $Q_Y(u)$.

5 Suffi cient and necessary condition for the existence of minimal routing 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 enters this node and its destination d is inside the critical region, the boundary line can be used as a part of the routing path to route around the $M(c)$ and

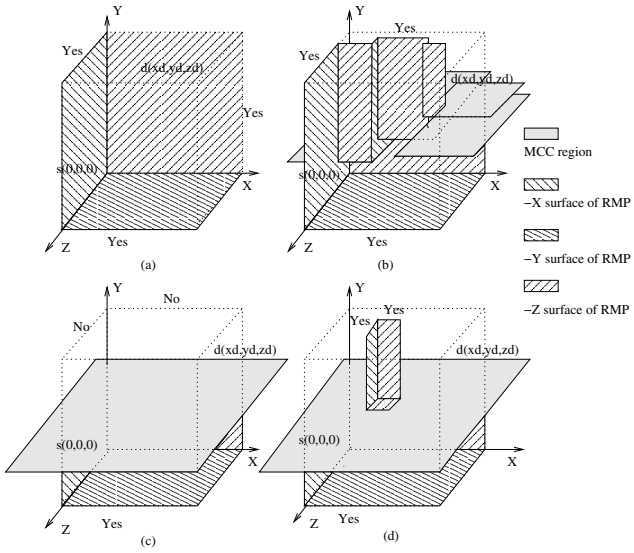

Figure 7. Samples of feasibility check.

avoid detours. We have the following sufficient and necessary condition for the existence of minimal routing in 3-D meshes:

Theorem 2: *A routing does not have a minimal 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 [$0 : x_d$, $0 : y_d$, $0 : 0$].*

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

Based on Theorem 2, Wu's routing in [9] in 3-D meshes is extended to a routing under the MCC model (see in Algorithm 6). Such a routing can find a minimal path from the source and destination nodes whenever this path exists.

Algorithm 6: Routing process.

1. Feasibility check at s: Send detection messages along three surfaces of RMP, which includes all the intermediate nodes along the shortest path from s to d: $(-X)$-surface on the $-X$ side, $(-Y)$-surface on the $-Y$ side, and $(-Z)$-surface on the $-Z$ side. Any message on $(-X)$-surface reaches the surface [$0 : x_d$, $y_d : y_d$, $0 : z_d$], it will return the result back to s. Similarly, the propagation on the $(-Y)$- or $(-Z)$-surface is to see if the surface [$0 : x_d$, $0 : y_d$, $z_d : z_d$] or [$x_d : x_d$, $0 : y_d$, $0 : z_d$] can be reached. If any surface

Proceedings of the 2005 International Conference on Parallel Processing (ICPP'05)
0190-3918/05 $20.00 © 2005 **IEEE**

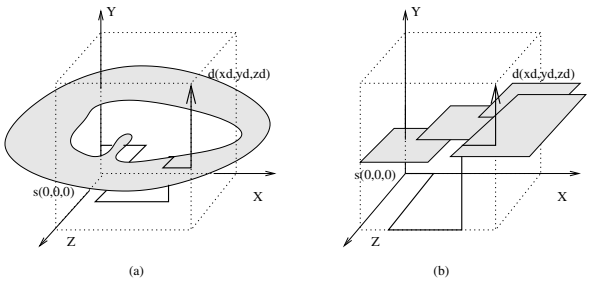

Figure 8. Samples of routing.

cannot be reached, stop the routing since there is no minimal path.

2. Routing decision and message sending at the current node u, including node s: same as step 2 in Algorithm 3.

Similar to the routing in 2-D meshes, at the source node s a detection is first activated to make sure if there exists a minimal path. The source node s will play the role of the destination in Theorem 2 and three detection messages will be sent from s along each surface of the region of minimal path (RMP): $(-Y)$-surface on the $-Y$ side, $(-Z)$-surface on the $-Z$ side, and $(-X)$-surface on the $-X$ side. The RMP includes each node along the shortest path from s to d. For each surface, say the $(-X)$-surface, a message will first propagate to the neighbors of s in the $+Y$ and $+Z$ directions. After that, at each node u that receives it, this message will continue to propagate to u's $+Y$ neighbor and $+Z$ neighbor. If any propagation in one of $+Y$ or $+Z$ directions intersects with another MCC, it will make a $+X$ turn until it can go back to the original direction. The propagation along $+Y$ and $+Z$ direction will continue from that new node. If the message can reach the surfaces $[0 : x_d, y_d : y_d, 0 : z_d]$, it will return this information back to s. The propagation of detection messages on other surfaces is similar. If the source s can know all the surfaces $[x_d : x_d, 0 : y_d, 0 : z_d]$, $[0 : x_d, y_d : y_d, 0 : z_d]$, and $[0 : x_d, 0 : y_d, z_d : z_d]$ can be reached, based on the sufficient and necessary condition for the existence of minimal path in Theorem 2, a minimal routing is feasible from d to s (i.e., a minimal routing from s to d). Figure 7 shows some samples of this feasibility check.

After the feasibility check, the routing process considering three preferred directions: the $+X$, $+Y$, and $+Z$ is similar to that in 2-D meshes which only considers $+X$ and $+Y$ directions. Figure 8 shows some samples of routing under our MCC model in 3-D meshes.

7 Conclusion

In summary, the contributions of this paper are listed as the following: (a) We have rewritten the MCC model in 2-D meshes without using global information so that the shape information at our boundaries can be used not only to ensure the existence of a minimal path, but also to form a minimal routing by routing decisions at intermediate nodes along the path. This routing will find a minimal path from source s to destination d whenever it exists. (b) We have extended the MCC model in 2-D meshes to 3-D meshes. The labelling process, edge identification process, edge construction, and boundary construction are presented to collect and distribute the MCC information for minimal routing. (c) Based on the information collected in the above limited-global-information model, a new minimal routing in 3-D meshes has been provided. It will find a minimal path from s to d whenever it exists. In our future work, we will extend our results to dynamic networks in which all the faulty components can occur during the routing process. Also, our results will be extended to higher dimension networks.

References

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

[2] 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.

[3] 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.

[4] Z. Jiang, J. Wu, and D. Wang. A new fault information model for fault-tolerant adaptive and minimal routing in 3-d meshes. Technical Report, West Chester University, 2004, available at http://www.cs.wcupa.edu/ zjiang/3d.pdf.

[5] 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.

[6] 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.

[7] 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.

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

[9] 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.

[10] J. Zhou and F. Lau. Fault-tolerant wormhole routing algorithm in 2d meshes without virtual channels. *Proc. of ISPA'2004*. 2004, pp. 688-697.

Proceedings of the 2005 International Conference on Parallel Processing (ICPP'05)
0190-3918/05 $20.00 © 2005 **IEEE**