A Limited-Global Fault Information Model for Dynamic Routing in 3-D Meshes *

Zhen Jiang
Dept. of Computer Science
West Chester University
West Chester, PA 19383

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

Abstract

In this paper, a fault-tolerant routing in 3-D meshes with dynamic faults is provided. It is based on an early work on fault-tolerant routing in dynamic 2-D meshes where faults occur during a routing process. Unlike many traditional models that assume all the nodes know global fault information, our approach is based on the concept of limited global fault information. First, a fault model called faulty block is used in which all faulty nodes in the system are contained in a set of disjoint faulty blocks. Then, the information of faulty block needs to be distributed to a limited number of nodes at the boundaries of the faulty block to avoid a message entering a detour area. When new faults occur, faulty blocks need to be reconstructed and their fault information needs to be redistributed. We study the limited distribution of fault information in 3-D meshes with dynamic faults. Our study shows that fault information can be distributed quickly during the routing process. In addition, the performance of routing process degrades gracefully in such a dynamic system.

1 Introduction

The *mesh-connected topology* is one of the most thoroughly investigated network topologies for multicomputers. Like 2-dimensional (2-D) meshes, 3-D meshes are low-dimensional meshes that have been commonly discussed due to structural regularity for easy construction and high potential of legibility of various algorithms. The current IBM Blue Gene project uses a 3-D mesh to build a supercomputer [1].

As the number of nodes in a mesh-connected multicomputer increases, the chance of failure also increases. The complex nature of networks also makes them vulnerable to disturbances which can be either deliberate or accidental. Therefore, the ability to tolerate failure is becoming increasingly important, especially in the communication subsystem. Several studies have been conducted which achieve fault tolerance by adding (or deleting) extra components of the system [5]. However, adding and deleting nodes and/or links requires modification of network topologies which may be expensive and difficult. We focus here on achieving fault tolerance using the inherent redundancy present in the mesh-connected multicomputer, without adding spare nodes and/or links.

Recently, a routing switching technique known as *pipelined circuit switching* (PCS) was developed by Gaughan and Yalamanichili [2]. Unlike wormhole routing, PCS allows backtracking during the path setup phase. Backtracking is a key element in providing fault tolerance in a system with dynamic faults. However, without fault information, the routing process may enter a region where all minimal paths to the destination are blocked by faulty nodes. Thus, PCS routing needs either to detour or to backtrack and causes routing difficulty which will increase routing delay and cause traffic congestion. The routing process here refers to the path setup phase. In PCS, the actual message sending occurs after a routing path is set up. Dynamic faults refer to ones appearing in the set-up phase only.

Most routing techniques are not suitable for networks with dynamic faults. In addition, a good analytical model is lacking while we resort mostly to simulations. A routing in 2-D meshes based on faulty block information, which is a special form of limited distribution of fault information, is presented in [4]. The information model exhibits desirable properties of self-stabilizing, self-optimizing, and self-healing in 2-D meshes with dynamic faults. In addition, the performance of routing process based on the fault information degrades gracefully. Compared with other fault information such as a routing table associated with each node, the update of faulty block information converges quickly, and it also reduces oscillation update caused by unstable information (also called inconsistent information). Compared with the routing-table-based routing, our approach reduces the memory requirement to store fault information in the whole network. When a disturbance occurs, only those affected nodes need to update fault information.

*This work was supported in part by NSF grants CCR 9900646 and ANI 0073736. Contact E-mail: jie@cse.fau.edu.

This paper is our first attempt to study the effect of dynamic faults on routing in 3-D meshes. We extend the results in 2-D meshes to 3-D meshes. First, a set of disjoint 3-dimensional faulty blocks are used to contain all faulty nodes in a 3-D mesh. The fault information will be propagated along the boundaries of a faulty block in our distribution process to help routing avoid routing difficulties. When dynamic faults occur, faulty blocks need to be reconstructed and their fault information needs to be redistributed. During the converging period, the routing process may experience more detours with inconsistent information. For a given pair of source and destination in 3-D meshes with dynamic faults, our fault-information-based PCS routing keeps certain levels of fault tolerance and adaptivity. Moreover, our results in 3-D meshes can be extended to n-D meshes easily. Throughout the paper, proofs to theorems are omitted and can be found in [3].

2 Preliminaries

3-D meshes. Each node u in a 3-D mesh is labeled as (x_u, y_u, z_u) or simply (x, y, z). Two nodes (x_v, y_v, z_v) and (x_u, y_u, z_u) are connected if their addresses differ in one and only one dimension. Basically, nodes along each dimension are connected as a linear array. The distance between two nodes u and v, $D(u, v)$, is equal to $\mid x_u - x_v \mid + \mid y_u - y_v \mid + \mid z_u - z_v \mid$. Assume node u is the current node, d is the destination node, and 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*.

Faulty block and its block information. The fault model in 3-D meshes uses a similar definition of faulty block as in 2-D meshes. In [6], Wu presents a model that activates the most non-faulty nodes from faulty block and uses the least steps to build faulty block in n-D mesh. It can be applied in 3-D meshes as:

Definition 1: *In a 3-D mesh, a non-faulty node is either marked enabled or disabled. Initially, all nonfaulty nodes are marked enabled. A nonfaulty node is marked disabled if there are two or more disabled or faulty neighbors along different dimensions. Connected disabled and faulty nodes form a faulty block, simply block.*

From Definition 1, we have the following definition of the six surfaces that are adjacent to the six surfaces of a faulty block.

Definition 2: *The six adjacent surfaces of a faulty block are defined as one unit distance away from the surfaces of the faulty block in each direction. Surfaces S_0 and S_3 are parallel to plane $X = 0$ with S_0 on the west side of S_3; surfaces S_1 and S_4 are parallel to plane $Y = 0$ with S_1 on the south side of S_4; surfaces S_2 and S_5 are parallel to plane $Z = 0$ with S_2 on the back side of S_5. The line connecting two adjacent surfaces is called an edge of the block. There are 12 different edges for a block. The node connected three edges of the block is called a corner, and there are 8 corners for a block (see in Figure 1 (b)).*

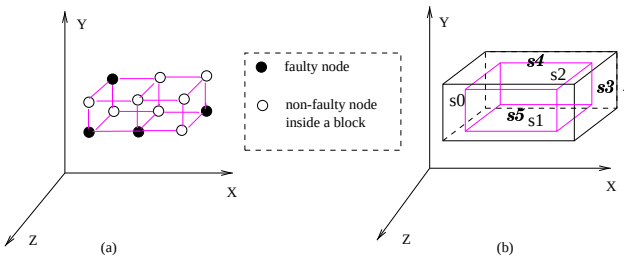

Figure 1. (a) Faulty block (Definition 1) and (b) its adjacent surfaces.

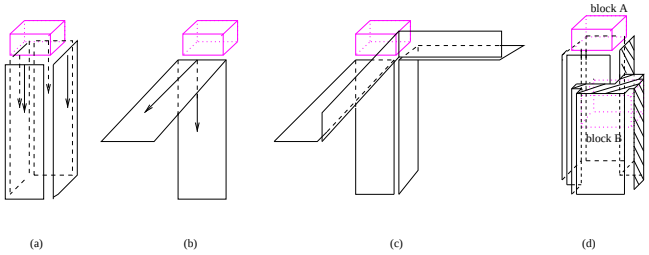

Figure 2. Boundaries of a block. (a) Boundary for S_4 starts from the edges of S_2. (b) Boundary of a block from the view of one edge. (c) Boundary of a block from the view of one corner. (d) Boundary of block A intersecting with block B.

As a result, a set of cube-type of faulty blocks is formed. For example (Figure 1 (a)), by four faults (3,5,4), (4,5,4), (5,5,3), and (3,6,3) in a 3-D mesh, the corresponding faulty block contains nodes which form a faulty block [3:5, 5:6, 3:4]. In general, [$x_{min}+1$:$x_{max}-1$, $y_{min}+1$:$y_{max}-1$, $z_{min}+1$:$z_{max}-1$] represents a faulty block with eight corners: (x_{min},y_{min},z_{min}), (x_{max},y_{min},z_{min}), (x_{max}, y_{max}, z_{min}), (x_{min}, y_{max}, z_{min}), (x_{min}, y_{min}, z_{max}), (x_{max}, y_{min}, z_{max}), (x_{max}, y_{max}, z_{max}), and (x_{min}, y_{max}, z_{max}).

For any two opposite adjacent surfaces, S_1 and S_4 in Figure 1 (b), if the routing message enters the area right below S_1 and its destination is right over S_4, there is no minimal path because the faulty block disconnects all shortest paths between the current node and the destination. The boundary surface, simply boundary, is used to enclose such an area. With the block information at each node on this boundary, the routing decision will avoid selecting a preferred direc-

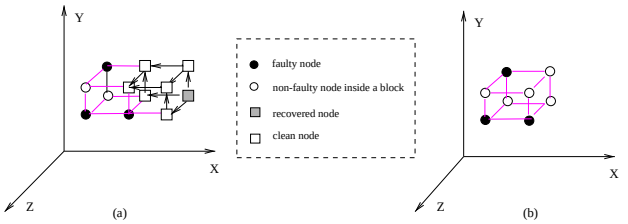

Figure 3. Recovery of a faulty node.

tion that leads the routing message to enter the dangerous area. That preferred direction is called *preferred but detour* direction, and such a routing is called *critical* routing. Otherwise, the selection of any preferred direction in routing decision will not affect the minimal routing, and the routing is called *non-critical* routing respectively.

As shown in Figure 2 (a), the boundary for S_4 starts from the edges of S_1 (except for the corner). The block information will propagate along this boundary in negative Y direction. Without any other blocks, the propagation of boundary information is forwarded node by node in one dimension until it meets the edge of the meshes. Figure 2 (b) shows boundaries of a block on the view of one edge, and Figure 2 (c) shows boundaries of a block on the view of a corner in 3-D meshes. If the boundary propagation for surface S_i starting from the edges of its opposite surface $S_{(i+3)\ mod\ 6}$ intersects with another block, from the first adjacent node of the second block it reaches, the propagation will merge into the surface S_i of the second block. Such propagation will continue along the other four adjacent surfaces of the second block. And it will merge into the boundary for S_i of the second block, which starts from the edges of $S_{(i+3)\ mod\ 6}$ of the second block (see in Figure 2 (d)).

3 Fault Information Constructions

In 3-D meshes, the shape of a faulty block may change during a routing process with the occurrence of new faults or by the recovery from faulty status to healthy ones. An extended enabled/disabled labeling scheme is introduced to quickly identify those non-faulty nodes in a faulty block that may cause routing difficulty.

Definition 3: *In a 3-D mesh, if any new fault occurs, Definition 1 is applied. If any node is recovered from faulty status, it is labeled clean. A disabled node is labeled clean if it has a clean neighbor and has not two faults in different dimensions. Once all its neighbors know its clean status in the clean process, the clean node is labeled enabled. Each enabled node applies Definition 1 until there is no status change.*

Based on Definition 3, there are three types of nodes after the procedure is stabilized: faulty nodes, enabled nodes, and disabled nodes. After a new fault occurs, an enabled node may change to disabled based on Definition 1 and affect the status of its enabled neighbors. This propagation will incur the construction of a new faulty block. Specifically, a recovered node is set to *clean*. This *clean* status will propagate to any disabled non-faulty neighbor and contribute further changes.

In Figure 3 (a), node (5,5,3) is recovered from faulty status. First, (5,5,3) is labeled clean and it triggers the change of status in its disabled neighbors (4,5,3), (5,6,3), and (5,5,4) to clean. The procedure continues until there is no more status change. The stabilized faulty blocks are shown in Figure 3 (b). Note that when (3,5,3) knows the status change of (4,5,3), it does not change its status to clean since it has two faulty neighbors in different dimensions. (4,5,3) changes to enabled once all its neighbors know its clean status. In the next round, it has one faulty neighbor (4,5,4) and one disabled neighbor (3,5,3). Then, this new enabled node will change to disabled when Definition 1 is applied.

This enabled/disabled labeling scheme in 3-D meshes can quickly identify those non-faulty nodes that may cause routing difficulty by labeling them disabled. Each active node collects its neighbors' status and updates its status. For each occurrence of a new fault or a new recovered node, the new node status can be easily determined through rounds of status exchanges among neighbors. Only those affected nodes update their status. Such a procedure is called *block construction*.

After block construction is incurred by new faults, a faulty block in 3-D meshes may enlarge its size, or a new faulty block may appear in the network. On the other hand, after block construction is incurred by some nodes recovered from faulty status, a faulty block in 3-D meshes may shrink its size or be divided into several small faulty blocks. The deletion process starts and will propagate along old boundaries when a corner of the old faulty block finds its existing condition cannot be satisfied. Also, to identify a new faulty block, the identification process starts whenever a *new* corner is formed.

From an initialization corner, an identification process has three phases. In phase one, two identification messages are initiated at that corner, for example, $C(x_{max}, y_{min}, z_{max})$ in Figure 4 (a). They carry the position information of node C, the partial block information, and will be sent along the X and Y dimensions on the edges of the block ($y = y_{min} \wedge z = z_{max}$ along the X dimension and $x = x_{max} \wedge z = z_{max}$ along the Y dimension). In phase two, at each node on the edges, for example, E(x_e, y_{min}, z_{max}) on edge $y = y_{min} \wedge z = z_{max}$, two identification messages are initiated and carry the same partial block information. They will be sent to those two neighbors adjacent to the section of this faulty block on plane $x = x_e$,

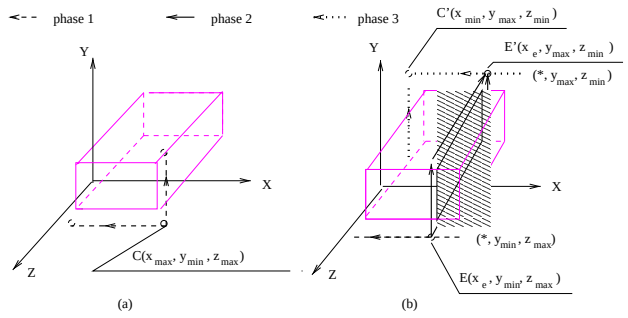

Figure 4. Information collection in the identification process.

$(x_e, y_{min}+1, z_{max})$ and $(x_e, y_{min}, z_{max}-1)$. Such propagation will continue until the message traverses all the enabled nodes adjacent to the section of the faulty block on plane $x = x_e$. These two messages from E(x_e, y_{min}, z_{max}) will reach the node E'(x_e, y_{max}, z_{min}) on the opposite edge $y = y_{max} \wedge z = z_{min}$. With the position information of E and E', the section of block on plane $x = x_e$ $[y_{min}+1 : y_{max}-1, z_{min}+1 : z_{max}-1]$ is identified (see in Figure 4 (b)). In a similar way, for each node E(x_{max}, y_e, z_{max}) on edge $x = x_{max} \wedge z = z_{max}$, the section of block on plane $y = y_e$: $[x_{min}+1 : x_{max}-1, z_{min}+1 : z_{max}-1]$ is identified at node E'(x_{min}, y_e, z_{min}). In phase three, the identified information is collected by a message from the edge neighbor of corner $(x_{max}, y_{max}, z_{min})$ along the X dimension (see in Figure 4 (b)) and a message from the edge neighbor of corner $(x_{min}, y_{min}, z_{min})$ along the Y dimension. These two messages will arrive at the opposite corner $C^{'}(x_{min}, y_{max}, z_{min})$. With the position information of two corners $C(x_{max}, y_{min}, z_{max})$ and $C^{'}(x_{min}, y_{max}, z_{min})$, the block is identified and block information $[x_{min}+1 : x_{max}-1, y_{min}+1 : y_{max}-1, z_{min}+1 : z_{max}-1]$ is formed.

Algorithm 1: Block construction and information distribution

1. Block construction by applying Definition 3.
2. Identification of adjacent nodes and corners.
3. Identification process from a new corner:
   (a) Two identification messages (one along X dimension and one along Y dimension) are sent along the edges from that initialization corner, until they reach the end of each edge.
   (b) As each node E(x_e, y_e, z_e) on the edge along the X/Y dimension received the above message, two identification messages are sent along the enabled nodes adjacent to the new block on the plane $x = x_e / y = y_e$, until they reach the node E' on the opposite edge. Each YZ plane or XZ plane section of the new block is identified with the position information of its E and E'.
   (c) The identified partial block information at each E' is collected and transferred along the edges to form faulty block information at the opposite corner of that initialization corner.
4. The faulty block information is propagated to all the nodes on the adjacent surfaces of the block by the above procedure from the opposite corner back to the initialization corner. A boundary construction is activated at each node on the edge of the new block that receives consistent faulty block information.

After that, by using the above procedure from the corner $C^{'}$ back to the corner C, the identified block information $[x_{min}+1 : x_{max}-1, y_{min}+1 : y_{max}-1, z_{min}+1 : z_{max}-1]$ is propagated to all the nodes on the adjacent surfaces of this block. To guide the routing process, the faulty block information is transferred along the boundary of the new faulty block from the nodes on the edges when they get the identified information. In our reactive model, if any node already has the new faulty block information, there is no need to start new boundary propagation. This propagation may also incur a deletion of out of date boundaries and update the boundaries of other faulty blocks. Such a procedure is called *boundary construction*. All these procedures are shown in Algorithm 1 and can easily be extended to n-D meshes.

Note that each identification message of the identification process is expected to go straight until the end of the edge. If there is a faulty or disabled neighbor in the forwarding direction, the new block is not stable. At each node in phase 3 of the identification process, there is a check of identified sections. If there is a different section, the block is also not stable. In both cases, the message is discarded at the current node to avoid generating incorrect faulty block information. If any message is discarded during the identification process, the opposite corner cannot receive two messages at the same time. That is, the shape of block may not be the exact one indicated by the positions of the initialization corner and its opposite corner. TTL is associated with each message in our 3-D meshes, and the corresponding message will be discarded once the time expires. After two messages meet at the opposite corner and form the block information, the procedure is reused to propagate identified information. But this time, the stable block ensures that the procedure will end at the initialization corner successfully.

4 Faulty-block-information-based Routing

Algorithm 2: Fault-information-based PCS routing

s	source node (x_s, y_s, z_s)
d	destination node (x_d, y_d, z_d)
u	the current node (x_u, y_u, z_u)
F	number of faults in a given $m \times m \times m$ mesh and $F < 3 * m$
f_i	i^{th} fault occurrence where $i \in \{1, 2, ..., 3 * F\}$
t_i	occurrence time of f_i
d_i	the interval between two consecutive fault occurrences f_i and f_{i+1}, i.e., $d_i = t_{i+1} - t_i$
t	start time of a routing process
p	number of fault changes before the routing starts
D	distance from source s to destination d
D_i	distance from the current node u to the destination d at time t_i
a_i	total steps the stabilizing block construction for f_i converges
a_{max}	$\max\{a_i\}$
e_{max}	maximum length of edges of faulty blocks
b_i	total rounds for the stabilizing identification process
c_i	total rounds for the stabilizing boundary propagation
λ	rounds of fault information collection and propagation per step

Table 1. List of notations used.

1. If the current node u is disabled, backtrack; otherwise,
2. pick an unused outgoing direction with the highest priority. The address of u and the direction selected is recorded in the message header.
3. If there is no unused outgoing direction, backtrack.
4. If the message is backtracked to the source, the destination is unreachable.

Faulty-block-information-based routing in 2-D meshes (see in [4]) can be easily extended to 3-D meshes. Algorithm 2 shows the routing process in 3-D meshes. For the current node $u(x_u, y_u, z_u)$ $(\neq d)$ with the incoming direction and five possible outgoing directions, the routing selects one of the directions as the forwarding direction in the priority order of preferred, spare (along with block), preferred but detour, and incoming directions. It is also noted that each forwarding direction at a participant node cannot be used again. Thus, like our routing in 2-D meshes, each routing header here includes a destination address and a list of used-directions for each forwarding node along the path. This is because the system is dynamic and the priority of directions may also change. Theorem 1 ensures the effectiveness of our fault information model when faults are recovered in the networks.

Theorem 1: *The constructions of the fault recovery do not affect the optimal routing.*

5 Dynamic Fault Model

We adopt the same model of a 2-D mesh for activities in a node of a 3-D mesh. The model used represents a reactive approach where information updates are done only when there is a change of status of at least one neighbor. At each step, every node in an $m \times m \times m$ mesh starts with fault detection of adjacent links and nodes, and then collects and distributes three kinds of fault information: block information, identification information, and boundary information through λ rounds of exchanges and updates. The disabled/enabled status propagation, any message header of identification information propagation, block information propagation and canceling propagation advance one hop further at each round. Before the end of each step, based on the fault information, a routing decision selects a forwarding node to forward the routing message. Therefore, every routing message advances one hop along with its routing path at each step. The actions within a step are shown in Figure 5.

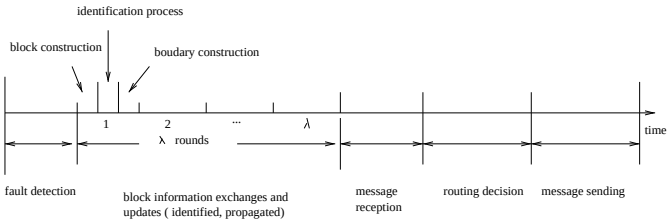

Figure 5. Actions within a step.

To simplify the discussion, it is assumed that any adjacent faults, links, and nodes are detected at the fault detection phase (any faults occurring after the fault detection phase will be detected at the next step). During the update of fault information, each node can also receive one incoming message (if any). It is also assumed that the action "message receive" occurs right before the "routing decision", as shown in Figure 5.

We assume there are at most F faulty nodes in a 3-D mesh network, including dynamically generated faults. Faults f_1, f_2, ..., f_F occur at time $t_1, t_2, ..., t_F$; respectively, where $t_{i+1} - t_i = d_i$ $(1 \leq i < F)$. To simplify our discussion, it is assumed that fault information updating in the mesh is already stabilized before the next fault occurrence, and there is no fault that occurs at the outmost surfaces of a 3-D mesh. Based on the properties discussed in [6], there is no disconnected area in such a mesh. It means that there is always a path from an enabled source to an enabled destination. Before a routing message is initiated at time t, it is assumed that the first p faults have already occurred; that is, $p = \max\{l | t_l \leq t\}$. D represents the distance from source s to destination d. D_i represents the distance from the current node u to the destination d at time t_i when the i^{th} fault occurrence $(1 \leq i \leq F)$ occurs. Before the start time t, the routing message is at its source and $D_i = D = \mid x_s - x_d \mid + \mid y_s - y_d \mid + \mid z_s - z_d \mid$ $(i \leq m)$. For the i^{th} fault change, the block construction will be stabilized in $\lceil \frac{a_i}{\lambda} \rceil$ steps, the identification process will be stabilized in $\lceil \frac{b_i}{\lambda} \rceil$ steps, and the boundary construction will be stabilized in $\lceil \frac{c_i}{\lambda} \rceil$ steps. We assume that $d_i > \max\{\frac{a_i + b_i + c_i}{\lambda}\}$. Therefore, before the next occurrence of fault (t_{i+1}), the boundaries for the faulty blocks at t_i are already stabilized. To simplify the discussion, Table 1

summarizes the notations used here.

The discussion below will show that a routing message has no more detours than an upper bound with the guide of fault information.

6 Detour Analysis

In [6], Wu defines safe node in n-D meshes, and it can be applied in 3-D meshes as follows:

Theorem 2: *Assume that node (0,0,0) is the source and node (x_d, y_d, z_d) is the destination. If there is no faulty block that intersects with the section of $[0 : x_d]$ along X-axis, the section of $[0 : y_d]$ along Y-axis, and the section of $[0, z_d]$ along Z-axis, the source node is safe (to the routing); otherwise, it is unsafe.*

If the source is safe, a minimal path is guaranteed to the destination as long as no new fault occurs during the routing process. When a new fault occurs, before the new fault information is stabilized in all three construction procedures, a routing message may use inconsistent information and enter a detour area. If the size of a faulty block is limited by e_{max}, the number of detours the routing needs is limited. Enlarging the interval will increase the number of optimal steps in each interval. Since the distance from s to d is limited in a 3-D mesh, the maximum number of intervals before the routing message reaches its destination can be reduced by this way; in other words, the number of total detours is reduced.

Theorem 3: *For any fault-information-based routing from a safe source s to an enabled destination d (not in any faulty block), if $D(i) > 0$: (a) $D(i) = D$ when $i \leq p$, (b) $D(i) \leq D - (d_{i-1} - t + t_p) + 2 * (a_{i-1} + e_{max})$ when $i = p + 1$, and (c) $D(i) \leq D(i-1) - d_{i-1} + 2 * (a_{i-1} + e_{max})$ when $p + 1 < i \leq F$.*

Assume that $m + q - 1$ is the largest index for the fault such that $D(m + q - 1) > 0$. That is, the f_{m+q-1} is the last fault change that could affect the routing process. q is the maximum number of intervals in which the routing message detours at least once. Based on Theorem 3, we have $q = \max\{l | D + t - t_p - \sum_{i=p}^{p+l-2}(d_i - 2 * a_i - 2 * e_{max}) > 0\}$ Let us consider several cases for a routing with a minimal path before it starts: (a) The routing message will get closer to the destination between two occurrences of consecutive faults as long as these two faults are separated by more than $2(a_{max} + e_{max})$ time steps. (b) When d_i's are uniform, i.e., $d_i = k$, $q = \max\{l | (l-1)(d_i - 2 \times (a_i + e_{max})) < D + t - t_p \leq D + k\}$. Then $q \leq \lfloor \frac{D+k}{k-2\times(a_{max}+e_{max})} \rfloor$. (c) Maximum detour for a message with a minimal path before it starts is $(a_{max} + e_{max}) \times \lfloor \frac{D+k}{k-2\times(a_{max}+e_{max})} \rfloor$.

Theorem 4: *For a routing message from a safe source s to an enabled destination d in an $m \times m \times m$ 3-D mesh, the routing process will end in the following k intervals and $k \leq \max\{l | D + t - t_p - \sum_{i=p}^{p+l-2}(d_i - 2 * a_i - 2 * e_{max}) > 0\}$. The number of maximum detours is $k * (e_{max} + a_{max})$ and the probability of maximum detour in a routing is no more than $\prod_{i=p}^{p+k-1}(\frac{1}{m^3})$.*

Theorem 3 shows the maximum detours in each interval for a routing message from a safe source. A routing from an unsafe source to its destination may need more detours. The discussion below extends the above result for any routing message, including that from an unsafe source.

Theorem 5: *Given a routing algorithm, if there is a path with length L from an enabled source s to an enabled destination d in a $m \times m \times m$ 3-D mesh, the routing process will end in the following k intervals and $k \leq \max\{l | L + t - t_p - \sum_{i=p}^{p+l-2}(d_i - 2 * a_i - 2 * e_{max}) > 0\}$.*

7 Conclusions

In this paper, we studied an upper bound of maximum detours in 3-D meshes with dynamic faults using fault-tolerant routing algorithm, based on our limited global information. The faulty block information associated with each node on the boundaries has been used to present limited global information. Our study shows that such limited global information can be collected and distributed quickly to help the routing process. Applying this approach to other fault models are interesting problems for future research.

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] P. T. Gaughan and S. Yalamanchili. A family of fault-tolerant routing protocols for direct multiprocessor networks. *IEEE Transactions on Parallel and Distributed Systems.* 6, (5), May, 1995, 482-497.

[3] Z. Jiang and J. Wu. A limited-global information model for dynamic routing in 3-d meshes. TR-CSE-02-02, Dept. of Computer Science and Engineering, Florida Atlantic University.

[4] Z. Jiang and J. Wu. A limited-global-information model for dynamic routing in 2-d meshes. *Proc. of the 3rd Workshop on Parallel and Distributed Scientific and Engineering Computing with Applications*, April 2002. CD-ROM.

[5] N. F. Tzeng and G. Lin. Maximum reconfiguration of 2-d mesh systems with faults. *Proc. of 1996 International Conference on Parallel Processing.* 1996, 77-84.

[6] J. Wu. A fault-tolerant adaptive and minimal routing approach in n-d meshes. *Proc. of International Conference on Parallel Processing (ICPP)*, pages 431–438, August 2000.