A Limited-Global Fault Information Model for Dynamic Routing in \( n \)-D Meshes

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

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

Abstract

In this paper, a fault-tolerant routing in \( n \)-D meshes with dynamic faults is provided. It is based on an early work on fault-tolerant routing in dynamic 2-D meshes [9] and 3-D meshes [10] 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 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. In this case, the update of fault information and the routing process proceed hand-in-hand. During the converging period, the routing process may experience more detours with inconsistent information. We study the limited distribution of fault information in \( n \)-D meshes with dynamic faults. Our study shows that fault information can be distributed quickly to help the routing process. Therefore, the performance of routing process degrades gracefully in such a dynamic system.

1 Introduction

In a multicomputer system, a collection of processors (also called nodes) works together to solve large application problems. These nodes communicate and coordinate their efforts by sending and receiving messages through the underlying communication network. Thus, the performance of such a multicomputer system is dependent on the end-to-end cost of communication mechanisms. Routing is the process of finding a path from the source node to the destination node in a given system. Routing time of messages is one of the key factors that are critical to the performance of multicomputers.

The mesh-connected topology is one of the most thoroughly investigated network topologies for multicomputers due to structural regularity for easy construction and high potential of legibility of various algorithms [5, 7, 8]. 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 sub-system. Several studies have been conducted which achieve fault tolerance by adding extra components to the system [4, 13]. However, adding nodes and/or links requires modification of network topologies which may be very expensive and difficult. We focus here on achieving fault tolerance using the inherent redundancy present in the mesh-connected multicomputer, without adding any spare node or link.

Recently, a switching technique for routing, known as pipelined circuit switching (PCS), was developed by Gaughan and Yalamanchili [6]. 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 detour or backtracking 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 that appear in the set-up phase only.

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 [9]. First, all faulty nodes are contained in disjointed faulty blocks by applying a labeling process. Routing is based on faulty block information
distributed at the nodes along the boundary lines of faulty blocks to avoid routing difficulties. When dynamic faults occur, faulty blocks need to be reconstructed and their fault information needs to be redistributed. In this case, the update of fault information and the routing process proceed hand-in-hand. During the converging period, the routing process may experience more detours with unstable information (also called inconsistent information). Results in [9] show that our fault information can be distributed quickly. In addition, the performance of routing process degrades gracefully in such a dynamic system. 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 inconsistent information. Moreover, 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. In [10], our results in 2-D meshes are extended to 3-D meshes.

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. This paper is our first attempt to study the effect of dynamic faults on routing in n-D meshes (n = 2, 3, ...). First, a set of disjoint n-dimensional faulty blocks is used to contain all faulty nodes in an n-D mesh. The fault information will be propagated along the boundaries of a faulty block in our collection and distribution process to avoid routing difficulties. Our information model exhibits desirable properties of self-stabilizing, self-optimizing, and self-healing. For a given source and destination pair in n-D meshes with dynamic faults, our fault-information-based PCS routing keeps certain levels of fault tolerance and adaptivity.

2 Preliminaries

2.1 k-ary n-dimensional meshes

A k-ary n-dimensional (n-D) mesh with \( N = k^n \) 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 connected 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. Each node \( u \) in an n-D mesh is labeled as \( (u_1, u_2, ..., u_n) \). The distance between two nodes \( u \) and \( v \) (\( D(u, v) \)) is equal to \( |u_1 - v_1| + |u_2 - v_2| + ... + |u_n - v_n| \). 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.

2.2 Faulty block and its related information

Most literature on fault-tolerant routing uses disjoint rectangular blocks [1, 2, 3, 11, 12] to model node faults (link faults can be treated as node faults) and to facilitate routing in mesh networks. In [14], Wu presents a model that activates most non-faulty nodes from a faulty block and uses the least number of steps to build a faulty block in an n-D mesh:

**Definition 1:** In an n-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 called as block.

Respectively, we define the adjacent nodes and corners of an n-D block as:

**Definition 2:** In an n-D mesh, an adjacent node is an enabled node with a neighbor in the block. A 2-level corner is an enabled node with two adjacent nodes of the same block in different dimensions. Recursively, an n-level edge node is an \((n - 1)\)-level corner and an n-level corner is an enabled node with n n-level edge neighbors of the same block.

As a result, a set of cube-type 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 block contains nodes which form a block [3:5, 5:6, 3:4]. Figure 2
shows the definition of a 3-level corner of block [3:5, 5:6, 3:4]: (6, 4, 5). It has three 3-level edge neighbors: (5, 4, 5), (6, 5, 5) and (6, 4, 4). Each 3-level edge node is a 2-level corner and has two neighbors adjacent to the block. For example, (5, 4, 5) has neighbors (5, 5, 5) and (5, 4, 4) adjacent to the block. In 3-D meshes, \( [x_{\text{min}} + 1 : x_{\text{max}} - 1, y_{\text{min}} + 1 : y_{\text{max}} - 1, z_{\text{min}} + 1 : z_{\text{max}} - 1]\) represents a block with eight corners: \((x_{\text{min}}, y_{\text{min}}, z_{\text{min}}), (x_{\text{max}}, y_{\text{min}}, z_{\text{min}}), (x_{\text{min}}, y_{\text{max}}, z_{\text{min}}), (x_{\text{min}}, y_{\text{min}}, z_{\text{max}}), (x_{\text{max}}, y_{\text{max}}, z_{\text{max}}), (x_{\text{max}}, y_{\text{min}}, z_{\text{max}}), (x_{\text{max}}, y_{\text{max}}, z_{\text{min}}),\) and \((x_{\text{min}}, y_{\text{max}}, z_{\text{max}})\).

In [10], we have the following definition of the six surfaces of a block in 3-D meshes that are adjacent to the six surfaces of a block.

**Definition 3:** The six adjacent surfaces of a block in 3-D meshes are defined as one unit distance away from the surface of the 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 connecting three edges of the block is called a corner, and there are 8 corners for a block (see in Figure 1 (b)).

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 block disconnects all shortest paths between the current node and the destination. The boundary surface, simply boundary, is used to enclose such a dangerous area. With the block information at each node on this boundary, the routing decision will avoid selecting a preferred direction 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 the routing decision will not affect the minimal routing, and the routing is called non-critical routing respectively.

As shown in Figure 3 (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 the negative \(Y\) direction. Without any other blocks, the propagation of boundary information is forwarded node by node in one dimension until it meets the outmost surface of the meshes. Figure 3 (b) shows boundaries of a block on the view of one edge, and Figure 3 (c) shows boundaries of a block on the view of one corner. (d) shows boundaries of a block A intersecting with block B.

**Figure 3.** Boundaries of a block in 3-D meshes. (a) Boundary for \(S_2\) starts from the edges of \(S_3\). (b) Boundary of a block on the view of one edge. (c) Boundary of a block on the view of one corner. (d) Boundary of block A intersecting with block B.

\[s_{i+3} = \text{mod}\ 6\]

The boundary of a block in an \(n\)-D mesh starts from one of its \(n\)-level corners. First, the boundary propagation will go through all the \(n\)-level edge nodes and reach other \(n\)-level corners. Each \(n\)-level edge node is also an \((n - 1)\)-level corner and has \(n - 1\) \((n - 1)\)-level edge nodes. The corresponding connecting direction is called surface direction. Then, once an \(n\)-level edge node receives the boundary information, the boundary propagation will continue along \(n - 1\) directions which are opposite to its surface directions. After that, the boundary propagation will continue along such a direction until it meets the outmost surface of this \(n\)-D mesh. Figure 3 (b) shows such a step of boundary propagation in 3-D meshes. If the boundary propagation intersects with another block, from the first node which has information of both boundaries, the propagation will merge into the boundary of the second block. Figure 3 (d) shows such a step of boundary propagation in \(3\)-D meshes. \n
### 3 Fault Information Constructions in \(n\)-D Meshes

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

**Definition 4:** In an \(n\)-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 no 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 4, 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 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 4 (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 further status change. The stabilized blocks are shown in Figure 4 (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 n-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. Algorithm 1 shows the whole procedure of block construction.

Algorithm 1: block construction

All non-faulty nodes are enabled; repeat
example, E(\(x_e, y_{min}, z_{max}\)) on edge \(y = y_{min} \land z = z_{max}\), two identification messages are initiated and carry the same partial block information. They will be sent to two neighbors \((x_e, y_{min} + 1, z_{max})\) and \((x_e, y_{min}, z_{max} - 1)\), which are adjacent to the section of this block on plane \(x = x_e\). Such propagation will continue until the message traverses all the enabled nodes adjacent to the section of the 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 \(y = y_{max} \land z = z_{min}\). With the position information of E and E’, the section of block on plane \(y = y_e\), \([y_{min} + 1 : y_{max} - 1, z_{min} + 1 : z_{max} - 1]\), is identified (see in Figure 5 (b)). In a similar way, for each node E(\(x_{max}, y_e, z_{max}\)) on edge \(x = x_{max} \land z = z_{max}\), 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 5 (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.

After an n-level identification process, by using the above procedure from the opposite n-level corner back to the initialization n-level corner, the identified block information is propagated to all the adjacent nodes, edge nodes and corners of this block (see in Figure 6). To guide the routing process, the block information is transferred along the \((n - 1)\)-dimensional boundary of the new block from the n-level edge nodes when they get the identified information. In our reactive model, if any node already has the new 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 blocks. Such a procedure is called boundary construction. All these procedures are shown in Algorithm 2.

Algorithm 2: Block construction and information distribution

1. Block construction by applying Definition 4.
2. Identification of adjacent nodes and all levels of edge nodes and corners.
3. n-level identification process from a new n-level corner:
   (a) \(n - 1\) identification messages are sent to n-level edge neighbors until all connected n-level edge nodes receive the identification message.
   (b) At each n-level edge node which is also an \((n - 1)\)-level corner, an \((n - 1)\)-level identification process is activated. The identified information will be collected at the opposite \((n - 1)\)-level corner.
   (c) The identified partial block information is collected and transferred to the opposite n-level corner to form block information.
4. By using the above procedure from the opposite n-level corner back to the initialization n-level corner, the identified block information is propagated to all the adjacent nodes, edge nodes and corners. A boundary construction is activated at each n-level corner node of that new block that receives consistent block information.

Note that each identification message of the identification process is expected to go straight to the outmost surface of the meshes. 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
block information. If any message is discarded during the
identification process, the opposite corner cannot receive all
messages at the same time. That is, the shape of the block
may not be the exact one indicated by the positions of the
initialization corner and its opposite corner. TTL is associ-
ated with each identifying message in our n-D meshes, and
the corresponding message will be discarded once the time
expires. After n – 1 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 initializa-
tion corner successfully.

4 Faulty-block-information-based Routing

Algorithm 3: Fault-information-based PCS routing

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 [9]) can be easily extended to n-D meshes. Algo-
rithm 3 shows the routing process in n-D meshes. It is noted
that at a boundary, if it is critical, one preferred direction
changes to preferred but detour direction. Otherwise, there is
no preferred but detour direction. For the current node
u( ≠ d) with the incoming direction and 2n-1 possible out-
going 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 incom-
ing directions.

After an n-level identification process, by using the
above procedure from the opposite n-level corner

It is also noted that each forwarding direction at a partic-

ipant 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 change. Theorem 1 en-

sures 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.

| s | source node |
| d | destination node |
| u | the current node |
| F | number of faults in a k-ary n-D mesh |
| f_i | i-th fault occurrence where i ∈ {1, 2, ..., 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 occurrences before the routing starts |
| D | distance from s to d |
| D(i) | distance from u to d at time t_i |
| a_i | total steps in which the stabilizing block construction
for f_i converges |
| a_{max} | max{a_i} |
| e_{max} | maximum length of edges of blocks |
| b_i | total rounds for the stabilizing identifying construction |
| c_i | total rounds for the stabilizing boundary construction |
| λ | number of rounds of fault information constructions at
each step |

Table 1. List of notations used in the paper

Proof: When nodes are recovered from a block, the old
block shrinks in each direction if it still exists. Suppose
that the block shrinks k hops in +X direction. According
to our procedure of block construction and information dis-
bution, the deletion of the old boundary will be activated
from a certain node after the propagation of the new bound-
dary reaches it. Assume that a routing meets the boundary. If
the routing can access the dangerous area, it will meet the
constructed boundary of new blocks. The detour caused by
the new block can also be avoided.

5 Dynamic Fault Model

We adopt the same model of a 2-D mesh for activities in
a node of an n-D mesh. The model used represents a re-
active approach where information updates are done only
when there is a change of status of at least one neigh-
bor. At each step, every node in an n-D mesh starts with
fault detection of adjacent links and nodes, and then col-
lects and distributes three kinds of fault information: block
information, identifying information, and boundary inform-
ation through λ rounds of exchanges and update. The
disabled/enabled status propagation, any message header of
identifying/identified propagation, block information prop-
gagation 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 and then the message
is sent to this forwarding node. Therefore, every routing
message advances one hop along with its routing path at each step. The actions within a step are shown in Figure 7 (a).

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 7 (a). The model used represents a reactive approach where information update is done only when there is a change of information of at least one neighbor.

We assume there are at most $F$ faulty nodes in an $n$-D mesh network, including dynamically generated faults. Faults $f_1, f_2, ..., f_F$ occur at time $t_1, t_2, ..., t_F$ (see in Figure 7 (b)); 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 surface of an $n$-D mesh. Based on the properties discussed in [14], there is no disconnected area in such a mesh. It means that there is always a path between the enabled source and the 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 occurs ($1 \leq i \leq F$). Before the start time $t$, the routing message is at its source and $D(i) = D = |x_s - x_d| + |y_s - y_d| + |z_s - z_d|$ ($i \leq m$). For the $i^{th}$ fault change, the block construction will be stabilized in $\left\lceil \frac{F}{p} \right\rceil$ steps, the identifying construction will be stabilized in $\left\lceil \frac{F}{p} \right\rceil$ steps, and the boundary construction will be stabilized in $\left\lceil \frac{F}{p} \right\rceil$ steps. We assume that $d_i > \max\{a_i + b_i + c_i\}$. Therefore, before the next occurrence of fault $(t_{i+1})$, the boundaries for the 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 [14], Wu defines safe node in $n$-D meshes as follows:

**Theorem 2 [14]:** Assume that node $(0,0, ..., 0)$ is the source and node $(u_1, u_2, ..., u_p)$ is the destination. If there is no block that intersects with the section of $[0 : u_1]$ along each axis for all $i \in \{1, 2, ..., n\}$, 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.

**Theorem 3:** For any fault-information-based routing from a safe source $s$ to an enabled destination $d$, if $D(i) > 0$:

$$
\begin{cases}
D(i) = D & \text{where } i \leq p \\
D(i) \leq D - (d_{i-1} - t + t_p - 2*\alpha_{i-1} - 2* \epsilon_{\text{max}}) & \text{where } i = p + 1 \\
D(i) \leq D(i-1) - (d_{i-1} - 2*\alpha_{i-1} - 2* \epsilon_{\text{max}}) & \text{where } i > p + 1
\end{cases}
$$

**Proof:** Since there is only one new block in each interval, the worst case for a routing message is that it goes along with the block construction and needs detours along the block after that. It needs at most $2*\alpha_i + 2* \epsilon_{\text{max}}$ extra steps in each interval $d_i$ ($F > i \geq p$). After the first $d_p - t + t_p - 2\alpha_p - 2\epsilon_{\text{max}}$ steps in interval $d_p$, the routing message reaches a node $D(p + 1)$ from its destination. It advances at least $d_p - t + t_p - 2\alpha_p - 2\epsilon_{\text{max}}$ steps closer to its destination. For any other interval $d_i$, if $D(i + 1) > 0$, the routing message advances at least $d_i - 2\alpha_i - 2\epsilon_{\text{max}}$ steps closer to the destination.

**Theorem 4:** For a routing message from a safe source $s$ to an enabled destination $d$ in an $n$-D mesh, the routing process will end in the following $k$ intervals and $k \leq \max\{l | D + t + t_p - \sum_{i=p}^{i-1} (d_{i-1} - 2*\alpha_i - 2* \epsilon_{\text{max}}) > 0\}$. The number of maximum detours is $k * (\epsilon_{\text{max}} + \alpha_{\text{max}})$.

**Proof:** Since there is only one new block in each interval, it needs at most $2*\alpha_i + 2* \epsilon_{\text{max}}$ extra steps in each interval. The routing message advances at least $d_p + t_p - t - 2*\alpha_p - 2* \epsilon_{\text{max}}$.
$e_{max}$ steps in the first interval. For any other interval $d_i$, the routing message advances at least $d_i - 2 + a_i - 2 + e_{max}$ steps. If the routing will end in the following $k$ intervals from time $t$, the routing message at least advanced $\sum_{i=p}^{p+k-2}(d_i - 2 + a_i - 2 + e_{max})$ steps along the path. Since the routing has a safe source node, it has a path with $D$ hops to the destination at start time $t$. $k \leq \max\{t|D + t - t_p - \sum_{i=p}^{p+k-2}(d_i - 2 + a_i - 2 + e_{max}) > 0\}$.

A routing from an unsafe source to its destination may need more detours. The following discussion extends the above results for any routing message including that from an unsafe source.

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

**Proof:** The routing will advance along the path until the path is disconnected by a new block. During each interval, the routing needs at most $2(e_{max} + e_{max})$ extra steps to go back to the path. If the routing will end in the following $k$ intervals from time $t$, the routing message at least advanced $\sum_{i=p}^{p+k-2}(d_i - 2 + a_i - 2 + e_{max})$ steps along the path and $k \leq \max\{|L + t - t_p - \sum_{i=p}^{p+k-2}(d_i - 2 + a_i - 2 + e_{max}) > 0\}$.

7 Conclusions

In this paper, we studied an upper bound of maximum detours in our limited-global-information based fault-tolerant routing in $n$-D meshes ($n \geq 3$) with dynamic faults. The block information associated with each node on the boundaries has been used to present limited global information. Fault information construction including fault detection, fault information exchanges and update, message reception, routing decision, and message sending have been proposed which are applicable to $n$-D meshes. Our study shows that such limited global information can be collected and distributed quickly to help the routing process. Application of this approach to other fault models is an interesting problem for future research.

8 Acknowledgments

This work was supported in part by NSF grants CCR 9900646, CCR 0329741, ANI 0073736, and EIA 0130806.

References


