A Limited-global Fault Information Model for Dynamic Routing in 2-D Meshes *

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

Abstract

In this paper, a fault-tolerant routing in 2-D meshes with dynamic faults is provided. It is based on an early work on minimal routing in 2-D meshes with static faults. 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 boundary lines of block to avoid a message entering a detour area. We study the limited distribution of fault information in a dynamic network where faults occur during a routing process. Our study shows that fault information can be distributed quickly to help the routing process. In addition, the performance of routing process degrades gracefully in such a dynamic system. PCS routing scheme used in this paper and its experimental results show that the fault tolerance and scalability can be offered for a routing message.

Keywords: *Dynamic faults, fault tolerance, hypercubes, routing, safety levels*

*This work was supported in part by NSF grant CCR 9900646.

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 in mesh-connected networks (also called k-ary n-dimensional meshes), such as 2-D meshes, has been commonly discussed due to the structural regularity for easy construction and the high potential legibility for variety of algorithms. Some multicomputers were built based on 2-D meshes [3, 4, 5, 6].

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 subsystems. Several studies have been conducted which achieve fault tolerance by adding (or deleting) extra components of the system [1, 7]. However, adding and deleting nodes and/or links require 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) is 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, 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 so-called 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 appeared in the set-up phase only.

An optimal routing algorithm using faulty block information, which is a special form of limited distribution of fault information, is presented in [8]. Comparing with other fault information such as a routing table associated with each node, the update of faulty block information converges quickly. It reduces oscillation update caused by unstable information (also called inconsistent information). First, all faulty nodes are contained in disjointed faulty blocks by applying a labeling process. The main idea of routing is the use of limited distribution of block information at the nodes

along the boundary lines of faulty blocks to avoid routing difficulties. Compared with the routing-table-based routing, this 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. However, the approach in [8] uses a static fault model; that is, it is assumed that no new fault will occur during a routing process.

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 inconsistent information. Most of routing techniques are not suitable for networks with dynamic faults. In addition, a good analytical model is lacking while we resort mostly simulations.

This paper is our first attempt to study the effect of dynamic faults on routing in 2-D meshes. We provide a collection and distribution process of fault information which exhibits desirable properties of self-stabilizing, self-optimizing, and self-healing. In a 2-D mesh, based on the extended safety level (a distance vector to closest faulty blocks along different dimensions) a safe source node can determine a minimal routing path. A detour may occur only if new faults occur during the routing process. Unlike a safe source, a routing message from an unsafe source needs one or more detours to reach its destination. We propose a fault-information-based PCS routing which keeps the adaptivity and fault-tolerance for the routing message from either a safe source or an unsafe source.

The paper is organized as follow: In Section 2, the limited-global information model and its relevant properties are introduced. Some related research work are discussed. A collection and distribution process of the information model is presented in Section 3. In Section 4, a PCS routing based on fault information is provided. In Section 5, a dynamic fault model for this PCS routing is introduced. In Section 6, we discuss some important features in 2-D meshes with dynamic faults. Section 7 shows simulation results. Section 8 concludes the paper and provides ideas for future research. Throughout the paper, proofs to theorems are shown in a seperate report.

2 Preliminaries

$K-$ary $n-$dimensional meshes. 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 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 a 2-D mesh is labeled as (x_u, y_u).

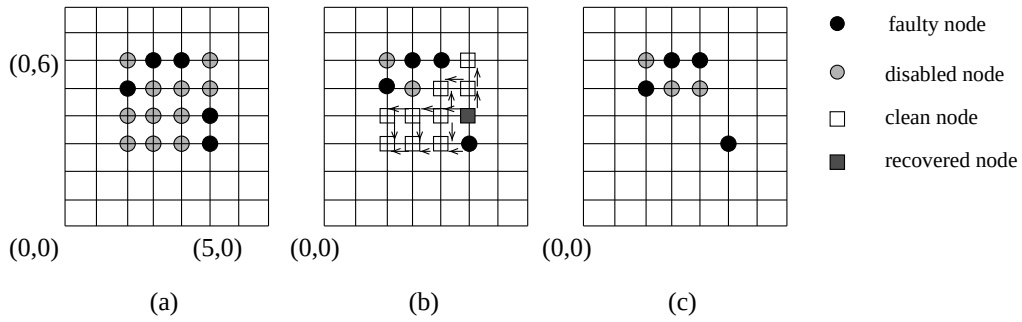

Figure 1: (a) A faulty block consisting of disabled and faulty nodes. (b) A clean process triggered by the recovery of (5,4). (c) Stabilized faulty blocks after the recovery.

The distance between two nodes s and d, $D(s,d)$, is equal to $\mid x_d - x_s \mid + \mid y_d - y_s \mid$. 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*.

Block fault model. Most existing literatures on fault-tolerant routing use disjoint rectangular blocks to model node faults and to avoid routing difficulties in meshes. A node-labeling scheme that identifies nodes is defined as follows:

Definition 1: *In a 2-D mesh, a non-faulty node is initially labeled enabled; however, its status is changed to disabled if there are two or more disabled or faulty neighbors in different dimensions.*

In a 2-D mesh, an enabled node is an *adjacent node* of a faulty block if it has one faulty or disabled neighbor in that faulty block. And its connecting direction to that faulty or disabled node is *blocked* direction of such an adjacent node. The blocked direction is used to identify an adjacent node for both two blocks. A *corner* of a faulty block is defined as an enabled node with two adjacent nodes of that faulty block in different dimensions; that is, it's connecting direction to one is the blocked direction of another. It is noted that an enabled node can be a corner for more than one (2, 3, and 4) faulty blocks. Connected disabled and faulty nodes form a faulty block. In Figure 1(a), five faults (2,5), (3,6), (4,6), (5,4), and (5,3) form a rectangle [2:5, 3:6]. In general, $[x_{min} : x_{max}, y_{min} : y_{max}]$ represents a rectangle with four corners: $(x_{min} - 1, y_{min} - 1)$, $(x_{min} - 1, y_{max} + 1)$, $(x_{max} + 1, y_{max} + 1)$, and $(x_{max} + 1, y_{min} - 1)$.

Extended safety level. The extended safety level [8] of a node in a given 2-D mesh is a 4-tuple: (E, S, W, N), where E stands for the distance from this node to the closet faulty block to its east. S,

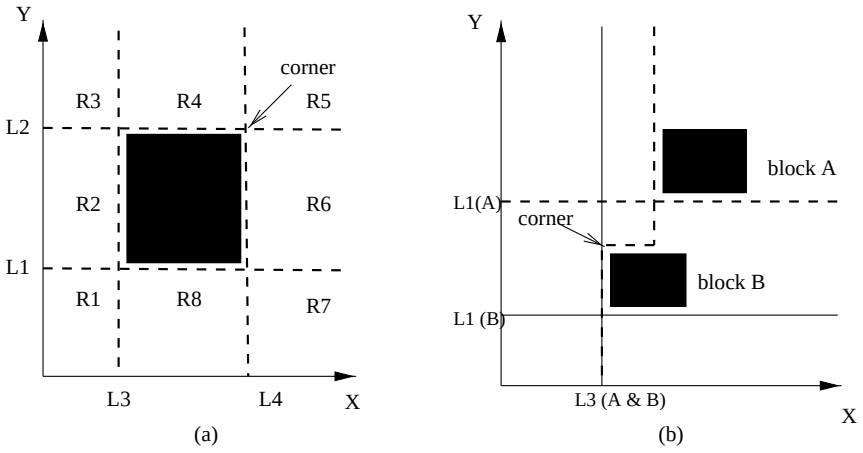

Figure 2: Boundaries of faulty blocks.

W, and N are defined in a similar way. To ensure a minimal path from source node, Wu [8] provided a safe node definition: *Assume that source node s:*$(0,0)$ *has an extended safety level* (E,S,W,N) *and destination node is* (x_d, y_d)*, with* $x_d, y_d \geq 0$*. The source node is safe (to the routing) if* $x_d \leq E$ *and* $y_d \leq N$*; otherwise, it is unsafe.* If a node is safe, a minimal path is guaranteed from source $(0,0)$ to (x_d, y_d) as long as no new fault occurs during the routing process.

Faulty-block-information used in minimal routing. A safe source of a routing message can ensure the existence of a minimal path. In [8], the notion of *region of minimal paths* (RMP) is introduced to include all the intermediate nodes of minimal paths for a given source and destination pair. That is, nodes and only nodes in this region are used to construct a minimal path. The task of routing process for a minimal path is to ensure that each forwarding node along the path is inside RMP. Once the boundary of RMP is known we can construct any minimal path to support fully adaptive routing.

In [8], Wu presented a *faulty-block-information* model and such information is associated with nodes in the adjacent lines of a faulty block. These lines (L_1, L_2, L_3, and L_4 in Figure 2(a)) are called *boundaries* of that faulty block. Based on these boundaries, RMP is easy to derive. Figure 2(b) shows an example of boundaries of multiple faulty blocks. The boundaries start from two corners (NE-corner $(x_{max}+1, y_{max}+1)$ and SW-corner $(x_{min}-1, y_{min}-1)$, or NW-corner $(x_{min}+1, y_{max}+1)$ and SE-corner $(x_{max}-1, y_{min}-1)$) of each faulty block and go forward along with each direction of X and Y dimensions. Without any other faulty block, the propagation of boundary information is forwarded node by node in each direction until it reaches an edge of the

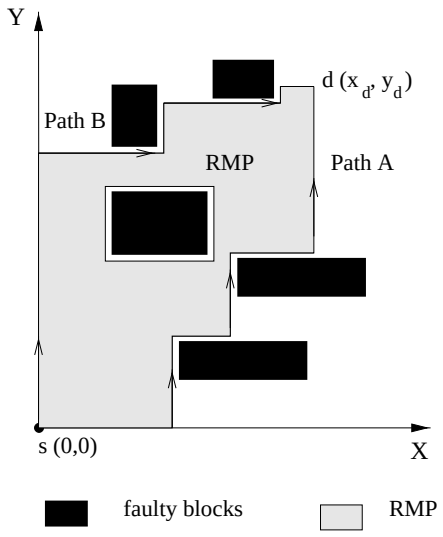

Figure 3: A sample RMP.

mesh. If a boundary L_i intersects with another faulty block, a turn is made towards L_i of the second faulty block. Another turn is made at the corner of the second faulty block to join L_i (see Figure 2(b) where L_3 of block A joins L_3 of block B).

For a message from source s: (0,0) to destination d: (x_d, y_d) with $x_d, y_d \geq 0$, a construction of RMP boundary from destination is provided in [8] and its reverse procedure from source is provided as following: If the source $(0, 0)$ is safe, RMP is enclosed by two paths, Path A and Path B (see Figure 3). Faulty blocks inside RMP are excluded. Starting from source $(0, 0)$, Path A is constructed by going east (positive X) until reaching the line $x = x_d$ and then by going north (positive Y) to reach destination (x_d, y_d). A turn from east to north is made if (a) the path hits a faulty block or (b) it crosses the lower section of boundary L_3 of a faulty block and the destination is in the area of R_4 which is one of regions divided by the boundaries of that faulty block (see Figure 2(a)). After that, a turn from north to east is made at the NW corner (x_{min}, y_{max}) of the faulty block and the process continues. Path B is constructed in a similar way. In a regular mesh without faults, the corresponding RMP is a rectangle $[0 : x_d, 0 : y_d]$. The RMP of a safe source s and a destination d in a 2-D mesh with multiple faulty blocks is shown in Figure 3.

The above construction of RMP boundaries can be used to prevent the routing message from moving out of RMP. In [8], Wu proposed a unicast algorithm based on the faulty block information for any routing from a safe source s:$(0, 0)$ to a destination d:(x_d, y_d) with $x_d, y_d \geq 0$. The safety status of the source is determined from the extended safety level associated with the source. The

routing starts from a safe source and uses any adaptive minimal routing until the boundary of any faulty block is met. If the selection of any preferred neighbor does not affect the minimal routing, the path is *non-critical*; otherwise, it is *critical*. In case of a critical path, one of preferred directions cannot be selected in a minimal routing due to the effect of faulty blocks. Such a direction is called *preferred but detour direction*. The selection should be done at current node u:(x_u, y_u) based on the relative location of the destination:

- (u is on the left section of L_1 of any faulty block): If the destination is in the area of R_6 divided by the boundaries of that faulty block (see in Figure 2(a)), the routing message should stay on L_1 until reaching the intersection of L_1 and L_4 (that is, positive X is a preferred direction and positive Y is a preferred but detour direction); otherwise, the next hop can be randomly selected.
- (u is on the lower section of L_3 of any faulty block): If the destination is in the area of R_4 divided by the boundaries of that faulty block the routing message should stay on L_3 until reaching the intersection of L_3 and L_2 (that is, positive Y is a preferred direction and positive X is a preferred but detour direction); otherwise, the next hop can be randomly selected.

For example, as shown in Figure 2(b), when the routing from $(0, 0)$ meets the lower section of L_3 of faulty block B, it also meets L_3 of faulty block A. If the destination is not in R_4 of A or R_4 of B, the routing is still non-critical and any of two preferred directions (positive X and positive Y) can be selected randomly; otherwise, the routing is critical and the message cannot be forwarded to positive X. In this case, positive X is the preferred but detour direction and positive Y is the only preferred direction that can be selected to construct a minimal path.

3 Faulty Block Information

In 2-D meshes, the shape of a faulty block may change by the occurrence of new faults or by the recovery of nodes from faulty status to healthy one. To identify the shape of new faulty blocks and propagate their block information along the boundaries, three procedures are used to exchange and update the related information among neighbors: *block construction*, *identification process*, and *boundary construction*. Here we use a *reactive approach* where information update at a node is done only when there are status changes among its neighbors.

First, a new labeling scheme is proposed as follows:

Definition 2: *In a 2-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 have known its clean status in the clean process, each clean node is labeled enabled. Each enabled node applies Definition 1 until there is no status change.

Specifically, a recovered node is set to *clean*. The change will be propagated to its disabled neighbors and contribute further changes. In Figure 1(b), node (5,4) is recovered from faulty status. First, (5,4) is labeled clean and it triggers the change of status in its disabled neighbors (4,4) and (5,5) to clean. The procedure continues until there is no more status change. The stabilized faulty blocks are shown in Figure 1(c). It is noted that the clean status of (4,4) will trigger the change of status in (4,5) and (3,4). When (3,5) knows the status changes of (4,5) and (3,4), it does not change its status to clean since it has two faulty neighbors in different dimensions. (4,5) changes to enabled once all its neighbors have known its clean status. In the next round, it has one faulty neighbor (4,6) and one disabled neighbor (3,5). And then, this new enabled node will change to disabled when Definition 1 is applied.

The new enabled/disabled labeling scheme can quickly identify those non-faulty nodes that may cause routing difficulty by labeling them disabled. 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 these affected nodes update their status. Connected disabled and faulty nodes form a faulty block. Any enabled node is also an adjacent node with blocked direction if and only if it has a faulty or disabled neighbor in that direction. An enabled node is also a corner if and only if it has two neighbors that its connecting direction to one is the blocked direction of another. Such a procedure is called *block construction*.

After the block construction incurred by some new faults, a faulty block may enlarge its size, and even a new faulty block may appear in the network. On the other hand, after the block construction incurred by some nodes recovered from faulty status, a faulty block may shrink its size or be divided into several small faulty blocks. If a corner of an old faulty block finds its existing condition cannot be satisfied, the existing boundaries of such a block is out of date. If such a corner is still an enabled node, a detection is propagated to all the nodes inside old block. Such propagation detects if there is any faulty node inside old block. Otherwise, the whole faulty block disappears and a deletion of its boundaries is activated at that corner. The deletion will propagate along those old boundaries. To identify a new faulty block, information of all the adjacent nodes of this block is identified by a procedure called *identification process*. For each new faulty block in the network that needs identification (if any), it has at least one new corner whose two neighbors are new adjacent nodes of this faulty block. The identification process is activated at such a new corner. Since the system

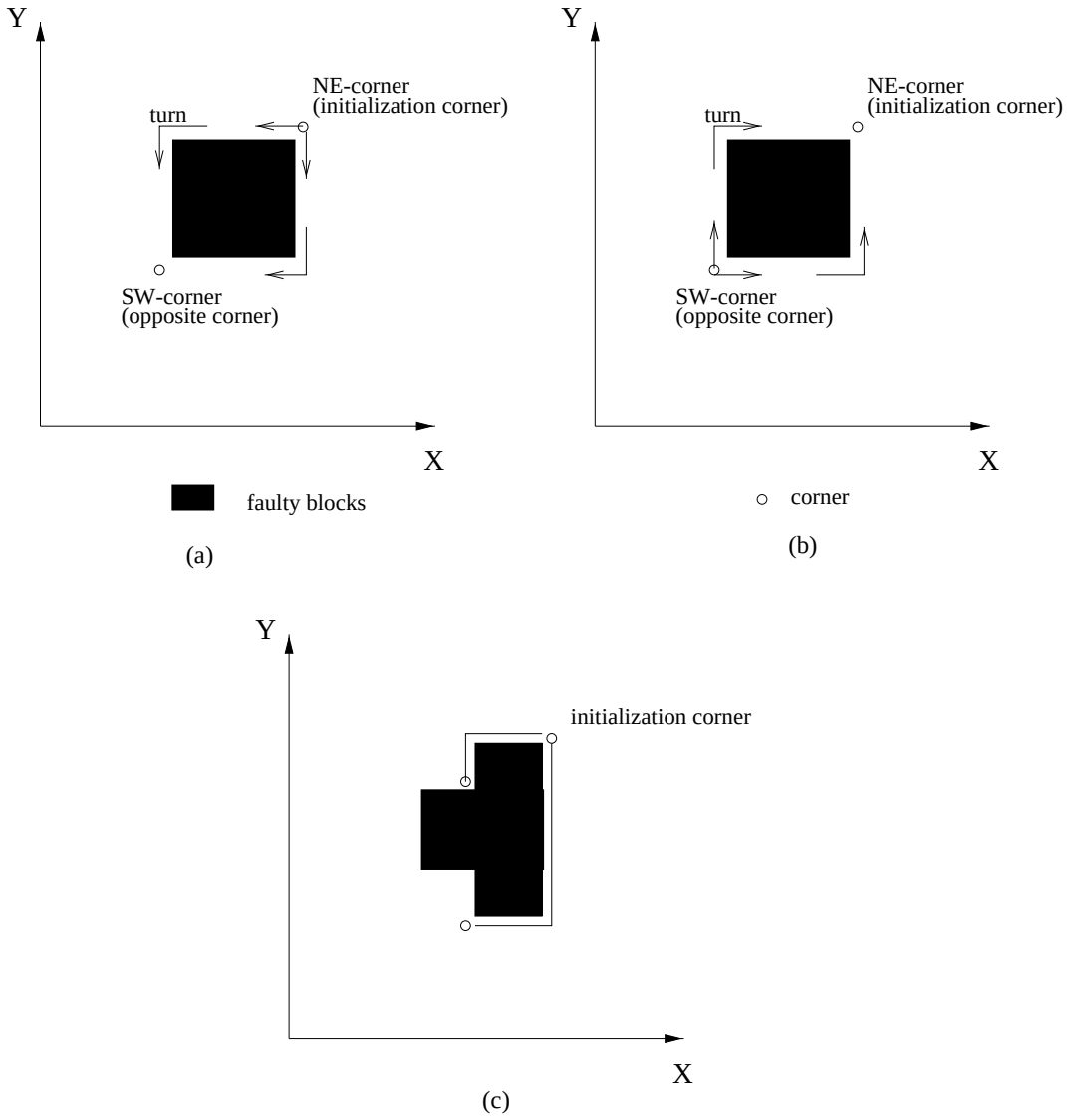

Figure 4: (a) Identification process activated at NE-corner and SW-corner. (b) Identified information re-sending. (c) Exceptions in identifying propagation.

Algorithm 1: Block construction and information distribution

1. Block construction by applying Definition 2.
2. Identification of adjacent nodes and corners.
3. Identification process: (a) Two identification messages (one clockwise and one counter-clockwise) are sent along the enabled nodes adjacent to the new block from a new corner, until they reach the opposite corner. (b) Partial block information from the initialization corner is transferred to form faulty block information at the opposite corner. (c) The faulty block information is sent along the adjacent nodes back to the initialization corner by these two messages.
4. A boundary construction is activated at initialization corner and its opposite corner that receives consistent faulty block information.

is distributed and dynamic, no corner knows if the block construction is completed. Thus, this procedure starts whenever a *new* corner is formed. For each initialization corner, two identification messages (one clockwise and one counter-clockwise) are initiated (see Figure 4(a)). Each message initiated at that corner C:(x_c, y_c) carries partial block information: (x_c, y_c) and the blocked direction of the neighbor it will be sent to. First, they will be sent to those two neighbors adjacent to the new block. Such propagation will continue until the message traverses all the enabled nodes adjacent to the new faulty block. The clockwise and counter-clockwise messages from one corner will reach the opposite corner of that initialization corner. With the position information of pair of corners (the initialization corner and its opposite corner), the new faulty block is identified and the faulty block information $[x_{min} : x_{max}, y_{min} : y_{max}]$ is formed at that meeting corners. After that, these two messages will continue their propagation (see Figure 4(b)) and carry this identified faulty block information back to the initialization corner.

To guide the routing process, the faulty block information is transferred along the boundary of the new faulty block from the initialization corner and its opposite corner when they get the identified information (see in Figure 2). In our reactive model, if any corner has already had the new faulty block information, there is no need to start a 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.

Note that each identification message (clockwise/counter-clockwise) sent from its initialization corner is expected to make one and only one *specific turn* (90° right/left turn) before reaching the opposite corner. If there is a faulty or disabled neighbor in the forwarding direction, it is ensured that the new block is not stable. The message is discarded at current node to avoid generating incorrect faulty block information. If only one message from the initialization corner is received at the opposite corner, the other is discarded in the propagation procedure or has a wrong turn. That is, the shape of block may not be the exact rectangle indicated by the positions of the initialization corner and its opposite corner. Normally, a TTL (time-to-live) is associated with each identification message and the corresponding message will be discarded once the time expires (see Figure 4(c)). After these two messages from the same initialization corner meet at the opposite corner, the propagation continues. But this time, the stable block ensures that they can go back to their initialization corner.

4 Faulty-block-information-based Routing

The PCS routing in [2] needs a detour when its preferred neighbors are all faulty and needs a backtracking when all its outward directions have been tried or blocked by faulty neighbors. The routing based on fault information is an adaptive routing in 2-D meshes. Like a regular minimal routing, at each step it tries to forward the message to a preferred neighbor. The difference is that the selected preferred neighbor can ensure the remaining path is minimal in the fault-information-based routing (if there is no occurrence of a new fault).

Algorithm 2 shows a fault-information-based PCS routing from s:(x_s, y_s) to d:(x_d, y_d) in a 2-D mesh with dynamic faults. For the current node $u(x_u, y_u)$ $(\neq d)$ with one incoming direction and three possible outgoing directions, the routing selects one of directions as the forwarding direction in the priority order of preferred, spare, preferred but detour, and incoming directions. Normally, we have the following cases for intermediate node u: (a) If $x_u = x_d$ or $y_u = y_d$, there is one preferred direction and two spare directions. (b) If $x_u \neq x_d$ and $y_u \neq y_d$, there are two preferred directions and one spare direction. (c) At a boundary line, if it is critical, one preferred direction changes to preferred but detour direction. If it is not critical, there is no preferred but detour direction.

It is noted that each forwarding direction at a participant node cannot be used again. Thus, each routing header in our PCS routing includes destination address and a list of used-directions for each forwarding node along the path. This is because that the system is dynamic and the priority of directions may also change.

Algorithm 2: 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.

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

Proof: When nodes are recovered from a block, the old block is shrunk in each direction (+X, -X, +Y, and -Y) if it still exists. Suppose that the block is shrunk k hops in +X direction. According to our procedure of block construction and information distribution, the deletion of old boundary L_4 will be activated from a certain node after the propagation of new boundary L_4 reaches it. Assume that a routing meets the lower section of old L_4 and its destination is inside R_4 which is one region divided by old boundaries. If the routing can access R_8 which is also one region divided by old boundaries because the old L_4 is deleted, it will meet the constructed lower section of new L_4. The detour can also be avoided. □

Theorem 1 ensures the effectiveness of our fault information model when faults are recovered in the networks.

5 Dynamic Fault Model

In general, it is impossible to design an optimal fault-tolerant routing algorithm when node failure and recovery can occur dynamically at any time. Under such general dynamic failure assumptions, it is not even possible to guarantee an upper bound on the routing distance since no algorithm can predict when and where faults will occur. It is therefore necessary to impose some failure conditions and restrictions.

We assume there are at most F faulty nodes in a 2-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 the fault information updating in

F	number of faults in a given $m \times m$ mesh and $F \leq 2 * m$
f_i	i^{th} fault occurrence where $i \in \{1, 2, ..., F\}$
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 source to destination
$D(i)$	distance from the current node to the destination at time t_i
a_i	total rounds the stabilizing block construction for f_i converges
a_{max}	max$\{a_i\}$
e_{max}	maximum of all length or width of fault blocks
b_i	total rounds of stabilizing identification process for f_i converges
c_i	total rounds of stabilizing boundary construction for f_i converges
λ	number of rounds of fault block construction and information distribution (block construction, identification process, and boundary construction) at each step

Table 1: List of notation used in the discussion

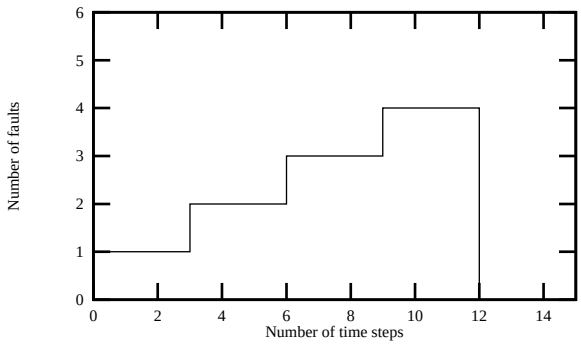

Figure 5: Distribution of fault occurrence when d_i's are uniform ($d_i = 3$).

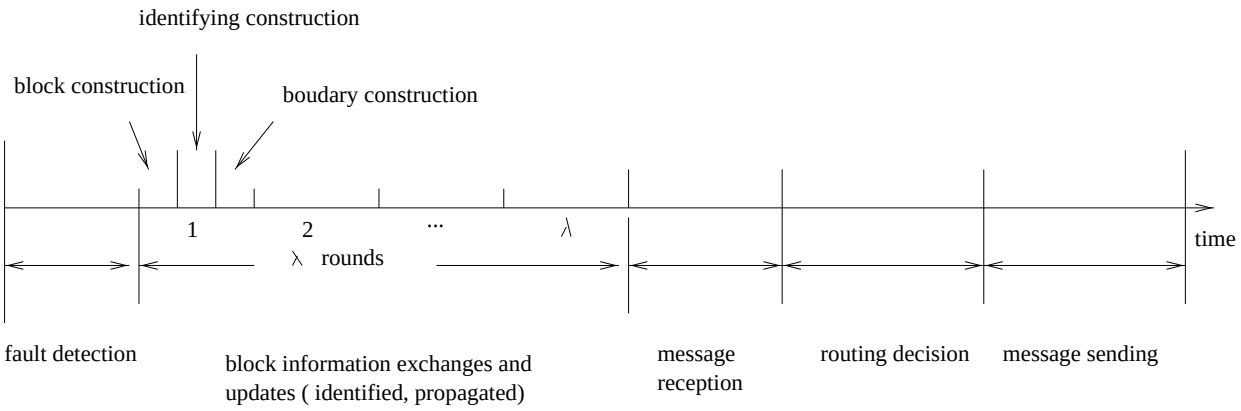

Figure 6: Actions within a step.

the mesh is already stabilized before the occurrence of the next fault and there is no fault that occurs at the edge of mesh. Based on the properties discussed in [8], there is no disconnected area in such a mesh. It means that there is always a path between the source and the destination. Before a routing message is initiated at time t, it is assumed that the first p fault occurrences have already occurred; that is, $p = \max\{l | t_l \leq t\}$. $D(i)$ represents the distance from the current node (u) to the destination (d) at time t_i when f_i occurs ($1 \leq i \leq F$) and D represents the distance from source to the destination. Before the start time t, the routing message is at its source and $D(i) = D = \mid x_s - x_d \mid + \mid y_s - y_d \mid (i \leq m)$. 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 change at t_i are already stabilized. When the intervals, d_i's, are uniform, i.e., $d_i = c$, the distributions of fault occurrence is shown in Figure 5. To simplify the discussion, Table 1 summaries the notation used in this paper.

We adopt the following model for activities in a node. At each step, every node in $m \times m$ mesh starts with fault detection of adjacent links and nodes, and then collects and distributes three kinds of fault information: block information, identifying information, and boundary information through λ rounds of exchanges and updates. The disabled/enabled status propagation, identifying message, block information updating 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 6. Thus, for the i^{th} fault occurrence, the block construction will be stabilized in $\lceil \frac{a_i}{\lambda} \rceil$ steps, the identifying construction 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.

To simplify the discussion, it is assumed that any adjacent faults, links and nodes, are detected at fault detection phase (any faults occur 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 6. 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.

6 Detour Analysis

Based on the definition of safe source, its PCS routing does not need any detour if there is no new fault. If a new fault occurs, before the new information distribution stabilized, 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 needed to re-enter RMP 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 2-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 2: *For any fault-information-based routing from a safe source s to an enabled destination d (not in any faulty block), if* $D(i+1) > 0$*:*

$$\begin{cases} D(i) & = D & i \leq p \\ D(i+1) & \leq D - (d_i - t + t_p - 2 * a_i - 2 * e_{max}) & p = i \\ D(i+1) & \leq D(i) - (d_i - 2 * a_i - 2 * e_{max}) & p < i < F \end{cases}$$

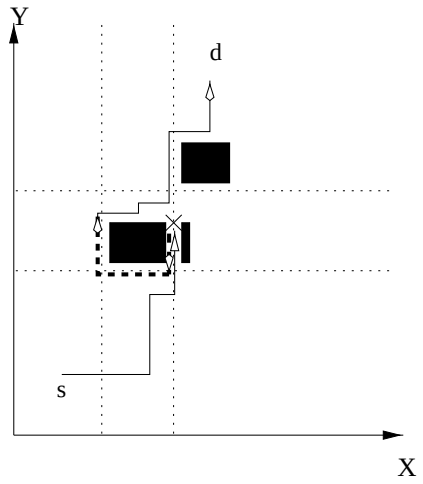

Figure 7: Maximum detours by a new fault.

Proof: Based on the definition of safe source, the routing has an optimal path before it starts, and the source and the destination are both in the RMP. If there is no new fault, the routing will reach the destination guided by the information on the boundary lines. The routing only detours by the fault occurs after time t_p. Since there is only one new faulty block in each interval, the routing message will re-enter the RMP. $D(i) = D$ $(i \leq p)$ is obvious by its definition. Considering the start time t at the first interval d_p, if $t = t_p$ and $D(p+1) > 0$, the routing message advances d_p steps since time t_p, i.e., $D(p+1) = D - d_p$; otherwise, the routing message advances $d_p - t + t_p$ steps during interval d_p, i.e., $D(p+1) = D - (d_p - t + t_p)$. So $D(i+1) = D - (d_i - t + t_p)$ $(i = p \geq F)$ and $D(i+1) = D(i) - d_i$ $(i \geq F)$ unless it reaches the destination d. When a new fault f_i occurs, the worst case is that it conjoins two small blocks (k_1 and k_2) into a big one (k). After the routing message goes outside of the block following the block construction, the big block (k) will still incur e_{max} detours. Totally, the routing message has at most $2 * a_i + 2 * e_{max}$ extra steps in each interval d_i $(p \leq i < F)$. During the first $d_p - t + t_p$ steps in interval d_p, the routing message starts from s and reaches a node $D(p+1)$ finally. Since the routing message goes one node further along with its path at each step and there is no more $2 * a_i + 2 * e_{max}$ extra steps, it advances at least $d_p - t + t_p - 2 * a_i - 2 * e_{max}$ steps closer to its destination. □

Assume that $m + q - 1$ is the largest index for fault such that $D(m + q - 1) > 0$. That is, the f_{m+q-1} is the last fault occurrence that could affect the routing process. Actually, that q is

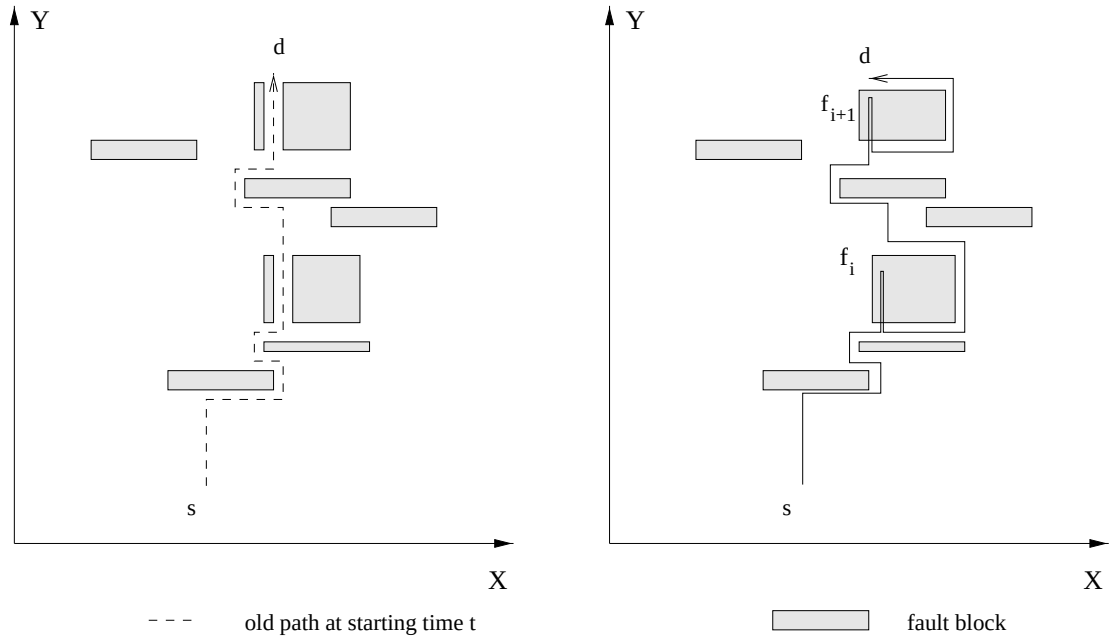

Figure 8: Maximum detours for a routing with unsafe source.

the maximum number of intervals in which the routing message detours at least once. Based on Thereom 2, 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 an optimal path before it starts:

- The routing message will get closer to the destination between two occurances of consecutive faults as long as these two faults are separated by more than $\max\{\frac{a_i+b_i+c_i}{\lambda}\}$ time steps.
- When d_i's are uniform, i.e., $d_i = c$, $q = \max\{l | (l-1)(d_i - 2 \times (a_i + e_{max})) < D + t - t_p \leq D + c\}$. Then $q \leq \lfloor \frac{D+c}{c-2\times(a_{max}+e_{max})} \rfloor$.
- Maximum detour for a message with an optimal path before it starts is $(a_{max} + e_{max}) \times \lfloor \frac{D+c}{c-2\times(a_{max}+e_{max})} \rfloor$.

Theorem 2 shows the maximum detours in each interval for a routing message from a safe source. Based on this, we get upper bound of maximum detours for such a routing message. A routing from an unsafe source needs more detours to enter RMP from its destination. The below

discussion extends the above result for any routing message including that from an unsafe source (see in Figure 8).

Theorem 3: *For a routing message from an enabled source s to an enabled destination d in a $m \times m$ 2-D mesh, if there is a routing path with length P at start time t, the routing process will end in the following k intervals and*

$$k \leq \max\{l | P + t - t_p - \sum_{i=p}^{p+l-2} (d_i - 2 * a_i - 2 * e_{max}) > 0\}.$$

The probability of maximum detour in a routing is no more than

$$\prod_{i=p}^{p+k-1} (\frac{1}{m^2}).$$

Proof: After the routing process starts at t, each step, the routing message advances one hop along the path until the path is disconnected by a new block. At most $a_i + e_{max}$ detours or backtracking are needed in interval d_i to let the routing message go back to the old routing path. Therefore, there are at most $2 * (a_i + e_{max})$ extra steps during interval d_i. If a routing process passes through $k-1$ intervals, the routing message at least advanced $\sum_{i=p}^{p+l-2}(d_i - 2 * a_i - 2 * e_{max})$ steps along the path. Suppose that f_{k+p-2} i s the last fault may affect the routing. At t_{k+p-2}, $P + t - t_p - \sum_{i=p}^{p+k-2}(d_i - 2 * a_i - 2 * e_{max}) > 0$. Therefore, the routing process will end in the following k intervals from time t and $k \leq \max\{l | P + t - t_p - \sum_{i=p}^{p+l-2}(d_i - 2 * a_i - 2 * e_{max}) > 0\}$.

For each interval, the worst case is that a new fault (f_i) can conjoin two blocks (k_1 and k_2) into a big block k. To block the routing message, the new fault should be on the path of routing. When the new fault occurs, the routing message should arrive at its neighbor, otherwise, the new disabled node caused by the block construction will make the routing message backtrack; that is, the total detours will be less than $a_i + e_{max}$. The possibility of maximum detours in each interval is no more than $\frac{1}{m^2}$. The probability of maximum total detour is that of all these happened. □

7 Simulation

A simulation has been conducted on a 100×100 mesh to test the estimated maximum detours using the PCS routing based on fault information. Comparing with the experimental result of PCS routing without any fault information, the scalability of PCS routing based on fault information is concluded in dynamic 2-D meshes.

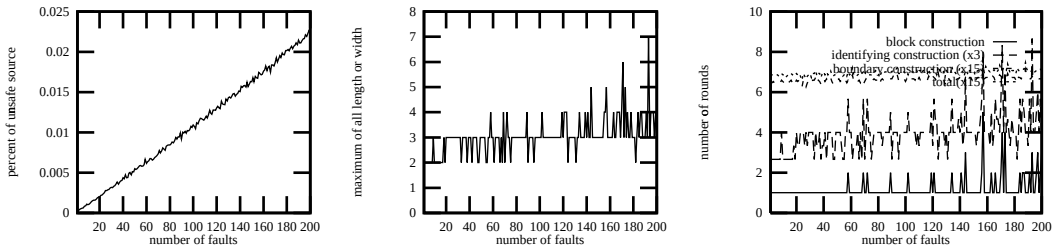

Figure 9: a_i, b_i, c_i, e_i, and $a_i + b_i + c_i$ in a 100×100 mesh with at most 200 faults (intervals).

We randomly generate faults, source and destination. Figure 9(a) shows only few source nodes are unsafe. It means that almost all the routings follow the case discussed in Theorem 2. Moreover, if the destination is inside any block, the routing will be interrupted as it reaches the boundary of this block or has tried all the nodes around the block. Under the control of upper protocols, the routing will re-try to reach the destination after the recovery of some faults.

Figure 9(b) shows that length of edge of changing block at each interval. From experimental result, $e_{max} = 7$. Figure 9(c) shows the rounds needed in for a new block construction and its information distribution. From experimental result, $a_{max} = 7$ and $\max\{a_i + b_i + c_i\} = 110$.

Table 2 shows the maximum detours of the routing with fault information at different information distribution speed (λ =200, 110, 30, 7, and 1) in a 100×100 2D mesh with up to 200 dynamic faults when $d_i = 110$. (S) is for the routing from safe source, (G) is for the routing from enabled source and (W) is for the routing without fault information.

We make the following observations from the comparison shown in Table 2.

- The analysis detours of routing algorithm using fault information are effective as an upper bound.
- The experimental results of PCS routing using fault information are lower than those of analysis results. It is because that the worst fault configuration is difficult to occur when the faulty nodes and pair of source and destination are all randomly generated. Even the worst case occur, based on Theorem 3, the probability of maximum detour is very small.
- Based on different requirement, we can adopt different information distribution speed. If we need the least retrying messages or interrupted messages, we need $\lambda \geq \max\{a_i + b_i + c_i\}$. If we can accept several more detours, the best choice is to select $\lambda \geq \max\{a_i\}$. It does not need a fast information distribution. If there is no extra support for the information propagation,

λ	maximum detours					average detours /message		
	analysis results		experiment results					
	(S)	(G)	(S)	(G)	(W)	(S)	(G)	(W)
200	42	177	24	48	184	0.0066	0.0067	0.0508
110	42	177	24	48	168	0.0068	0.0069	0.0593
30	42	177	32	78	175	0.0132	0.0134	0.0520
7	42	177	32	78	173	0.0136	0.0139	0.0511
1	42	177	42	85	158	0.0202	0.0231	0.0604

Table 2: Comparison of maximum detours among routing from a safe source (S) (or from an unsafe source (G)) with fault information and routing without fault information (W).

the propagation is processed as routing message ($\lambda = 1$). Although it is the worst case of all, the scalable routing using fault information is still achieved based on the detours analysis.

In summary, the experimental results of routing algorithm using fault information are close to those analysis results. Comparing the number of detours in this fault-information-based PCS routing with that in the PCS routing without fault information, the scalability in our PCS routing improves significantly even when the routing starts from an unsafe source. As an upper bound of maximum detours, the analysis results are effective. The scalability of routing algorithm is achieved by such an upper bound. And the upper bound of maximum detours in our routing process does not increase much when the faults in the cubes increase.

8 Conclusions

We have studied an upper bound of maximum detours in a 2D mesh with dynamic faults using fault-tolerant routing algorithm based on limited global information. The concept of fault information associated with each node at the boundary lines of faulty blocks has been used to represent limited global information. Our study shows that there is an upper bound of maximum detours. Simulation results show the effectiveness of such an upper bound and the scalability of routing algorithm based

on fault information. Applying this approach to other fault models are interesting problems for future research.

References

[1] S Dutt and J. P. Hayes. Some practical issues in the design of fault-tolerant multiprocessors. *IEEE Trans. on Computers.* May 1992, 588-598.

[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] INTEL. *A Touchstone DELTA System Description.* Intel Corp., Santa Clara, CA, 1990.

[4] INTEL. *Paragon XP/S Product Overview.* Intel Corp., Santa Clara, CA, 1991.

[5] S. L. Lillevik. The touchstone 30 gigaflop delta prototype. *Proc. of the 6th Distributed Memory Computing Conference*, pages 671–677, 1991.

[6] C. L. Seitz. The architecture and programming of the ametek series 2010 multi computer. *Proc. of the 3rd Conference on Hyperucbe Concurrent Computers an d Applications*, pages 176–182, 1988.

[7] 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.

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