A Fault-tolerant Adaptive and Minimal Routing Scheme in n-D Meshes

JIE WU

Department of Computer Science and Engineering, Florida Atlantic University, Boca Raton, FL 33431, USA
Email: jie@cse.fau.edu

In this paper a sufficient condition is given for minimal routing in n-dimensional (n-D) meshes with faulty nodes contained in a set of disjoint fault regions. It is based on an early work of the author on minimal routing in low-dimensional meshes (such as 2-D meshes with faulty blocks). Unlike many traditional models that assume all the nodes know global fault distribution, our approach is based on the concept of *limited global fault information*. First, a novel fault model called *Fault Region* is proposed in which all faulty nodes in the system are contained in a set of disjoint regions. Fault information is coded in a $2n$-tuple called *Extended Safety Level* associated with each node of an n-D mesh to support minimal routing. Specifically, we study the existence of minimal paths at a given source node, limited distribution of fault information, minimal routing and deadlock-free routing. Our results show that any minimal routing that is partially adaptive can still be applied, as long as the destination node meets a certain safety condition. A dynamic planar-adaptive routing scheme is presented that offers better fault tolerance and adaptivity than the regular planar-adaptive routing scheme in n-D meshes. In 3-D meshes, both regular and dynamic planar-adaptive routing need three virtual channels. Our approach is the first attempt to address minimal routing in n-D meshes with faulty nodes using limited global fault information.

Received 15 March 2001; revised 18 December 2001

1. INTRODUCTION

In a multicomputer system, a collection of processors (or nodes) work together to solve large application problems. These nodes communicate data and coordinate their efforts by sending and receiving packets through the underlying communication network. Thus, the performance of such a multicomputer system depends on the end-to-end cost of communication mechanisms. The routing time of packets is one of the key factors that are critical to the performance of multicomputers. Basically, routing is the process of transmitting data from one node called the *source* node to another node called the *destination* node in a given system. The *mesh-connected topology* [1, 2] is one of the most thoroughly investigated network topologies for multicomputer systems. Mesh-connected topologies, also called k-ary n-dimensional (n-D) meshes, have an n-D grid structure with k nodes in each dimension such that every node is connected to two other nodes in each dimension by a direct link. Mesh-connected topologies include n-D meshes, tori and hypercubes. Examples of commercial products based on n-D hypercubes include the Ncube's nCUBE, the Thinking Machine's Connection Machine, which is a hypercube interconnected bit-serial SIMD machine and, more recently, the SGI's Origin 2000 [3]. Many multicomputers that use 2-D meshes include the MIT J-machine [1], the Symult 2010 [4] and the Intel Touchstone [5]. The CRAY T3D and T3E [2] systems use a 3-D torus.

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 [6, 7, 8]. However, adding and deleting nodes and/or links require modifications 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.

An important and challenging issue is to extend communication subsystems which include various routing algorithms to cope with faulty components. To this end, fault models and routing algorithms are the two keys to successfully extending the existing approaches. We introduce a convex type of fault region as our fault model and propose a novel *information model* in which each node in a mesh-connected multicomputer collects and distributes fault information concurrently but in a

decentralized way. This collection and distribution process exhibits the desirable property of self-stabilization. In addition, this process converges quickly to meet the demands of many real-time applications. To ensure that this approach is scalable for a large and complex network, only specially coded fault information is distributed rather than detailed information. Unlike many existing information models that require each node to have knowledge of the entire network, the coded fault information associated with each node represents limited global information by exploring the locality of disturbances in the network. It also reduces the memory requirement [9] to store fault information at each node. When a disturbance occurs, only those nodes affected update their information to keep it consistent.

The *safety-level-based* (or *safety-vector-based*) routing [10, 11], a special form of *limited-global-information-based* routing, is a compromise between local-information- and global-information-based approaches. In this type of routing, a routing function is defined based on current node, destination node and limited global fault information gathered at the current node. This approach differs from many existing ones where information is brought by the *header* of the routing packet [12] and the routing function is defined based on the header information and the local state of the current node [13]. In this approach, neighborhood fault information is captured by an integer (safety level) or a binary vector (safety vector) associated with each node. For example, in a binary hypercube, if a node's safety level is m (an integer), there is at least one Hamming distance (or minimal) path from this node to any node within Hamming distance m [11]. Using the safety level (or safety vector) associated with each node, a routing algorithm can obtain an optimal or suboptimal solution and it requires a relatively simple process to collect and maintain fault information in the neighborhood. Therefore, limited-global-information-based routing can be more cost effective than routing based on global or local information. The safety-level-based routing has been successfully applied to binary hypercubes, but is less efficient when it is directly applied to mesh topologies such as 2-D and 3-D meshes. In [14], the author introduced the concept of the *extended safety level* with its use in achieving minimal routing in 2-D meshes with faulty nodes contained in a set of faulty blocks.

In this paper, the extended safety level concept is further extended for general n-D meshes. The challenge is to find a minimal path in an n-D mesh with faulty nodes contained within a set of disjoint *fault regions* (a fault model extended from the commonly used faulty block model in 2-D meshes). In 3-D meshes fault regions are called *faulty cubes*. The amount of limited global information should be kept to a minimum and should be easy to obtain and maintain. The requirement to be deadlock-free and livelock-free adds another challenging dimension. A deadlock occurs when some packets from different packets cannot advance towards their destinations because the channels requested by them are not available. A livelock occurs when a routing packet travels around its destination node, but never reaches it. Designing a routing protocol that is both fault-tolerant and deadlock-free (and livelock-free) poses a major challenge. Fault tolerance demands an adaptive and flexible routing process to get around faults. On the other hand, added flexibility increases the chances of deadlock. In addition, some deadlock and livelock situations are fault-induced [15]. Our approach is to provide just enough adaptivity in the routing process to ensure fault tolerance so that the cost of preventing deadlock and livelock (using the virtual channel approach) is minimized.

Specifically, we address the issues of the existence of a minimal path at a given source node, limited distribution of fault information, minimal routing and deadlock-free routing. The concept of partial adaptive routing is defined and a dynamic planar-adaptive routing approach is proposed that trades routing adaptivity for a simple deadlock-free routing with a better fault tolerance capability than Chien and Kim's regular planar-adaptive routing [16]. Our approach is the first attempt to address the minimal routing in n-D meshes with faulty nodes using limited fault information. Our main results include the following.

- A fault model called Fault Region is introduced in n-D meshes. A simple labeling scheme is introduced that quickly identifies those non-faulty nodes that cause routing difficulty and disables them. A fault region consists of adjacent faulty and disabled nodes. The labeling scheme produces a set of disjoint fault regions that contain all faulty nodes.
- A new limited global information model called Extended Safety Level is proposed, which is coded fault information represented by a $2n$-tuple associated with each node. The safety level information can be used to determine the existence of a minimal path for a given pair of source and destination nodes in n-D meshes.
- The concepts of fully and partially adaptive routing in n-D meshes are formally defined. It is shown that the planar-adaptive routing [17] fails to meet the proposed partial adaptivity requirement.
- A dynamic planar-adaptive routing is proposed and it is used to prove that any minimal routing that is partially adaptive can be applied in our model as long as the destination node meets a certain safety requirement.
- A simple deadlock-free implementation of dynamic planar-adaptive routing is presented using n ($n+1$ when n is even) virtual channels in an n-D mesh.

The selection of a switching method is not covered in this paper. Basically, the proposed approach can potentially be used for all methods: packet switching, circuit switching, wormhole switching [18], virtual cut-through [19] and pipelined circuit switching [20].

The collision-free routing in the presence of obstacles is also studied in other fields such as routing urban vehicles, motion planning in robotics and wire routing in VLSI. The foci in these fields are different. For example, given a set of obstacles and two points in the plane, most studies try to find a shortest path, not necessarily a minimal one, among all the available collision-free paths. In addition,

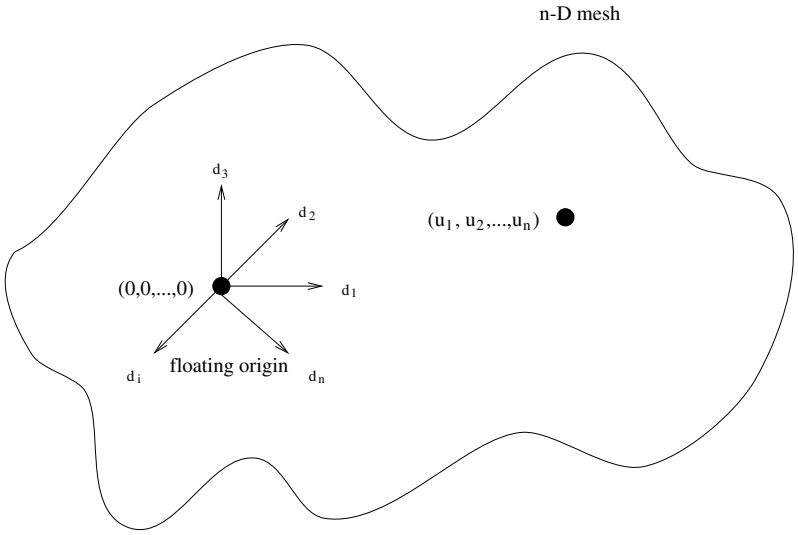


FIGURE 1. An n-D mesh.

most problems are optimization problems associated with a certain optimization function, such as the minimum number of bends as in VLSI routing. See [21] for a survey of research in these fields.

This paper is organized as follows. Section 2 presents some preliminaries. Section 3 proposes the Fault Region model in n-D meshes. Section 4 introduces the concept of Extended Safety Level as a special form of limited global information. Section 5 offers a simple adaptive and minimal routing algorithm based on limited global information provided in n-D meshes. It is shown that any partially-adaptive and minimal routing algorithm can be applied in our model, as long as the destination meets a certain safety requirement. Section 6 discusses possible extensions, including a strengthened sufficient condition, the application of the proposed approach in an n-D torus and deadlock-free and livelock-free routing. Section 7 concludes the paper. The proofs of all the theorems are listed in the Appendix.

2. NOTATION AND PRELIMINARIES

A k-ary n-D mesh with k^n nodes has an interior node degree of $2n$ and a network diameter of $n(k-1)$. Each node u has an address $(u_1, u_2, \ldots, u_n)$, where $u_i \in \{0, 1, \ldots, k-1\}$ corresponds to the location of u in dimension d_i. Two nodes v, $(v_1, v_2, \ldots, v_n)$, and u, $(u_1, u_2, \ldots, u_n)$, are connected if their addresses differ in one and only one element (dimension), say dimension d_i; moreover, $|u_i - v_i| = 1$. Basically, nodes along each dimension are connected as a linear array. In the subsequent discussion, a k-ary n-D mesh is simply denoted as an n-D mesh without specifying its size.

A coordinate system in an n-D mesh is defined with its 'floating' origin in node $(0, 0, \ldots, 0)$ adjustable by constant c (see Figure 1). A d_i-dimensional axis (denoted as d_i-axis) consists of nodes with addresses $(0, 0, \ldots, 0, u_i, 0, \ldots, 0)$, where u_i is an integer. Because of the floating origin of the mesh, we can assume without loss of generality that the source in a routing is $(0, 0, \ldots, 0)$ and the destination is $(u_1, u_2, \ldots, u_n)$. Along each dimension d_i, there are two directions: positive d_i (also d_i+) and negative d_i (d_i-). Along the positive d_i (negative d_i) dimension, the coordinate in dimension d_i increases (decreases). Clearly, there are $2n$ directions in an n-D mesh.

Routing is a process of sending a packet from a source to a destination. A routing is *minimal* if the length of the routing path from source $(0, 0, \ldots, 0)$ to destination $(u_1, u_2, \ldots, u_n)$ is the distance between these two nodes, i.e. $\sum_{i=1}^{n} |u_i|$. Throughout this paper, we focus on minimal routing in an n-D mesh with faulty nodes. The challenge is to find a minimal path (if there exists one) by avoiding faults in the system.

The simplest routing algorithms are *deterministic* which define a single path between the source and destination nodes. The routing following the dimension order is an example of deterministic routing in which the packet is first forwarded along dimension d_1 and is then routed along dimension d_2, and so on. Finally the routing packet is forwarded along dimension d_n. *Adaptive* routing algorithms, on the other hand, support multiple paths between the source and destination nodes. *Fully adaptive and minimal* routing algorithms allow all packets to use any minimal paths. A *preferred direction* is one along which the neighbor is closer to the destination. In an n-D mesh, there are at most n preferred directions, out of $2n$ possible directions, for a routing process. Actually, the number of preferred directions is equal to the number of dimensions spanned by the source and destination pair. For example, suppose in a routing in a 3-D mesh that the source is $(2, -2, -4)$ and the destination is $(1, 2, -3)$, then preferred dimensions at the source are west (negative d_1), north (positive d_2) and front (positive d_3). During a minimal routing, the number of preferred directions from an intermediate node to the

destination reduces and it eventually becomes zero upon reaching the destination.

DEFINITION 1. *A minimal routing is fully adaptive if it can select any preferred direction at any step of the routing process. A minimal routing is partially adaptive if it can select from at least two preferred directions at any step whenever there are two or more preferred directions.*

3. FAULT REGION

Before proposing a minimal routing algorithm in an n-D mesh with faulty components, we first discuss the fault model under consideration. This paper considers node faults only. To simplify the routing process, a labeling scheme is introduced to quickly identify those non-faulty nodes that cause routing difficulty and disable them. As a result, a set of convex-type *fault regions* is formed.

DEFINITION 2. *In an n-D mesh, a non-faulty node is either marked enabled or disabled. Initially, all non-faulty nodes are marked enabled. A non-faulty node is marked disabled if there are two or more disabled or faulty neighbors along different dimensions. A fault region contains all the connected disabled and faulty nodes.*

Based on Definition 2, there are three types of nodes: faulty nodes, enabled nodes and disabled nodes. The node status can be easily determined through rounds of status exchanges among neighboring nodes. The number of rounds of information exchanges is dependent on the maximum size of the fault region. Rounding information can be implemented asynchronously and independently at each node. That is, each node updates its status only when there is a status change of a neighbor. It is assumed that both source and destination nodes in a routing process are non-faulty and they are marked enabled. A fault region has the following desirable features that facilitate simple and minimal routing.

THEOREM 1. *In an n-D mesh, a fault region defined by Definition 2 has the following properties:*

(1) *every neighbor of a fault region has one and only one faulty or disabled neighbor (in the fault region);*
(2) *the distance between any two fault regions is at least two.*

With the first property of fault region (also called the convex feature), the address of a fault region can be simply described by a range along each dimension, e.g. f_i: f_i', with $f_i \leq f_i'$, specifying the range along dimension d_i. A general fault region can be represented by [f_1: f_1', f_2: f_2', ..., f_n: f_n'] which covers $\prod_{i=1}^{n}(f_i' - f_i + 1)$ nodes. When the range along a dimension is one, say $f_i : f_i$, the number of dimensions spanned by the corresponding region is reduced by one.

For n-D meshes with boundary, we can add 'ghost' nodes around boundary nodes to change these nodes to regular interior nodes. Ghost nodes are assumed to be non-faulty and are marked enabled. The ghost nodes are 'imaginary' nodes. They do not exist. With such imaginary nodes, boundary nodes become interior nodes. In this case, we can use a uniform procedure to calculate and update node status. Note that in some models, such as the one proposed by Chien and Kim [17], a pessimistic fault model is used. Specifically, one of the ghost nodes has to be marked disabled, hence generating more disabled nodes in the given mesh.

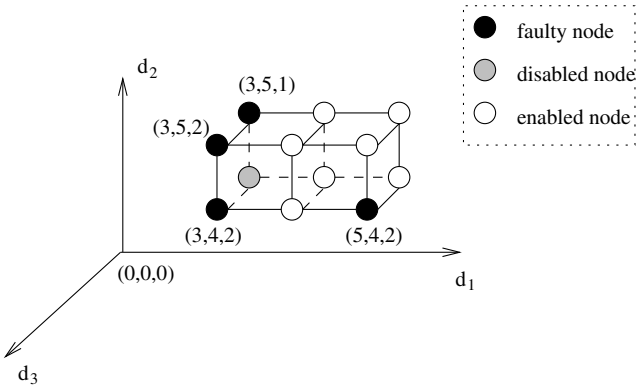


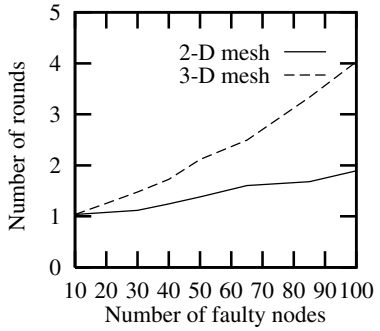
FIGURE 2. Fault regions in a 3-D mesh.

The concept of faulty cube in 3-D meshes stems from the faulty block model [16, 22, 23, 24, 25] in 2-D meshes. In 3-D meshes, fault regions are called *faulty cubes*. Figure 2 shows a 3-D mesh with four faulty nodes (3, 4, 2), (3, 5, 1), (3, 5, 2) and (5, 4, 2). Based on Definition 2, these four faulty nodes generate two disjoint faulty cubes: [3:3, 4:5, 1:2] (also called a *faulty block* in 2-D meshes) and [5:5, 4:4, 2:2], a single node. To study properties of a faulty cube, we first give the following definition: a *cube* is a solid that has six surfaces and any two cross sections perpendicular to the same surface generate two rectangles of the same size and shape.

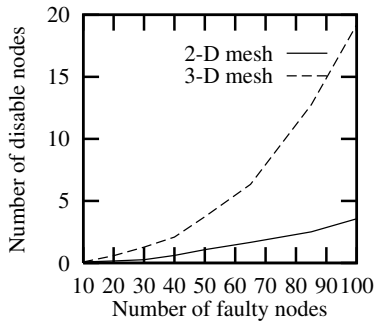
THEOREM 2. *In a 3-D mesh, a faulty cube defined by Definition 2 has the following properties:*

(1) *each faulty cube is a cube;*
(2) *each of the six surfaces of the faulty cube is perpendicular to an axis in 3-D meshes.*

Figure 3a shows the average number of (synchronous) rounds needed to form a set of disjoint fault regions in 2-D meshes (which is 100 × 100) and 3-D meshes (which is 21 × 21 × 21). Figure 3b shows the average number of non-faulty nodes marked as disabled in 2-D meshes and 3-D meshes. Results show that the average number of rounds needed to form fault regions is between one and four if the number of faulty nodes stays within 100. The number of non-faulty nodes marked as disabled is small compared to the number of faulty nodes in the system (this is especially true in 2-D meshes). Recently, Wu [26] proposed a new fault model called *Orthogonal Convex Fault Region* which can be built from a given fault region by enabling many disabled nodes. The bisection of the region in any 2-D space is an orthogonal convex polygon.



(a)



(b)

FIGURE 3. (a) The average number of rounds needed to construct fault regions. (b) The average number of non-faulty nodes marked disabled.

4. EXTENDED SAFETY LEVEL

In this section, an information model representing fault distribution is studied. We first extend the safety level concept to n-D meshes. The *Safety Level* [11] concept was originally proposed to capture limited global information in a binary hypercube. It was extended to 2-D meshes as *Extended Safety Level* [14] which includes four elements, each of which indicates the distance to the closest faulty block to east, south, west and north of the current node. The limited global information (captured by Extended Safety Level) at each node can be used to decide the feasibility of a minimal routing. The following shows an important theorem that leads to our Extended Safety Level definition in n-D meshes and it serves as a basis of our approach.

THEOREM 3. *Assume that node* $(0, 0, \ldots, 0)$ *is the source and node* $(u_1, u_2, \ldots, u_n)$ *is the destination. If there is no fault region that intersects with any of the axes, there exists at least one minimal path between* $(0, 0, \ldots, 0)$ *and* $(u_1, u_2, \ldots, u_n)$. *This result holds for any location of the destination and any number and distribution of fault regions.*

The above result can be strengthened by including the location of destination $(u_1, u_2, \ldots, u_n)$.

COROLLARY 3. *Assume that node* $(0, 0, \ldots, 0)$ *is the source and node* $(u_1, u_2, \ldots, u_n)$ *is the destination. If there is no fault region that intersects with the sections of* $[0, u_i]$ *along the* d_i*-axis for all* $i \in \{1, 2, \ldots, n\}$, *then there exists at least one minimal path between the source and destination nodes.*

Note that the role of source and destination can be interchanged if there exists a minimal path between them. That is, if there exists a minimal path from a source to a destination, then there exists a minimal path from the destination to the source. However, their roles cannot be interchanged in Theorem 3 (and Corollary 3). This is because if a source node is extended safe with respect to a destination, it does not imply that the destination is extended safe with respect to the source.

The following definition gives an extended safety level definition for n-D meshes. The source node $(0, 0, \ldots, 0)$ is associated with a $2n$-tuple $(p_1, n_1, p_2, n_2, \ldots, p_n, n_n)$, where p_i and n_i represent the distance to the closest fault region along the positive and negative d_i dimensions, respectively. Basically, Extended Safety Level is coded information about fault distribution in the neighborhood. Such information can be used to determine the existence of a minimal path between a given pair of source and destination nodes. Symbol $-$ is used to represent the fact that there is no fault region along the corresponding direction. A node is called *safe* if its extended safety level is $(-, -, \ldots, -)$; otherwise, it is *unsafe*. In a 3-D mesh, each node is associated with a vector (E, W, N, S, F, B) to represent the distance to the closest faulty cube along the east (positive d_1), west (negative d_1), north (positive d_2), south (negative d_2), front (positive d_3) and back (negative d_3) directions. In a 2-D mesh, each node is associated with a vector (E, W, N, S). Figure 4 shows a faulty 8×8 2-D mesh with extended safety levels associated with unsafe nodes.

DEFINITION 3. *The extended safety level of node* $(0, 0, \ldots, 0)$ *in a given n-D mesh is a* $2n$*-tuple:* $(p_1, n_1, p_2, n_2, \ldots, p_n, n_n)$. *This node is extended safe with respect to a destination* $(u_1, u_2, \ldots, u_n)$ *if* $|u_i| \leq p_i$ *(when* $u_i > 0$*) and* $|u_i| \leq n_i$ *(when* $u_i < 0$*) for all* $i \in \{1, 2, \ldots, n\}$*; otherwise, it is extended unsafe.*

An intuitive explanation of the extended safe node is the following: a node is extended safe to a destination node, as long as there is no fault region that intersects with the sections between the source and the destination along each axis. Based on Corollary 3, there always exists a minimal path between two nodes, as long as one node is extended safe with respect to the other.

Like fault regions, the extended safety level of each node can be calculated through iterative rounds of message exchanges among neighboring nodes. Assume that each node knows the status of its neighbors (faulty, enabled and disabled). When a node identifies a faulty or disabled neighbor, it passes information to the neighbor in the opposite direction. For example, if the neighbor to its positive d_i dimension is faulty or disabled, the current node passes information (distance '2' and direction 'positive d_i') to its neighbor at the negative d_i dimension. Once a node receives fault information it keeps a copy and increments its distance value by one before forwarding it to the neighbor

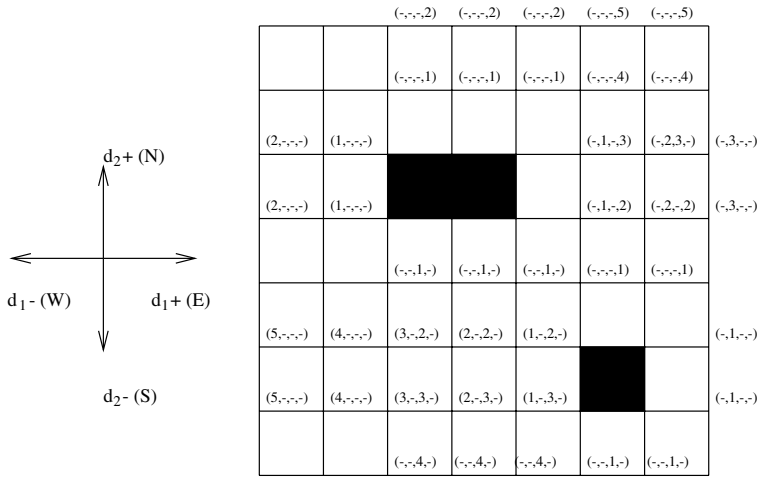


FIGURE 4. A faulty 8×8 2-D mesh with extended safety levels associated with unsafe nodes.

in the opposite direction. Clearly, each node will receive up to $2n$ distance values together with their directions from $2n$ different directions. The default value for each direction is $-$; that is, there will be no overhead when there is no fault in an n-D mesh. Since information is transmitted along one direction in a dimension, the number of (synchronous) rounds of message exchanges among neighboring nodes is bounded by k in a k-ary n-D mesh; that is, $O(k)$ in a k^n-node n-D mesh.

5. FAULT-TOLERANT ADAPTIVE AND MINIMAL ROUTING

In this section, we first show that any fully adaptive and minimal routing can be applied in any n-D meshes with fault regions, as long as the destination is extended safe with respect to the source. Then, we extend the result to any partially-adaptive and minimal routing. It is shown that the planar-adaptive routing [17] fails to meet the partially-adaptive routing requirement.

Our fault-tolerant adaptive and minimal routing is based on the following assumptions:

(1) the fault region defined earlier is used as the fault model;
(2) the source knows the extended safety level of the destination;
(3) each node knows the status of its adjacent nodes;
(4) only the static fault model is used, i.e. it is assumed that no new fault occurs during a routing process.

5.1. Fully adaptive and minimal routing

For the convenience of feasibility checking of a routing process, $(0, 0, \ldots, 0)$ is considered as a destination with an Extended Safety Level $(p_1, p_2, \ldots, p_n)$ and $(u_1, u_2, \ldots, u_n)$ as a source with an Extended Safety Level $(n'_1, n'_2, \ldots, n'_n)$ (still with $u_i \geq 0$ for all $i \in \{1, 2, \ldots, n\}$). Although destination (source) still holds a safety vector $(p_1, n_1, p_2, n_2, \ldots, p_n, n_n)$ $((p'_1, n'_1, p'_2, n'_2, \ldots, p'_n, n'_n))$, a subvector is used because of the specific locations of source and destination in the assumption.

The routing algorithm consists of two parts: *feasibility check* and *routing*. Feasibility check at the source is applied to check if it is possible to perform a minimal routing. This can be easily done by comparing the relative coordinates between the source and destination nodes with the safety vector of the destination.

FEASIBILITY_CHECK_n-D-MESHES
{At source $(u_1, u_2, \ldots, u_n)$, destination $(0, 0, \ldots, 0)$
with extended safety level $(p_1, p_2, \ldots, p_n)$}
if $(u_1, u_2, \ldots, u_n) \leq (p_1, p_2, \ldots, p_n)$
then returns YES
else returns NO.

FT-ROUTING_IN_n-D-MESHES
{At source $(u_1, u_2, \ldots, u_n)$}
if Feasibility_Check_n-D-Meshes = YES
then apply any fully adaptive and minimal routing in regular n-D meshes
else the proposed routing approach cannot be applied.

Before proving the correctness of the above approach, let us look at its application in 2-D meshes. Again, assume that $(0, 0)$ is the destination and node (u_1, u_2) is the source, with $u_1, u_2 \geq 0$. If there is no fault region (faulty block in 2-D meshes) that intersects with the d_1-axis and d_2-axis, then there exists at least one minimal path from (u_1, u_2) to $(0, 0)$, i.e. the length of this path is $|u_1| + |u_2|$. Results in [14] show that any fully-adaptive and minimal routing in a 2-D mesh can still be applied if the above condition holds and there is no need of additional fault information during the routing process. Whenever a packet reaches a faulty block, it just goes around the block towards the destination and it will never be forced on to a detour path or a trap where backtracking is required.

In fact, given a source s and a destination d in a 2-D mesh, all the intermediate nodes of a minimal path between s and d are enclosed in a *region of minimal paths* (RMP) (see Figure 5) which contains nodes and only nodes along

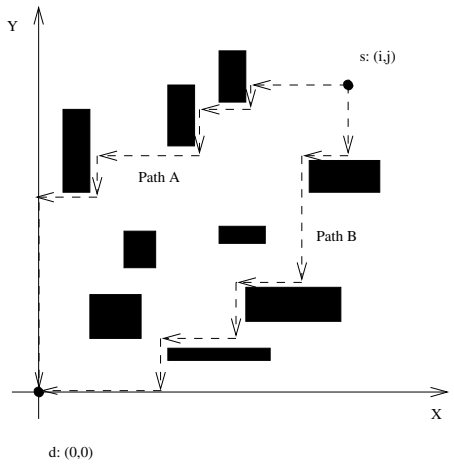


FIGURE 5. A sample RMP in 2-D meshes.

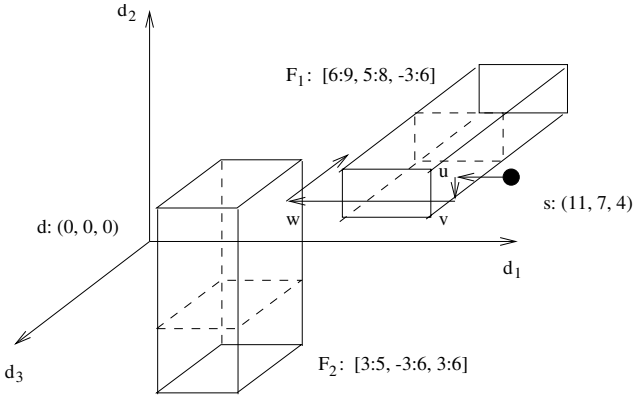


FIGURE 6. A routing example in 3-D meshes.

minimal paths between s and d. The RMP is constructed by determining two special paths (Path A and Path B) from source to destination. Starting from the source, Path A is constructed by going west (negative d_1) until the d_2-axis is reached and then by going along the d_2-axis towards the destination. If the path hits a faulty block, it goes around the faulty block by going South (negative d_2). It then makes a south-west turn when possible and continues going west (negative d_1). Path B is constructed in a similar way. It starts from the source and goes south (negative d_2) until the d_1-axis is reached and then goes along the d_1-axis towards the destination. If the path hits a faulty block, it goes around the faulty block by going west (negative d_1). It then makes a west-south turn when possible and continues going south.

It is clear that any fully-adaptive and minimal routing for regular 2-D meshes can still be applied to find a minimal path, as long as the destination meets the above condition. An intuitive explanation is that, because of the convex nature of a faulty block, each faulty block can block at most one dimension. Therefore, at least one dimension remains free for any source and destination pair that spans two dimensions. When the source (or an intermediate node) and destination pair spans only one dimension, the condition associated with the destination ensures that there is no faulty block along that dimension.

The correctness of the proposed algorithm can be described as follows through induction on dimension n. Clearly, this algorithm works for 2-D meshes. Assume that this algorithm works for meshes with up to $n-1$ dimensions. In n-D meshes, we assume that a source (or an intermediate node) and destination pair spans n dimensions; otherwise, the problem is reduced to minimal routing in l-D meshes (with $l < n$) and its correctness follows directly. In other words, at each intermediate node the packet can be forwarded along any one of the n dimensions. When an intermediate node is adjacent to a fault region, since each neighbor of a fault region is adjacent to exactly one disabled (or faulty) node in the fault region, the packet can still be forwarded along either one of the other directions (a fully-adaptive routing algorithm allows this). The disjoint property of fault regions ensures that the routing process can still enjoy $(n-1)$-D freedom until it hits either a new fault region or the offset along one dimension d_i is reduced to zero, i.e. $u_i = 0$. Based on the induction, a minimal path is guaranteed in the remaining routing process.

Figure 6 shows a routing example in a 3-D mesh with two faulty cubes. The routing starts from node $(11, 7, 4)$ and goes west (negative d_1). Once the routing packet hits faulty cube F_1, [6:9, 5:8, $-$3:6], at node u, it turns south (negative d_2) and makes a south-west turn at node v (which is the intersection of two adjacent surfaces of faulty cube F_1). The routing packet then goes west until it hits another faulty cube F_2, [3:4, $-$3:6, 3:5], at node w. It then turns back (negative d_3). Once the routing packet passes the intersection of two adjacent surfaces of faulty block F_2, the remaining routing resembles the one in a regular 3-D mesh without faulty cubes.

The proposed approach is simple, but a bit too conservative. In fact, the destination can use its subvector of the extended safety level $(n'_1, n'_2, \ldots, n'_n)$ to strengthen the sufficient condition for a minimal routing.

EXTENDED_FEASIBILITY_CHECK_n-D-MESHES
{At source $(u_1, u_2, \ldots, u_n)$ with extended safety level $(n'_1, n'_2, \ldots, n'_n)$ destination $(0, 0, \ldots, 0)$ with extended safety level $(p_1, p_2, \ldots, p_n)$}
if there exists i $(= 1, 2, \ldots, n)$ such that $(u_1, u_2, \ldots, u_i - (n'_i - 1), \ldots, u_n) \leq (p_1, p_2, \ldots, p_i, \ldots, p_n)$
then returns YES
else returns NO.

Identify a node v, $(u_1, u_2, \ldots, u_i - (n'_i - 1), \ldots, u_n)$, which is $n'_i - 1$ hops away from destination u along dimension d_i. The minimal routing can be divided into two phases: first a deterministic minimal route from u to v and second a dynamic minimal route from v to s. Figure 7 shows such a two-phase routing process in a 2-D mesh. Note that the deterministic routing constraint is essential. Otherwise, the routing process may enter a position (w in Figure 7) where no minimal path to the source exists. Clearly, the EXTENDED_FLEXIBILITY_CHECK_N-D_MESHES extends the sufficient condition of Theorem 3.

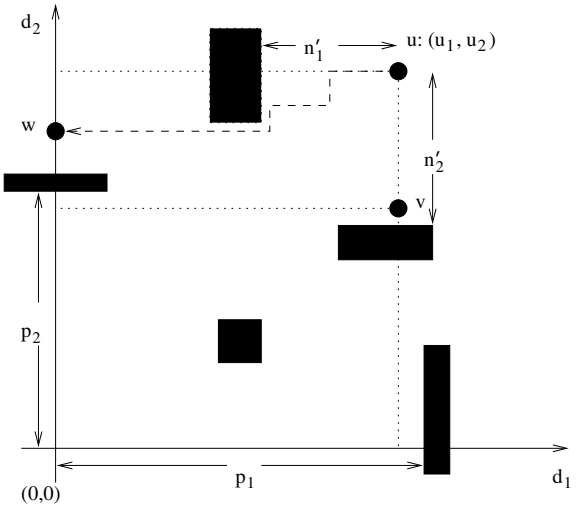


FIGURE 7. A two-phase routing process in 2-D meshes: from u to v and from v to s.

That is, a minimal path may still exist even if a fault region intersects with the section $[0, u_i]$ along the d_i-axis.

The sufficient condition can be further enhanced by making use of the neighbors' safety status. For example, a minimal routing is still possible if one of the preferred neighbors (of the source) meets the safety requirement with respect to the destination. A *sub-minimal routing* exists if one of the spare neighbors (ones that are not preferred neighbors) meets the safety requirement with respect to the destination. The path length in a sub-minimal routing is the corresponding minimal path plus two. In either case, the source first forwards the routing packet to the selected neighbor (preferred or spare) and the proposed routing process is applied with the selected neighbor being the new source. Other possible extensions of the sufficient condition are discussed in the section on extensions.

5.2. Minimal routing based on planar-adaptive routing

Deadlock due to dependencies on consumption resources (such as channels) is a fundamental problem in routing. A deadlock involving several routing processes occurs when there is a cyclic dependency for consumption channels. *Livelock* occurs when a routing packet travels around its destination node, never reaching it because the channels required to do so are occupied by other packets. Livelock is relatively easy to avoid, in fact any minimal routing is livelock-free.

To ensure freedom of deadlock and to support a 'truly' fully-adaptive routing without using the flow control mechanism, Linder and Harden [27] showed a *virtual network* approach that requires $O(2^n)$ *virtual channels* [28] in an n-D mesh. In this approach, each physical channel may support several logical or virtual channels multiplexed across the physical channel. The reason for using multiple channels is to avoid cyclic dependencies among channels to prevent deadlock. Other simpler approaches [29, 30] exist that support fully-adaptive routing using a constant number of virtual channels. However, routing decisions have to be made based on accurate buffer status, i.e. routing and flow control have to be coupled.

Planar-adaptive routing [17] is one of the popular partially-adaptive routings that requires few virtual channels (three), which, at the same time, allows flow control and routing to be decoupled. It offers cost-effectiveness in preventing deadlock while still keeping a certain degree of adaptivity. Planar-adaptive routing restricts the way the routing packet is routed. Specifically, the routing packet is routed following a series of 2-D planes, $A_1, A_2, \ldots, A_n$, in an n-D mesh. Each 2-D plane A_i is formed by two dimensions d_i and d_{i+1}. Plane A_n consists of dimensions d_n and d_1. Planes A_i and A_{i+1} share dimension d_{i+1}. However, the order of dimensions is arbitrary. If the offset in dimension d_i is reduced to zero, then routing can be immediately shifted to plane A_{i+1}. Applying this routing approach to 3-D meshes, we first construct two planes A_1 and A_2. Assume A_1 contains dimensions d_1 and d_2 and plane A_2 contains d_2 and d_3 (see Figure 8a). Again, assume that the source is (u_1, u_2, u_3) and the destination is $(0, 0, 0)$. The routing starts from (u_1, u_2, u_3) along plane A_1 which is plane $d_3 = u_3$; once the offset in dimension d_1 is reduced to zero it switches to plane A_2 which is plane $d_1 = 0$ (see Figure 8a).

Unfortunately, planar-adaptive routing cannot be directly applied to achieve fault-tolerant and minimal routing using our model. Consider a routing example in a 3-D mesh with source (3, 3, 3) and destination (0, 0, 0). Assume that there is a faulty cube [1:2, −1:4, 2:4]. Clearly, all the minimal paths from (3, 3, 3) in plane $A_1, d_3 = 3$, to any node along an adjacent line, $d_3 = 3$ and $d_1 = 0$, of planes A_1 and A_2 are blocked by the faulty cube.

However, if we strengthen the constraint at destination (0, 0, 0) in a 3-D mesh to the statement that: there is no faulty cube that intersects with planes $d_1 = 0$, $d_2 = 0$ and $d_3 = 0$, then the planar-adaptive routing can still be applied.

THEOREM 4. *Consider a 3-D mesh with faulty cubes. If there is no faulty cube that intersects with planes $d_1 = 0$, $d_2 = 0$ and $d_3 = 0$, then the planar-adaptive routing can be applied in* FT-ROUTING_IN_n-D-MESHES *to any source (u_1, u_2, u_3) to generate a minimal path to* $(0, 0, 0)$.

The above result can be extended to n-D meshes using a similar proof.

COROLLARY 4. *Consider an n-D mesh with fault regions. If there is no fault region that intersects with planes $d'_1 = d'_2 = \ldots = d'_{n-2} = 0$, where $\{d'_i i \mid \in \{1, 2, \ldots, n-2\}\}$ is a subset of $\{d_i \mid i \in \{1, 2, \ldots, n\}\}$ then the planar-adaptive routing can be applied in* FT-ROUTING_IN_n-D-MESHES *to any source $(u_1, u_2, \ldots, u_n)$ to generate a minimal path to* $(0, 0, \ldots, 0)$.

The above result shows that planar-adaptive routing can still be applied in an n-D mesh with fault regions under a strengthened constraint (i.e. a weaker sufficient condition associated with the destination node). Note that there are

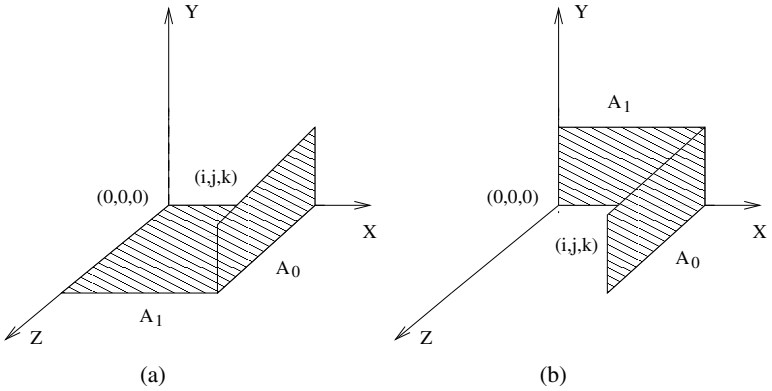


FIGURE 8. Planar-adaptive routing.

$\binom{n}{2}$ planes defined in Corollary 4, compared with n axes in Theorem 3. That is, it is less likely for a destination to meet the strengthened constraint than the one based on the extended safety level. Moreover, it is more difficult and expensive for each node to calculate its safety status under the strengthened constraint: each node needs to collect information in $\binom{n}{2}$ adjacent planes instead of nodes along n dimensions.

Clearly, the above problem stems from the planar-adaptive routing itself, which is too restrictive. The question is whether other partially-adaptive and minimal routing exists that can still be used in FT-ROUTING_IN_n-D-MESHES.

5.3. Partially-adaptive and minimal routing

The proposed fault-tolerant and minimal routing applies to any fully-adaptive routing in a regular n-D mesh but fails to apply to the planar-adaptive routing. The traditional X–Y routing in 2-D meshes is not a partially-adaptive routing, since at any step the routing process can have only one choice. The planar-adaptive routing also fails to meet the partially-adaptive routing requirement. In the 2-D plane A_i, when the offset in dimension d_{i+1} is first reduced to zero, the planar adaptive routing is forced to reduce the offset of d_i before switching to the 2-D plane A_{i+1}. That is, only one preferred direction can be selected even though more than one exists.

Partially-adaptive and minimal routing can also be ranked in terms of the degree of adaptivity. A set of preferred directions that can be selected at an intermediate node (including the source) is called a set of *legitimate preferred directions* at this node. A partially-adaptive routing R_1 is *more restrictive* than another one R_2 if at any intermediate node (including the source) the set of legitimate preferred directions of R_1 is a subset of that of R_2; in addition, the set of legitimate preferred directions of R_1 is a proper subset of that of R_2 at at least one intermediate node (including the source). Note that the relation 'more restrictive' is a partial order; that is, not every two partially-adaptive routing algorithms can be compared under this relation.

We introduce here a most restrictive partially-adaptive routing, called *dynamic planar-adaptive routing*. Like regular planar-adaptive routing, the routing packet is routed through a series of 2-D planes. Two adjacent planes still share a common dimension. The difference is that the planes in the series are dynamically generated. Again we use 3-D meshes to illustrate this approach. Suppose we select dimensions d_1 and d_2 in A_1, then there are two possible choices in selecting dimensions in A_2. One possibility is dimensions d_1 and d_3 and the other one is dimensions d_2 and d_3. Again, the routing starts from plane A_1, $d_3 = u_3$, and within this plane randomly reduces offsets in dimension d_1 and dimension d_2. If the offset in dimension d_1 is reduced to zero before that in dimension d_2, A_2 which spans dimensions d_2 and d_3 is selected (see Figure 8a); otherwise, A_2 that spans dimensions d_1 and d_3 is used (see Figure 8b).

LEMMA 5. *Consider a 3-D mesh with faulty cubes. If there is no faulty cube that intersects with the axes d_1, d_2 and d_3, then the dynamic planar-adaptive routing can be applied in* FT-ROUTING_IN_n-D-MESHES *to any source* (u_1, u_2, u_3) *to generate a minimal path to* $(0, 0, 0)$ *(assume that destination* $(0, 0, 0)$ *is extended safe with respect to the source).*

THEOREM 5. *Consider an n-D mesh with fault regions. If there is no fault region that intersects with any axis d_i, $i \in \{1, 2, \ldots, n\}$, then the dynamic planar-adaptive routing can be applied in* FT-ROUTING_IN_n-D-MESHES *to any source* $(u_1, u_2, \ldots, u_n)$ *to generate a minimal path to* $(0, 0, \ldots, 0)$ *(assume that destination* $(0, 0, \ldots, 0)$ *is extended safe with respect to the source).*

Based on the result of Theorem 5, we conclude that any partially-adaptive and minimal routing (which is less restrictive than the dynamic planar-adaptive routing) can be applied in our model.

6. EXTENSIONS

In this section, we discuss possible extensions, including an enhanced sufficient condition, the application of the proposed approach in an n-D torus and deadlock-free and livelock-free routing.

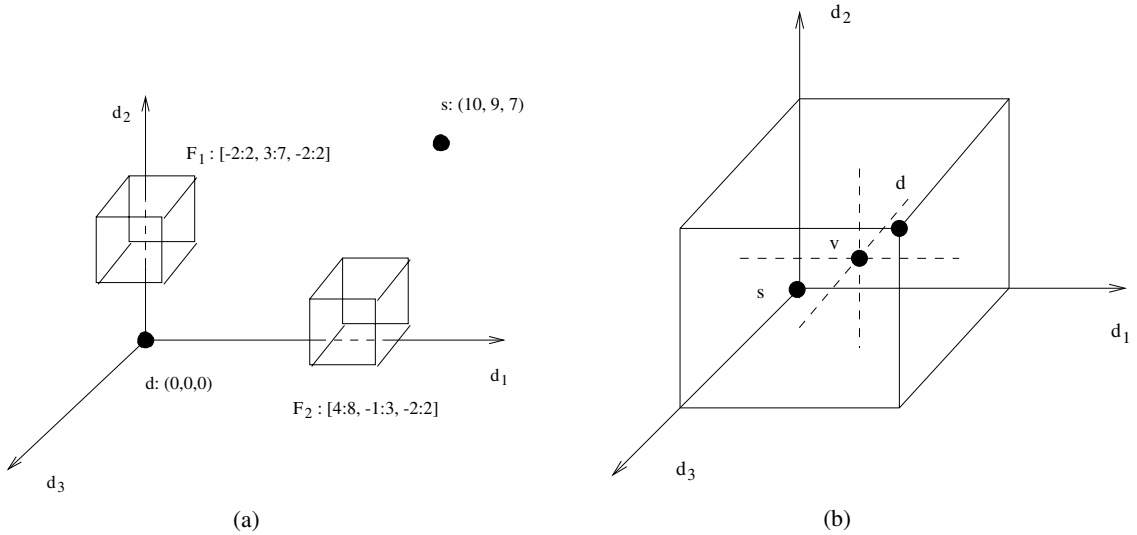


FIGURE 9. (a) A failure condition. (b) An enhanced sufficient condition.

6.1. An enhanced sufficient condition

Before considering possible extensions of the proposed model, we first re-examine the sufficient condition. The EXTENDED_FLEXIBILITY_CHECK_N-D_MESHES extends the sufficient condition of Theorem 3. That is, a minimal path exists even if a fault region intersects with the section $[0, u_i]$ along the d_i-axes. However, the sufficient condition still fails when fault regions intersect two or more axes. On the other hand, a minimal path may still exist as in the case of Figure 9a where two faulty cubes F_1 and F_2 intersect the d_2 and d_1 axes, respectively. However, a minimal path exists between nodes $(10, 9, 7)$ and $(0, 0, 0)$. Clearly, a stronger condition cannot be directly associated with source or destination node.

The following result provides an enhanced sufficient condition for the existence of a minimal path between nodes $(0, 0, 0)$ and (u_1, u_2, u_3). The condition is associated with node v in region $[0{:}u_1, 0{:}u_2, 0{:}u_3]$ (see Figure 9b).

THEOREM 6. *Consider two nodes* $(0, 0, 0)$ *and* (u_1, u_2, u_3) *in a 3-D mesh. If there exists a node* v, (v_1, v_2, v_3), *with* $0 \leq v_i \leq u_i$, *such that nodes along three line sections* $d_1 = v_1$ *and* $d_2 = v_2$, $d_1 = v_1$ *and* $d_3 = v_3$, $d_2 = v_2$ *and* $d_3 = v_3$ *within region* $[0{:}u_1, 0{:}u_2, 0{:}u_3]$ *are fault-free, a minimal path exists between* $(0, 0, 0)$ *and* (u_1, u_2, u_3).

Clearly, Theorem 3 is a special case of Theorem 6 with node s chosen as node v. Theorem 6 can be easily extended to n-D meshes using a similar proof. In the following, we use $(v_1, v_2, \ldots, v_{i-1}, *, v_{i+1}, \ldots, v_n)$ to represent a line section $(d_1 = v_1) \wedge (d_2 = v_2) \wedge \ldots \wedge (d_{i-1} = v_{i-1}) \wedge (d_{i+1} = v_{i+1}) \wedge \ldots \wedge (d_n = v_n)$, that is, this line section is along dimension d_i.

COROLLARY 7. *Consider two nodes* $(0, 0, \ldots, 0)$ *and* $(u_1, u_2, \ldots, u_n)$ *in an n-D mesh. If there exists a node* v, $(v_1, v_2, \ldots, v_n)$, *with* $0 \leq v_i \leq u_i$, *such that nodes along n line sections* $(*, v_2, \ldots, v_n)$, $(v_1, *, \ldots, v_n), \ldots, (v_1, v_2, \ldots, v_{n-1}, *)$ *within region* $[0{:}u_1, 0{:}u_2, \ldots, 0{:}u_n]$ *are fault-free, a minimal path exists between* $(0, 0, \ldots, 0)$ *and* $(u_1, u_2, \ldots, u_n)$.

6.2. Extensions to 3-D meshes with boundary

The proposed approach can also be applied to k-ary n-D meshes; that is, the number of nodes along a dimension is bounded by k. Healthy 'ghost' nodes are added along the boundary of each dimension. In this way, a given mesh with boundary is converted to one without boundary. Again, 3-D meshes are used to illustrate the approach. Faulty cubes can still be defined in the same way. For example, the south-west and front corner (a node with three adjacent 'ghost' nodes along west, south and front) has an extended safety level $(*, -, *, -, -, *)$, where $*$ represents a component that depends on the fault distribution in the given 3-D mesh.

Notice the difference between our approach and the one proposed by Chien and Kim [16]. In [16], 'ghost' nodes along the boundary of each dimension are considered faulty. In fact, the rule for enabled/disabled nodes is more complicated. Corner nodes (including ones with two or more adjacent 'ghost' nodes) are considered to have only one adjacent 'ghost' node. Based on Chien and Kim's fault region definition which is the same as the faulty cube definition, a faulty boundary node (a node with at least one adjacent 'ghost' node) will disable the whole 2-D boundary plane that contains this node (one of the six 2-D boundary planes of a 3-D mesh)! We can easily prove that if there is one fault in each cross-section (along an axis) of a given 3-D mesh, all the nodes in the 3-D mesh will be marked disabled based on Chien and Kim's model.

Consider an example of a $k \times k \times k$ mesh as shown in Figure 10. Suppose there is one column of faulty nodes (the black column in Figure 10a) in the $d_3 = c$ cross-section.

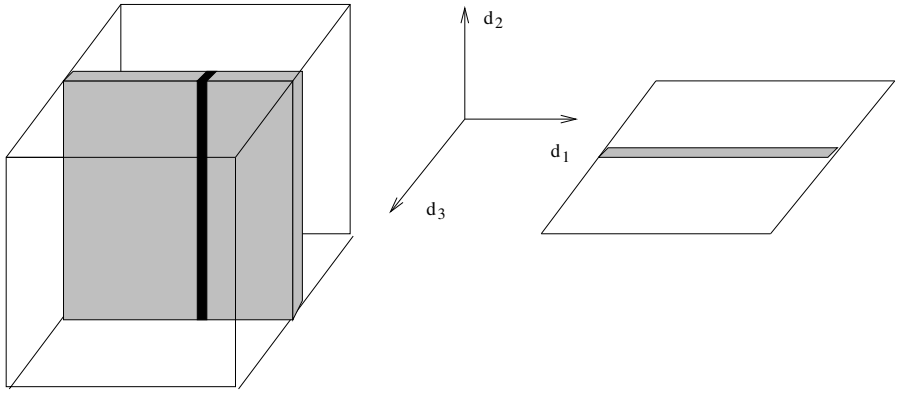


FIGURE 10. (a) Plane $d_3 = c$ and (b) plane $d_2 = i$, $i \in \{0, 1, \ldots, k-1\}$, in a $k \times k \times k$ mesh.

Based on Chien and Kim's fault region definition, the complete $d_3 = c$ cross-section will be disabled. Then for each cross section $d_2 = i$, $i \in \{0, 1, \ldots, k-1\}$, as shown in Figure 10, since one strip (marked as gray) is marked disabled, the corresponding cross-section is also disabled. Therefore, all the nodes in the given 3-D mesh are marked disabled (although there are only as few as k faulty nodes among k^3 nodes!). However, using our faulty cube model, all non-faulty nodes are marked enabled in the example of Figure 10. In other words, any partially-adaptive and minimal routing can still be applied as long as the destination meets the safety requirement.

Note that Chien and Kim's planar-adaptive routing cannot be applied using the fault model proposed in this paper, not even for non-minimal routing. Consider again the example of Figure 10, with two planes A_1 (d_1 and d_2) and A_2 (d_2 and d_3). Suppose the source is on cross-section $d_3 = c$ and is at the east side of the faulty column, and the destination is at the south-west corner of another cross-section $d_3 = c'$ ($c \neq c'$). Clearly, regular planar-adaptive routing fails, since the offset in dimension d_1 cannot be reduced to zero using any non-minimal path in plane $d_3 = c$, although there exists a minimal path in the 3-D mesh. Applying adaptive planar-adaptive routing, the offset in dimension d_2 will be first reduced to zero and then the routing continues on A_2 (d_1 and d_3) to the destination. The above argument and results can be easily extended to n-D meshes with boundary.

6.3. Extensions to n-D tori

Our results here can be easily extended to an n-D torus. A torus is a mesh with wraparound connections. Since an n-D mesh is a subgraph of an n-D torus, any solutions for n-D meshes can be directly applied to n-D tori. However, since an n-D torus has extra connections, solutions can be simplified and cost can be reduced. Another difference is that a fault region in an n-D torus may affect the safety level of a node in both directions of a dimension because of the wraparound links.

However, once the extended safety level has been decided at each node, the same sufficient condition (Theorem 3 and Corollary 3) can be applied in an n-D torus. Specifically, when the source and destination nodes are randomly distributed, say source $(u_1, u_2, \ldots, u_n)$ with safety vector $(p_1, n_1, p_2, n_2, \ldots, p_n, n_n)$ and destination $(0, 0, \ldots, 0)$, the conditions in Corollary 3 can be changed to the following: $|u_i| < p_i$ (if $u_i \bmod k \geq -u_i \bmod k$) or $|u_i| < n_i$ (if $u_i \bmod k \leq -u_i \bmod k$) for all i, where $1 \leq i \leq n$.

6.4. Deadlock and livelock freedom

Unlike many non-minimal fault-tolerant routing algorithms, the deadlock issue in the proposed model can be easily solved through the use of *virtual networks* [27], where a given physical network consists of several virtual networks. Each virtual network is partitioned into several virtual channels arranged in such a way that no cycle exists among channels, i.e. there is no *intra-virtual-network cycle*.

Partition a 3-D mesh into eight subnetworks: $d_1 + d_2 + d_3+$, $d_1 + d_2 + d_3-$, $d_1 + d_2 - d_3+$, $d_1 + d_2 - d_3-$, $d_1 - d_2 + d_3+$, $d_1 - d_2 + d_3-$, $d_1 - d_2 - d_2+$, $d_1 - d_2 - d_3-$. Figure 11a shows the subnetwork $d_1 + d_2 + d_3+$. Depending on the relative location of the source and destination nodes, one of the eight virtual subnetworks is selected and the corresponding routing can be completed within the selected subnetwork without using any other subnetwork. In this way, any *inter-virtual-network cycle* is avoided. Converting to virtual channel usage, this approach needs four virtual channels. For example, if source and destination are (u_1, u_2, u_3) and (0, 0, 0), respectively, if $u_1 > 0$, $u_2 < 0$ and $u_3 > 0$, subnetwork $d_1 - d_2 + d_3-$ is selected. To reduce the number of virtual channels, eight subnetworks can be paired together to form four subnetworks: $d_1 - d_2 - d_3*$, $d_1 * d_2 + d_3-$, $d_1 * d_2 + d_3+$ and $d_1 + d_2 - d_3*$, where $*$ stands for $+$ and $-$, i.e. a bidirectional channel. Figure 11b shows the $d_1 * d_2 + d_3+$ subnetwork. Clearly, at most three virtual channels are required along each dimension. It can be easily shown that three virtual channels are required for dynamic planar-adaptive routing for minimal routing. Therefore, within the context of minimal routing in 3-D meshes, dynamic planar-adaptive routing offers better fault

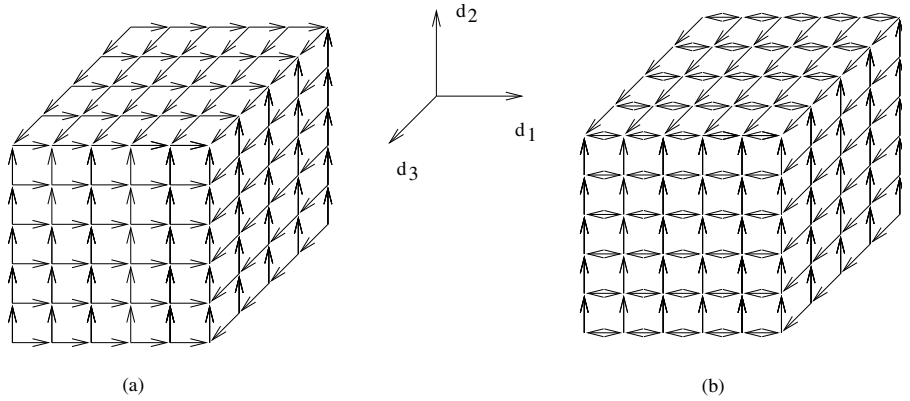


FIGURE 11. (a) A network partition $d_1 + d_2 + d_3+$. (b) A network partition $d_1 * d_2 + d_3+$.

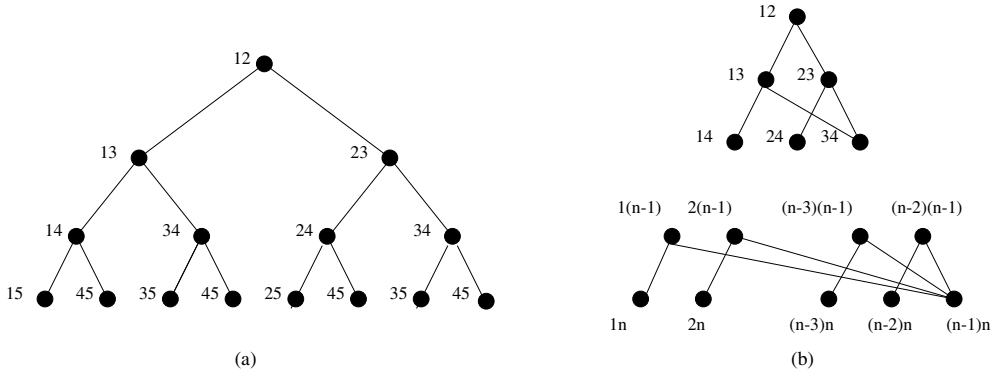


FIGURE 12. (a) The tree structure of adjacent 2-D planes. (b) The compact tree structure.

tolerance and adaptivity without using extra virtual channels compared with planar-adaptive routing.

In general, the adjacent 2-D planes in the dynamic planar-adaptive routing form a tree as shown in Figure 12a. Here, ij ($i, j \in \{1, 2, \ldots, n\}$ and $i < j$) represents a plane that includes dimensions d_1 and d_2. Clearly, planes that have the same dimensions can be combined to form a compact tree structure as shown in Figure 12b. Based on Figure 12b, a series of virtual subnetworks VSN_{ij} is constructed, with one subnetwork for each node ij of the tree in Figure 12b. Specifically,

$$VSN_{ij}: \quad d_i + d_j+, d_i + d_j-, d_i - d_j+, d_i - d_j - .$$

Clearly, two virtual channels are used in SN_{ij}. Since each, $i \in \{1, 2, \ldots, n\}$, appears exactly $n-1$ times in the compact tree in Figure 12b, $2(n-1)$ virtual channels are needed. A total order on *VSN* can be defined based on the partial order defined in Figure 12b. The use of these subnetworks strictly follows the total order to avoid cyclic dependency among virtual subnetworks *VSN*. To reduce the overall number of virtual channels, VSN_{ij} can be defined as

$$VSN_{ij}: \quad d_i + d_j*, d_i - d_j * .$$

Three virtual channels are used for VSN_{ij} with one for d_i and two for d_j. Since there are $n(n-1)/2$ *VSN*'s and n dimensions, based on the pigeon hole principle, at least $3(n-1)/2$ virtual channels are needed. The following theorem shows that it is possible to further reduce the number of virtual channels to n ($n+1$ is n is even) in an n-D mesh.

THEOREM 7. *It is possible to construct a set of virtual networks with n* ($n+1$ *when n is even*) *virtual channels for an n-D mesh to ensure freedom of deadlock using dynamic planar-adaptive routing.*

7. CONCLUSIONS

We have proposed a general theory of minimal routing in n-D meshes with fault regions. Unlike many traditional models that assume all the nodes know the global fault distribution or only adjacent fault information, our approach is based on the limited distribution of fault information. Specifically, we have proposed three fault-tolerant adaptive and minimal routing algorithms based on the proposed Extended Safety Level information associated with each node in n-D meshes. Our approach is the first attempt to provide insight into the design of fault-tolerant and minimal routing in n-D meshes. The Extended Safety Level model is a practical one that captures fault information in a concise format and supports various applications such as minimal routing. This study has shown that the safety level concept for binary hypercubes can still be effectively used in low-dimensional meshes with a proper

extension. We have also shown that any partially-adaptive and minimal routing can be applied as long as the destination node meets a certain condition. Our future research will focus on extending the proposed approach to collective communications [31], which include multicast, broadcast and barrier synchronization.

ACKNOWLEDGEMENT

This work was supported in part by NSF grants CCR 9900646 and ANI 0073736.

REFERENCES

[1] Dally, W. J. (1992) The J-machine: system support for Actors. In Hewitt and Agha (eds), *Towards Open Information Science*. MIT Press.

[2] Koeninger, R. K., Furtney, M. and Walker, M. (1994) A shared memory MPP from Cray research. *Digital Tech. J.*, **6**, 8–21.

[3] Silicon Graphics (1996) *Origin 200 and Origin 2000*. Technical Report, December.

[4] Seitz, C. L. (1988) The architecture and programming of the Ametek Series 2010 multicomputer. In *Proc. 3rd Conf. Hypercube Concurrent Computers and Applications*, Pasadena, CA, January, I, pp. 33–36. ACM Press.

[5] Lillevik, S. L. (1991) The Touchstone 30 gigaflop DELTA prototype. In *Proc. 6th Distrib. Memory Computing Conf.*, Portland, Oregon, May, pp. 671–677. IEEE Computer Society Press, Los Alamitos, CA.

[6] Chen, H. L. and Hu, S. H. (1997) Distributed submesh determination in faulty tori and meshes. In *Proc. 11th Int. Parallel Processing Symp.*, Geneva, Switzerland, April, pp. 65–70. IEEE Computer Society Press, Los Alamitos, CA.

[7] Dutt, S. and Hayes, J. P. (1992) Some practical issues in the design of fault-tolerant multiprocessors. *IEEE Trans. Comput.*, **41**, 588–598.

[8] Tzeng, N. F. and Lin, G. (1996) Maximum reconfiguration of 2-D mesh systems with faults. In *Proc. 1996 Int. Conf. Parallel Processing*, Bloomingdale, IL, August, I, pp. 77–84. CRC Press.

[9] Fraigniaud, P. and Gavoille, C. (1995) Memory requirement for universal routing schemes. In *Proc. 14th Annual ACM Symp. on Principles of Distributed Computing*, Ottawa, Ont., Canada, August, pp. 223–230. ACM Press.

[10] Wu, J. (1998) Adaptive fault-tolerant routing in cube-based multicomputers using safety vectors. *IEEE Trans. Parallel Distrib. Syst.*, **9**, 321–334.

[11] Wu, J. (1997) Reliable unicasting in faulty hypercubes using safety levels. *IEEE Trans. Comput.*, **46**, 241–247.

[12] Chen, M. S. and Shin, K. G. (1990) Depth-first search approach for fault-tolerant routing in hypercube multicomputers. *IEEE Trans. Parallel Distrib. Syst.*, **1**, 152–159.

[13] Fleury, E. and Fraigniaud, P. (1998) A general theory for deadlock avoidance in wormhole routed networks. *IEEE Trans. Parallel Distrib. Syst.*, **9**, 626–638.

[14] Wu, J. (2000) Fault-tolerant adaptive and minimal routing in mesh-connected multicomputers using extended safety levels. *IEEE Trans. Parallel Distrib. Syst.*, **11**, 149–159.

[15] Duato, J., Yalamanchili, S. and Ni, L. (1997) *Interconnection Networks: An Engineering Approach*. IEEE Computer Society Press, Los Alamitos, CA.

[16] Chien, A. A. and Kim, J. H. (1995) Planar-adaptive routing: low-cost adaptive networks for multiprocessors. *J. ACM*, **42**, 91–123.

[17] Chien, A. A. and Kim, J. H. (1992) Planar-adaptive routing: low cost adaptive networks for multiprocessors. In *Proc. 19th Int. Symp. on Computer Architecture*, Queensland, Australia, May, pp. 268–277. IEEE Computer Society Press, Los Alamitos, CA.

[18] Dally, W. J. and Seitz, C. L. (1986) The torus routing chip. *J. Distrib. Comput.*, **1**, 187–196.

[19] Kermani, K. and Kleinrock, L. (1979) Virtual cut-through: a new computer communication switching technique. *Comput. Networks*, **3**, 267–286.

[20] Gaughan, P. T. and Yalamanchili, S. (1995) A family of fault-tolerant routing protocols for direct multiprocessor networks. *IEEE Trans. Parallel and Distrib. Syst.*, **6**, 482–497.

[21] Lee, D. T., Yang, C. D. and Wong, C. K. (1996) Rectilinear paths among rectilinear obstacles. *Discrete Appl. Math.*, **70**, 185–215.

[22] Boppana, R. V. and Chalasani, S. (1995) Fault tolerant wormhole routing algorithms for mesh networks. *IEEE Trans. Comput.*, **44**, 848–864.

[23] Boura, Y. M. and Das, C. R. (1995) Fault-tolerant routing in mesh networks. In *Proc. 1995 Int. Conf. Parallel Processing*, Oconomowoc, WI, August, pp. 106–109. CRC Press.

[24] Libeskind-Hadas, R. and Brandt, E. (1995) Origin-based fault-tolerant routing in the mesh. In *Proc. 1st Int. Symp. High Performance Computer Architecture*, Raleigh, North Carolina, January, pp. 102–111. IEEE Computer Society Press, Los Alamitos, CA.

[25] Su, C. C. and Shin, K. G. (1996) Adaptive fault-tolerant deadlock-free routing in meshes and hypercubes. *IEEE Trans. Comput.*, **45**, 672–683.

[26] Wu, J. (2001) A distributed formation of orthogonal convex polygons in mesh-connected multicomputers. In *Proc. 15th Int. Parallel & Distrib. Processing Symp. (IPDPS 2001)*, San Francisco, April, p. 70. IEEE Computer Society Press, Los Alamitos, CA.

[27] Linder, D. H. and Harden, J. C. (1991) An adaptive and fault tolerant wormhole routing strategy for k-ary n-cubes. *IEEE Trans. Comput.*, **40**, 2–12.

[28] Dally, W. J. and Seitz, C. L. (1987) Deadlock-free message routing in multiprocessor interconnection networks. *IEEE Trans. Comput.*, **36**, 547–553.

[29] Duato, J. (1995) A necessary and sufficient condition for deadlock-free adaptive routing in wormhole networks. *IEEE Trans. Parallel Distrib. Syst.*, **6**, 1055–1067.

[30] Pifarre, G. D., Felperin, S. A., Gravano, L. and Sanz, J. L. C. (1991) Routing technique for massively parallel systems. *Proc. IEEE*, **74**, 488–503.

[31] Panda, D. K. (1995) Issues in designing efficient and practical algorithms for collective communication on wormhole-routed systems. In *Proc. 1995 ICPP Workshop on Challenges for Parallel Processing*, Oconomowc, Wisconsin, August, pp. 8–15. CRC Press.

APPENDIX

Proof of Theorem 1. (1) can be derived straightforwardly from the definition of fault region. In order to show that the distance between any two fault regions is at least two, we show that the distance is at least two and distance-two

cases exist. If two fault regions are less than distance two, they are adjacent and must be grouped into one based on the definition. When an n-D mesh contains two faulty nodes that are distance two apart along one dimension and they have the same coordinates along the other two dimensions, these two nodes form two fault regions. For example, faulty nodes $(u_1, u_2, \ldots, u_n)$ and $(u_1+2, u_2, \ldots, u_n)$ are such examples. The common adjacent node $(u_1 + 1, u_2, \ldots, u_n)$ is marked enabled. □

Proof of Theorem 2. Construct two 2-D planes that are perpendicular to an axis and both intersect with a given faulty cube. Without loss of generality, we assume that these two planes are adjacent, $d_1 = u_1$ and $d_1 = u_1 + 1$, and are perpendicular to the d_1-axis. If we can show that the two regions generated from the intersections of the faulty cube and these two planes are two identical rectangles, we prove that the faulty cube is a cube (since dimension d_1 can be replaced by d_2 and d_3 to show cases for other cross-sections). The shapes of the regions in both planes are rectangles based on the definition of faulty blocks in 2-D meshes [14]. Suppose the two rectangles are not identical. Without loss of generality, we assume that there exists a node (u_1, u_2, u_3) that is just outside the fault region in plane $d_1 = u_1$ (i.e. it has one neighbor on plane $d_1 = u_1$ that is inside the faulty region); however, node $(u_1 + 1, u_2, u_3)$ is inside the fault region in plane $d_1 = u_1 + 1$. Since node (u_1, u_2, u_3) is adjacent to two disabled nodes, it should be marked disabled based on Definition 1. This contradicts the condition that it is outside the fault region. Therefore, any cross-section perpendicular to the d_1-axis generates two rectangles of the same size and shape. The same procedure can be applied to the other two cross-sections. Therefore, the faulty cube is a cube. In addition, each of the six surfaces of the faulty cube is perpendicular to an axis in 3-D meshes. □

Proof of Theorem 3. We assume that $u_i \geq 0$ for all $i \in \{1, 2, \ldots, n\}$. Other cases can be proved in a similar way. This theorem is proved by induction on n, the dimension of the given mesh. When $n = 2$, it has been proved [14] that this theorem hold for 2-D meshes. Assume this theorem holds for all n-D meshes with $n < l$, we then prove this theorem holds for l-D meshes. It is assumed that u_i are non-zero in destination $(u_1, u_2, \ldots, u_l)$; otherwise, the routing is reduced to that within a mesh with a smaller dimension and, based on the induction assumption, the theorem holds.

We can take another level of induction on the number of fault regions (f) in an l-D mesh. When $f = 1$, i.e. there is only one fault region in the l-D mesh, the following minimal routing approach can be followed.

- Route along the negative d_l dimension, i.e. starting at the destination node, u_l (the distance to the destination along dimension d_l) is reduced (to zero) by going along the negative d_l dimension.
- If the fault region is not along the path, the routing process is reduced to that in an $(l - 1)$-D mesh.
- If the fault region is along the path, assume the fault region is $[f_1\colon f'_1, \ldots, f_{l-1}\colon f'_{l-1}, f_l\colon f'_l]$. Clearly, $f'_1 < u_l$. The routing process is blocked at node $(u_1, u_2, \ldots, u_{l-1}, f'_l + 1)$. The remaining routing process is done by going along the negative d_{l-1} dimension to node $(u_1, u_2, \ldots, 0, f'_l+1)$. The problem is again reduced to that in an $(l - 1)$-D mesh.

It is assumed that the theorem holds for l-D meshes with up to $f = k - 1$ faulty regions. Cases for l-D meshes with $f = k$ faulty regions are considered. Again, the above minimal routing approach is adopted.

- Route along the negative d_l dimension, i.e. starting at destination node u, u_l is reduced (to zero) by going along the negative d_l dimension.
- If the fault region is not along the path, the routing is reduced to that in an $(l - 1)$-D mesh.
- If there is a fault region along the path, assume the first fault region F is $[f_1\colon f'_1, \ldots, f_{l-1}\colon f'_{l-1}, f_l\colon f'_l]$. Clearly, $f'_l < u_l$. The routing process is blocked at node $(u_1, u_2, \ldots, u_{l-1}, f'_1 + 1)$. The remaining routing process is done by going along the negative d_{l-1} dimension to either node $(u_1, u_2, \ldots, f_{l-1} - 1, f'_l + 1)$ (it is a 'corner node' of the fault region) or node $(u_1, u_2, \ldots, 0, f'_l + 1)$, whichever comes first. In the later case, since the dimension size is reduced by one, the theorem holds true based on the induction assumption.
- Consider $(u_1, u_2, \ldots, f_{l-1} - 1, f'_l + 1)$ as a new destination (to the source $(0, 0, \ldots, 0)$). Since fault region F is no longer along any minimal path from the destination to the source, the number of fault regions is reduced by at least one in the routing process. Based on the second induction assumption, a minimal path exists.
- Combining two minimal paths, one from $(u_1, u_2, \ldots, u_l)$ to $(u_1, u_2, \ldots, f_{l-1} - 1, f'_l + 1)$ as a new destination (to the source $(0, 0, \ldots, 0)$) and another one from $(u_1, u_2, \ldots, f_{l-1} - 1, f'_l + 1)$ to $(0, 0, \ldots, 0)$, a minimal path from $(u_1, u_2, \ldots, u_l)$ to $(0, 0, \ldots, 0)$ is constructed. □

Proof of Theorem 4. Without loss of generality, assume that plane A_1 contains dimensions d_1 and d_2 and plane A_2 contains dimensions d_2 and d_3. The routing starts from (u_1, u_2, u_3) along plane A_1 (which is plane $d_3 = u_3$). Plane A_1 can be represented by that in Figure 5. The condition at destination $(0, 0, 0)$ ensures that there is no faulty block (a cross-section of a faulty cube) along the d_1- or d_2-axis. The minimal routing in this plane tries to reach any node along the d_2-axis. The existence of Path A ensures its feasibility. Once the routing packet reaches a node along the d_2-axis (meaning the offset in dimension d_1 has been reduced to zero), the routing process is then switched to plane A_2. The problem is then reduced to a minimal routing in a 2-D mesh. □

Proof of Lemma 5. Without loss of generality, assume that plane A_1 contains dimensions d_1 and d_2 and plane A_2 contains dimensions d_2 and d_3 or dimensions d_1 and d_3. The routing starts from (u_1, u_2, u_3) along plane A_1 (which is

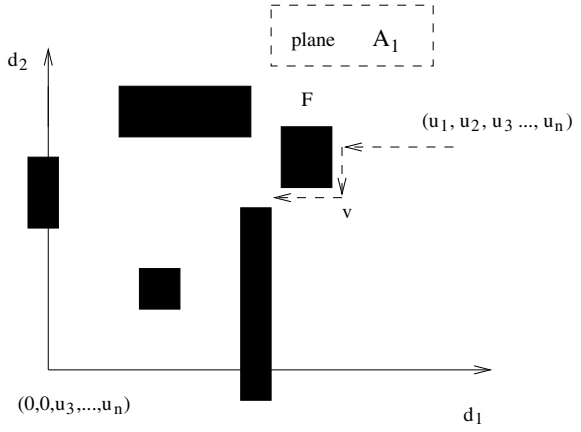


FIGURE 13. Routing in plane A_1.

plane $d_3 = u_3$). Some faulty blocks (cross-sections of faulty cubes) may intersect with both the d_1- and d_2-axes in plane A_1. The minimal routing in this plane tries to reach any node along either the d_1- or d_2-axis. The existence of such a minimal path can be proved by induction on f, the number of faulty blocks in this plane. When $f = 1$, the path starts from (u_1, u_2, u_3) and goes straight along negative d_2 until reaching the d_1-axis. If it hits the only faulty block, it turns south and goes along negative d_2 until reaching the d_1-axis. Assume that the theorem holds for $f = k-1$. When $f = k$, we use the same approach for the $f = 1$ case. Again, the path starts from (u_1, u_2, u_3) and goes straight along negative d_1 until it reaches the d_2-axis. If it hits a faulty block, say F, there are two cases: (1) if faulty block F intersects with the d_1 axis, then the routing packet turns south and goes along negative d_2 until it reaches the d_1-axis; (2) if faulty block F does not intersect with the d_1-axis, it still turns south to the lower-right corner v of the faulty block. Clearly, faulty block F becomes irrelevant to the remaining minimal routing from v to $(0, 0, 0)$. Since the number of faulty blocks is reduced to $k - 1$, a minimal path exists from v to $(0, 0, 0)$ based on the induction assumption. □

Proof of Theorem 5. The theorem is proved by induction on dimension n in a given n-D mesh. Based on Lemma 5 the theorem holds for 3-D meshes. Assume the theorem holds for meshes with up to $l - 1$ dimensions (with $l \geq 4$). Without loss of generality, assume that plane A_1 contains dimensions d_1 and d_2 and plane A_2 contains dimensions d_1 and d_3 or dimensions d_2 and d_3. The routing starts from $(u_1, u_2, \ldots, u_l)$ along plane A_1. Plane A_1 can be represented by that in Figure 13. Some faulty blocks (cross-sections of fault regions) may go across both the d_1- and d_2-axes of node $(0, 0, u_3, \ldots, u_n)$. The minimal routing in this plane tries to reach any node along either the d_1- or d_2-axis. The existence of such a minimal path can be proved by induction on f, the number of faulty blocks in this plane. When $f = 1$, the path starts from $(u_1, u_2, \ldots, u_l)$ and goes straight along the negative d_1 direction until it reaches the d_2-axis. If it hits the only faulty block, it turns south and goes along negative d_2 until it reaches the d_1-axis (of node $(0, 0, u_3, \ldots, u_n)$). Assume that the theorem holds for $f = k - 1$. When $f = k$, we use the same approach for the $f = 1$ case. Again, the path starts from $(u_1, u_2, \ldots, u_l)$ and goes straight along the negative d_1 until it reaches the d_1-axis. If it hits a faulty block (a cross-section of a fault region), say F (see Figure 13), there are two cases: (1) if faulty block F intersects with the d_1-axis, then the routing packet turns south (negative d_2) and goes along negative d_2 until it reaches the d_2-axis; (2) if faulty block F does not intersect with the d_1-axis, it still turns south to the lower-right corner v of the faulty block (see Figure 8). Clearly, faulty block F becomes irrelevant to the remaining minimal routing from v to $(0, 0, \ldots, 0)$. Since the number of faulty blocks is reduced to $k - 1$, a minimal path exists from v to $(0, 0, \ldots, 0)$ based on the induction assumption on the number of faulty blocks.

Once the routing packet reaches the d_1- or d_2-axis (or both), the number of dimensions that spans from the (new) source to the destination nodes reduces by at least one. A minimal path exists from the new source to the destination based on the induction assumption on the dimension size. □

Proof of Theorem 6. Applying Theorem 3 to nodes v and s with node v being the source and s the destination, we know that there exists a minimal path between v and s. We then apply Theorem 3 to nodes v and d with node v being the source and d the destination. Again a minimal path between d and v is found. Combining these two minimal paths, a minimal path between s and d is constructed. □

Proof of Theorem 7. We have the following constructive proof. Assume dimension n is odd (if the given dimension is even, add one to it to make it odd) and $k = (n - 1)/2$. d_{i+j} (including $j = 0$) should be interpreted as $d_{(i+j-1) \bmod n+1}$. The following n virtual subnetworks are constructed:

$$
\begin{array}{ll}
VSN_1: & d_1 + d_2 * d_3 * \ldots * d_{k+1}*, \\
 & d_1 - d_2 * d_3 * \ldots * d_{k+1}*, \\
VSN_2: & d_2 + d_3 * d_4 * \ldots * d_{k+2}*, \\
 & d_2 - d_3 * d_4 * \ldots * d_{k+2}*, \\
\ldots & \ldots \\
VSN_i: & d_i + d_{i+1} * d_{i+2} * \ldots * d_{i+k}*, \\
 & d_i - d_{i+1} * d_{i+2} * \ldots * d_{i+k}*, \\
\ldots & \ldots \\
VSN_n: & d_n + d_{n+1} * d_{n+2} * \ldots * d_{n+k}*, \\
 & d_n - d_{n+1} * d_{n+2} * \ldots * d_{n+k}*.
\end{array}
$$

In the above subnetwork, VSN_i covers k nodes $d_i d_{i+1}$, $d_i d_{i+2}, \ldots, d_i d_{i+k}$ of Figure 12b. Also VSN_i, $i \in \{1, 2, \ldots, n\}$, cover all nodes in Figure 12b. The sequence $VSN_1, VSN_2, \ldots, VSN_n$ respects the partial order defined in Figure 12b. In the VSN each d_i* appears $2k$ times and d_i+ and d_i- once each; therefore, $2k + 1 = n$ virtual channels are used. □