Adaptive Fault-Tolerant Routing in Cube-Based Multicomputers Using Safety Vectors

Jie Wu, *Senior Member*, *IEEE*

Abstract—Reliable communication in cube-based multicomputers using the safety vector concept is studied in this paper. In our approach, each node in a cube-based multicomputer of dimension n is associated with a safety vector of n bits, which is an approximated measure of the number and distribution of faults in the neighborhood. The safety vector of each node can be easily calculated through $n - 1$ rounds of information exchange among neighboring nodes. Optimal unicasting between two nodes is guaranteed if the kth bit of the safety vector of the source node is one, where k is the Hamming distance between the source and destination nodes. The concept of *dynamic adaptivity* is introduced, representing the ability of a routing algorithm to dynamically adjust its routing adaptivity based on fault distribution in the neighborhood. The feasibility of the proposed unicasting can be easily determined at the source node by comparing its safety vector with the Hamming distance between the source and destination nodes. The proposed unicasting can also be used in disconnected hypercubes, where nodes in a hypercube are disjointed (into two or more parts). We then extend the safety vector concept to general cube-based multicomputers.

Index Terms—Disconnected networks, fault tolerance, generalized hypercubes, multicomputers, reliable communication, unicast.

1 Introduction

With its numerous attractive features, the binary hypercube has been one of the popular topological structures for distributed-memory systems. An *n-dimensional hypercube* (n-cube) system consists of exactly 2^n processors which can be addressed distinctively by n-bit binary numbers. Two nodes are directly connected by a link if and only if their binary addresses differ in exactly one bit position. The hypercube structure has been used in several experimental and commercial machines including NCUBE-2 [7] and Intel iPSC [17]. Many variations of the hypercube topology have been proposed to improve certain parameters, such as diameter, node degree, etc. A *cube-based multicomputer* refers to a multicomputer system where communication is based on message passing and processors are connected using the hypercube topology, or one of its variations. Two well-known variations of the hypercube topology are *generalized hypercubes* [2] and *cube-connected-cycles* [15].

1.1 Problem Definition

Efficient interprocessor communication is a key to the performance of a cube-based multicomputer. *Unicasting* is a one-to-one communication between two nodes; one is called the source node and the other the destination node. Unicasting in fault-free hypercubes and its variations have been extensively studied ([8], [14], [18], [19]). As the number of processors in a cube-based multicomputer increases, the probability of processor failure also increases. There have been a number of fault-tolerant unicasting schemes proposed ([4], [5], [9], [10], [11], [16]). Most of them assume that each node knows either only the status of its neighbors (such a model is called *local-information-based*) or the status of all the nodes (*global-information-based*). The main challenge is to devise a simple and effective way of representing fault information such that an optimal (or suboptimal) unicast algorithm can be designed based on such information. An *optimal unicast algorithm* (also called *minimal unicast algorithm*) forwards a message to the destination node through a minimal path (also called a Hamming distance path). A routing is adaptive if it can use alternative paths between the source and destination nodes, making more efficient use of network bandwidth and providing resilience to failure.

1.2 Related Results

Normally, global-information-based routing can obtain an optimal or suboptimal result. However, a separate process is needed to collect global information. In some cases, global information is captured in a more concise form, as in [16], where the concept of *fault-free dimension* was used. A dimension is fault-free if there is no pair of faulty nodes across this dimension. It has been shown that fault-free dimensions can be found in $O(n)$ steps in an n-cube. A nonminimal unicasting algorithm based on fault-free dimension has also been proposed.

Local-information-based routing uses a weaker but more reasonable assumption; however, such an approach can achieve only local optimization and is heuristic in nature. Therefore, the length of a routing path is unpredictable (most likely bounded) in general, and global optimization, such as time and traffic, is difficult. For example, in [9], a sidetracking

• *J. Wu is with the Department of Computer Science and Engineering, Florida Atlantic University, Boca Raton, FL 33431.*
E-mail: jie@cse.fau.edu.

Manuscript received 26 Feb. 1996; revised 30 Jan. 1997.
For information on obtaining reprints of this article, please send e-mail to: tpds@computer.org, and reference IEEECS Log Number 100133.

approach was used to route messages in a faulty hypercube with node faults. A message is rerouted to a randomly chosen fault-free neighboring node when there is no fault-free neighbor along a minimal path to the destination node. In [5], Chen and Shin proposed a scheme based on depth-first search in which backtracking is required when all forward links are blocked by faulty components (links and/or nodes). However, a history of visited nodes has to be kept as part of the routing message. A simplified version of this approach, which tolerates fewer faults, was presented in [10], where routing is progressive without backtracking. Still, it is a nonminimal routing. Moreover, most of the existing fault-tolerant unicasting algorithms do not address the routing issue in *disconnected hypercubes*, where nonfaulty nodes in a faulty hypercube are partitioned into several disconnected subcubes. The challenge is to distinguish an infeasible routing, where the source and destination nodes are in disconnected subcubes, from a feasible routing, where the source and destination nodes are in the same subcube.

Limited-global-information-based routing is a compromise between local-information-based and global-information-based approaches. A routing algorithm of this type normally obtains an optimal or suboptimal solution and requires a relatively simple process to collect and maintain fault information in the neighborhood (such information is called limited global information). Therefore, such an approach can be more cost effective than the ones based on global or local information. One simple but ineffective approach is to use k-Hamming-distance information in which each node knows the status of all components within k distance. However, optimality cannot be guaranteed, as a routing process could possibly go to either a state where all minimal paths are blocked by faulty components or a dead end where backtracking is required. In addition, each node has to maintain a relatively large table containing k-Hamming-distance information.

Lee and Hayes [11] proposed the safe node concept to capture limited global information, where nonfaulty nodes are classified into *safe* and *unsafe*. A nonfaulty node is unsafe if and only if there are at least two unsafe or faulty neighbors. The safe node set can be decided in $O(n^2)$ rounds of information exchange among neighboring nodes in an n-cube. A routing algorithm was proposed based on the status of each node and can route a message via a path with a length of no more than two plus the Hamming distance between the source and destination nodes as long as the n-cube under consideration is not fully unsafe. An n-cube will not be fully unsafe if the number of faults is no more than $\lceil \frac{n}{2} \rceil$. A similar safety node concept has been used to represent convex-type fault blocks in a two-dimensional mesh [3].

Wu and Fernandez [23] extended the Lee and Hayes' safe node concept by relaxing certain conditions and, hence, increased the size of safe node set and the degree of fault tolerance. A nonfaulty node is unsafe if and only if there are two faulty neighbors or there are at least three unsafe or faulty neighbors. The process which identifies node status needs fewer rounds than the one in [11]. However, it still requires $O(n^2)$ rounds in the worst case. A unicast algorithm based on this extended safe node concept was proposed by Chiu and Wu [6]. They showed that a path with a length of no more than the Hamming distance between the source and destination nodes plus four can always be established as long as the hypercube is not fully unsafe.

The *safety level* concept [22] is one further step to extend the safe node concept in an n-cube. In this model, each node is assigned a safety level k, $0 \le k \le n$. A node with a safety level $k = n$ is called safe and a faulty node is assigned the lowest level 0. If a node has a safety level k, there is at least one Hamming distance path from this node to any node within k-Hamming-distance. The safety level concept covers a larger set of safe nodes than the ones based on Lee-Hayes' and Wu-Fernandez' definitions, and it can be used for unicasting in various faulty hypercubes (including disconnected hypercubes) more effectively. When a faulty n-cube has fewer than n faulty nodes, the unicasting algorithm based on safety level ensures an optimal unicast or suboptimal unicast (generating a path with a length of the Hamming distance between the source and destination nodes plus two). Wu [22] also proved that, under both Lee-Hayes' and Wu-Fernandez' safe node definitions, the safe node set is empty for any disconnected hypercube. That is, both unicast algorithms proposed by Lee-Hayes [11] and Chiu-Wu [6] are not applicable to disconnected hypercubes. However, the safety level concept has the following two pitfalls:

1) Suppose the safety level of a node is k; its safety level only tells that there exists a Hamming distance path to any node within k-Hamming-distance. There is no information about the existence of a Hamming distance path to nodes that are more than k-Hamming-distance away.
2) The safety level concept applies to hypercubes with faulty nodes, but it is ineffective to cover link faults.

1.3 Proposed Approach

The *safety vector* concept proposed in this paper can effectively include faulty link information and provide more accurate information about the number and distribution of faults in an n-cube. Basically, each node in an n-cube is associated with a bit vector, called a safety vector, calculated through $n - 1$ rounds of information exchange among neighboring nodes. An optimal unicast between two nodes is guaranteed if the kth bit of the safety vector of the source node is one (this bit is set), where k is the Hamming distance between the source and destination nodes. Again, unicasting based on safety vectors can also be used in disconnected hypercubes by distinguishing infeasible routings from feasible ones. We also show that the safety vector concept can be extended to other cube-based multicomputers, such as generalized hypercubes.

A novel concept of *dynamic adaptivity* is introduced to represent the ability of a routing algorithm to dynamically adjust its adaptivity based on fault distribution in the neighborhood. More specifically, a routing algorithm is k-adaptive at an intermediate node if it can use k neighbors that are along minimal paths between this intermediate node and the destination node. In fault-free hypercubes, the maximum k value of an intermediate node is the Hamming distance between this node and the destination node. In the presence of faults, this k value is normally smaller than the Hamming distance depending on

fault distribution. In our approach, safety vectors are used to achieve maximum possible adaptivity at each intermediate node and the routing algorithm dynamically adjusts its adaptivity among fully adaptive, partially adaptive, and deterministic (one-adaptivity) algorithms depending on the safety degree of the intermediate node. zero-adaptivity corresponds to an infeasible unicasting, which can be easily determined at the source node by comparing its safety vector with the Hamming distance between the source and destination nodes.

We make the following assumptions in this paper:

1) All faults are fail-stop, i.e., there are no malicious faults.
2) Fault detection and diagnosis algorithms exist, but we do not require such algorithms to be perfect.

We do assume that each node knows the safety status of all its neighbors and can distinguish an adjacent faulty link from an adjacent faulty node. Throughout the paper, the terms unicasting and routing are used interchangeably.

2 Notation and Preliminaries

2.1 Hypercubes

An n-cube (Q_n) is a graph having 2^n nodes labeled from 0 to $2^n - 1$. Two nodes are joined by an edge if their addresses, as binary numbers, differ in exactly one bit position. More specifically, every node u has an address $u(n)u(n-1)\cdots u(1)$ with $u(i) \in \{0, 1\}$, $1 \le i \le n$, and $u(i)$ is called the ith bit (dimension) of the address. We denote node $u^{(i)}$ the neighbor of u along dimension i. $u^{(i)}$ is calculated by setting or resetting the ith bit of u. For example, $1101^{(3)} = 1001$. Note that this notation can be used to set or reset the ith bit of any binary string. A faulty n-cube includes faulty nodes and/or links. A faulty n-cube may or may not be disconnected depending on the number and location of faults. A path connecting two nodes s and d is termed a minimal path (also called a Hamming distance path) if its length is equal to the Hamming distance between these two nodes. An optimal (or minimal) routing is one which always generates a minimal path. In general, optimal routing has a broader meaning which always generates a shortest path, not necessarily a minimal one, among the available ones. It is possible that all minimal paths are blocked by faults. In this case, a shortest (available) path is not a minimal one. In this paper, this situation will never occur and we use the terms shortest and minimal interchangeably.

The distance between two nodes s and d is equal to the Hamming distance between their binary addresses, denoted by $H(s, d)$. Symbol $\oplus$ denotes the bitwise exclusive OR operation on binary addresses of two nodes. Clearly, $s \oplus d$ has value 1 at $H(s, d)$ bit positions corresponding to $H(s, d)$ distinct dimensions. These $H(s, d)$ dimensions are called *preferred dimensions* and the corresponding nodes are termed *preferred neighbors*. The remaining $n - H(s, d)$ dimensions are called *spare dimensions* and the corresponding nodes are *spare neighbors*. A minimal path can be obtained by using links at each of these $H(s, d)$ preferred dimensions in some order. For example, suppose $s = 0101$ and $d = 1011$, then $s \oplus d = 0101 \oplus 1011 = 1110$. Therefore, dimensions 4, 3, 2 are preferred dimensions and dimension 1 is a spare dimension. Among neighbors of $s = 0101$, nodes 1101, 0001, and 0111 are preferred neighbors and node 0100 is a spare neighbor. Any path from $s = 0101$ to $d = 1011$ that uses links at dimensions 4, 3, and 2 in some order is a minimal path, e.g., $0101 \rightarrow 0001 \rightarrow 1001 \rightarrow 1011$ is a minimal path from 0101 to 1011.

2.2 Dynamic Adaptivity

We introduce the concept of *dynamic adaptivity* representing the ability of a routing algorithm to dynamically adjust its adaptivity based on fault distribution in the neighborhood. To simplify our discussion, we define dynamic adaptivity only in the context of minimal routing.

Definition 1. *A minimal routing algorithm is k-adaptive at intermediate node u with respect to destination node d if it can select the next forwarding node among k neighbors of node u which are along minimal paths from node u to node d.*

A routing algorithm is fully adaptive if it can use all possible minimal paths between source and destination nodes. Obviously, a hypercube routing algorithm is fully adaptive at node u with respect to destination node d if it is k-adaptive at u, where $H(u, d) = k$. Partially adaptive routing algorithms use only a subset of available minimal paths between the source and destination nodes. A routing algorithm is deterministic if it is one-adaptive at each intermediate node. The traditional e-cube routing is a deterministic routing.

In a hypercube with faulty nodes and links, it is possible that all minimal paths from intermediate node u are blocked by faults. For example, assume that intermediate node $u = 0111$ and destination node $d = 0010$ in the four-cube shown in Fig. 1, where faulty nodes are represented by black nodes. Two minimal paths from 0111 to 0010 are blocked by faulty nodes 0110 and 0011, respectively. Therefore, any routing algorithm is zero-adaptive at 0111 (with respect to destination 0010). A routing algorithm is *infeasible* if it is zero-adaptive at an intermediate node. Clearly, any routing algorithm that forwards a message with destination node 0010 to node 0111 becomes infeasible.

The challenge is to find a feasible routing algorithm without sacrificing adaptivity. In the example of Fig. 1, suppose that 1111 is the source node and 0010 is the destination node; if the cube was fault-free, the maximum adaptivity at source 1111 would be $k = H(1111, 0010) = 3$. However, in the faulty four-cube shown in Fig. 1, node 0111 cannot be used as an intermediate node since it will cause zero-adaptivity. Therefore, the maximum adaptivity at source 1111 with respect to destination 0010 is two. That is, the message can be forwarded to either 1110 or 1011, but not 0111. To find maximum adaptivity, we need a simple but effective way of representing fault distribution in the neighborhood. The safety vector concept discussed in the next section provides such a mechanism.

3 The Safety Vector and Its Properties

3.1 Safety Vectors

In the proposed approach, fault information is captured in a safety vector of n bit numbers, $(u_1, u_2, \ldots, u_n)$, associated with each node u in an n-cube. Specifically, u_k represents the routing capability of node u to a k-Hamming-distance node.

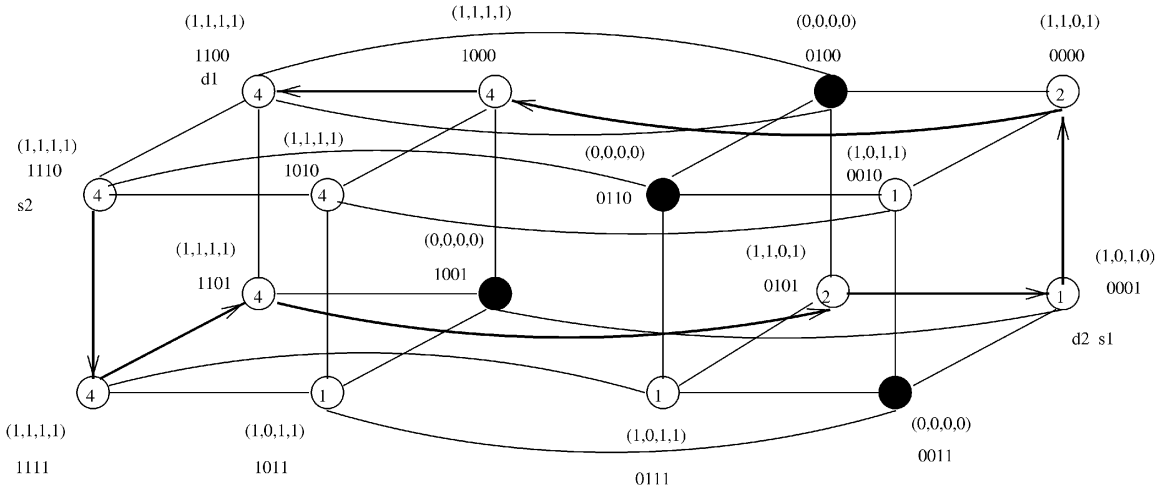

Fig. 1. A four-cube with four faulty nodes.

Based on the topological property of a binary hypercube, the kth bit of a safety vector can be determined from the $(k-1)$th bit of its neighbors' safety vectors, except for the first bit, which can be decided locally. Let $\left(u_1^{(i)}, u_2^{(i)}, \ldots, u_n^{(i)}\right)$ be the safety vector of node u's neighbor along dimension i. We have the following inductive definition of safety vector.

DEFINITION 2.

- *The safety vector of a faulty node is* (0, 0, ..., 0). *If node u is an end node of a faulty link, the other end node will be registered with a safety vector of* (0, 0, ..., 0) *at node u.*
- *Base for the first bit*:

$$u_1 = \begin{cases} 0 & \text{if node } u \text{ is an end node of a faulty link} \\ 1 & \text{otherwise.} \end{cases}$$

- *Inductive definition for the kth bit:*

$$u_k = \begin{cases} 0 & \text{if } \sum_{1 \le i \le n} u_{k-1}^{(i)} \le n-k \\ 1 & \text{otherwise.} \end{cases}$$

The safety vector defined in Definition 2 ensures the following property: If the kth bit of the safety vector of node u is one, then for any k neighbors, there exists at least one neighbor that has 1 in the $(k-1)$th bit of its safety vector. That is, at least one preferred neighbor has its $(k-1)$th bit set. Note that this neighbor is one step closer to the destination. Using this property inductively, we can construct a minimal path to any destination which is k-Hamming-distance away from node u. This property will be discussed in detail later and it will be used as a basis of the proposed routing scheme.

The safety degree of node u is strictly higher than the safety degree of node v if and only if $u_k \ge v_k$ for all $1 \le k \le n$ and there exists at least one l such that $u_l > v_l$. Note that the safety degree is a partial order, not all pairs of safety vectors can be ordered, such as vectors (1, 0) and (0, 1) in a two-cube. A faulty node is associated with (0, 0, ..., 0) which corresponds to the lowest order of safety degree, while a node with (1, 1, ..., 1) as its safety vector has the highest order of safety degree and the corresponding node is called a *safe node*; otherwise, it is called an *unsafe node*.

Note that there are different views of safety status between two end nodes of a faulty link. Because these two nodes cannot reach each other and, hence, cannot determine the actual safety vector of the other. Therefore, each is assumed to have a safety vector (0, 0, ..., 0) by another. However, other neighbors treat these two nodes as regular nodes by obtaining their actual safety vectors. The safety vector of a nonfaulty node is inductively defined in terms of its neighbors' safety vectors. The following theorem shows that, for any faulty hypercube, there is one and only one safety vector for each node that satisfies the condition in Definition 2.

THEOREM 1. *The safety vector concept is well-defined; that is, the safety vector associated with each node in a given n-cube is unique.*

Proof. Note that the first bit of each safety vector is predetermined. That is, it is 0 if the corresponding node is faulty or it is an end node of a faulty link; otherwise, it is 1. Based on the definition of a safety vector, the kth bit depends on all the $(k-1)$th bits of neighbors' safety vectors. Clearly, all the kth bits can be determined once all the $(k-1)$th bits are fixed. Because all the first bits are fixed, there is only one possible assignment of all the other bits. □

Algorithm GLOBAL_STATUS (GS)

begin /* **determine safety vector** $(u_1, u_2, \ldots, u_n)$ **of node** u **in** n-**cube** Q_n */
  forall $u \in Q_n$ /* **determine the first bit** u_1 */
    if u is an end node of a faulty link
    then $u_1 = 0$ **else** $u_1 = 1$;
  for $k = 2$ **step** 1 **to** n /* **determine the** k**th bit** u_k, **where** $2 \le k \le n$ */
    forall $u \in Q_n$
      collects all the $(k-1)$th bits, $u_{k-1}^{(i)}$, of neighbors' safety vectors;
      if $\sum_{1 \le i \le n} u_{k-1}^{(i)} \le n-k$ **then** $u_k = 0$ **else** $u_k = 1$
end

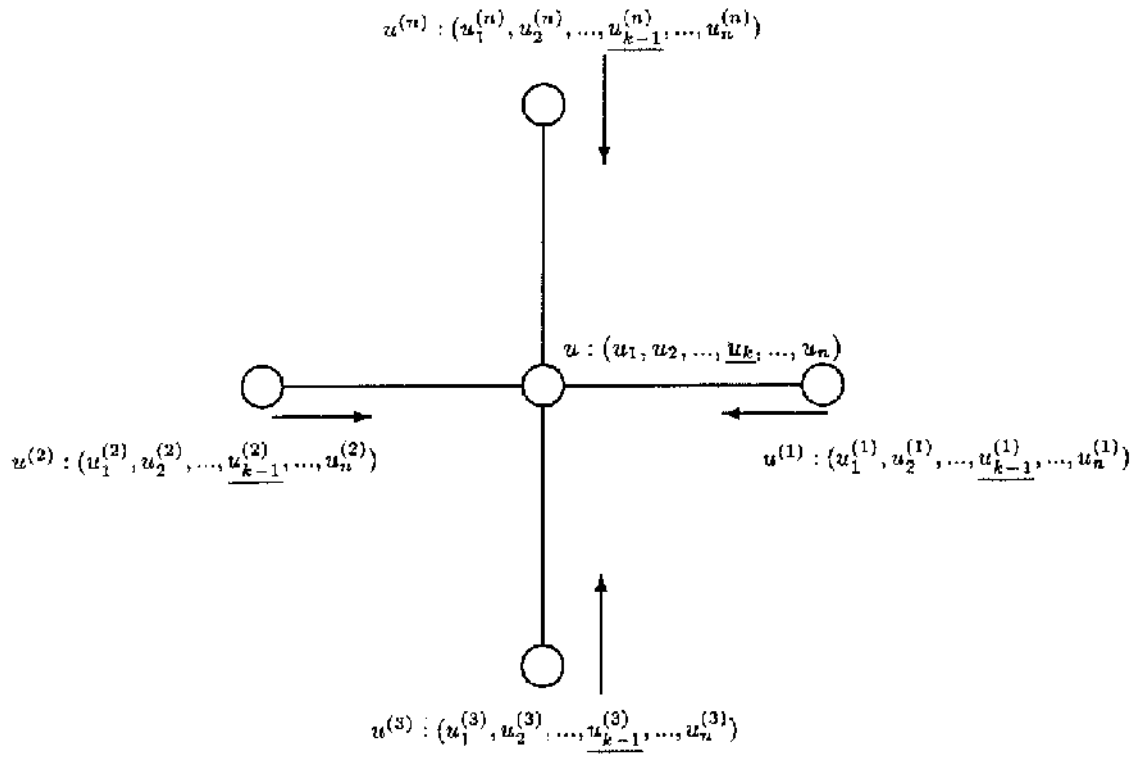

Fig. 2. Calculating safety vectors.

3.2 Calculation of Safety Vectors

The proof of Theorem 1 also suggests a simple way to determine the safety vector of each node in a given faulty hypercube. The GLOBAL_STATUS (GS) algorithm calculates the safety vector of each node in an n-cube. Instead of updating all the bits in a safety vector at each round of information exchange, we only need to update the kth bit at the kth round. As shown in Fig. 2, the kth bit, u_k, of node u's safety vector depends on the $(k-1)$th bit, $u_{k-1}^{(i)}$, of neighboring node $u^{(i)}$'s safety vector, where $1 \le i \le n$. We assume that all nonfaulty nodes have (1, 1, 1, ..., 1) as their initial safety vectors. By doing so, there is no need to execute GS in a fault-free hypercube. That is, our approach does not introduce additional overhead in a fault-free hypercube. As we will see later, the routing scheme based on safety vectors is a regular, fully adaptive minimal routing in a fault-free hypercube. All faulty nodes have (0, 0, 0, ..., 0) as their safety vectors. Note that each of two end nodes of a faulty link treats the other end node as a faulty node.

Table 1 shows the safety vectors obtained by applying GS to the four-cube of Fig. 1. In this case, three rounds are used. The safety vector (1, 1, 0, 1) associated with node 0000 indicates the existence of a minimal path to any other node, except for nodes that are three-Hamming-distance away. Bits (in safety vectors) that change at each round are underlined in the table.

3.3 Properties of Safety Vectors

We consider here properties of safety vectors that are useful for unicasting in a faulty hypercube.

THEOREM 2. *Given an n-cube (including disconnected one), the safety vector of each node can be determined through $n-1$ rounds of information exchange between neighboring nodes.*

Theorem 2 can be easily derived from Theorem 1 and the GS algorithm. The following theorem serves as a basis of the proposed routing algorithm to be discussed in the next section.

THEOREM 3. *Assume that $(u_1, u_2, ..., u_n)$ is the safety vector associated with node u in a faulty n-cube. If $u_k = 1$, then there exists at least one Hamming distance path from node u to any node which is exactly k-Hamming-distance away.*

Proof. We prove this theorem by induction on k. If $u_1 = 1$ (where $k = 1$), there is no adjacent faulty link. Clearly, node u can reach all the neighboring nodes, faulty and nonfaulty. Assume that this theorem holds for $k = l$, i.e., if $u_l = 1$, there exists at least one Hamming distance path from node u to any node which is exactly l-Hamming-distance away. When $k = l + 1$, if $u_k = 1$, then $\sum_{1 \le i \le n} u_l^{(i)} > n - (l+1)$, which means that there are at most l neighbors which have 0 at the kth bit of their safety vectors. Therefore, among $l + 1$ preferred neighbors, there is at least one neighbor, say node v that has its kth safety bit set. Based on the induction assumption, there is at least one Hamming distance path from node v to any destination node, say w, which is l-Hamming-distance away. Connecting the link from node u to node v to the path originated from node v to destination node w, we construct a Hamming distance path from node u to destination node w which is $(l + 1)$-Hamming-distance away. □

TABLE 1
THE CALCULATION OF THE SAFETY VECTOR OF EACH NODE IN THE FOUR-CUBE OF FIG. 1

(A) INITIAL SAFETY VECTOR ASSIGNMENTS

node address	0000	0001	0010	0011	0100	0101	0110	0111
safety vector	(1, 1, 1, 1)	(1, 1, 1, 1)	(1, 1, 1, 1)	(0, 0, 0, 0)	(0, 0, 0, 0)	(1, 1, 1, 1)	(0, 0, 0, 0)	(1, 1, 1, 1)
node address	1000	1001	1010	1011	1100	1101	1110	1111
safety vector	(1, 1, 1, 1)	(0, 0, 0, 0)	(1, 1, 1, 1)	(1, 1, 1, 1)	(1, 1, 1, 1)	(1, 1, 1, 1)	(1, 1, 1, 1)	(1, 1, 1, 1)

(B) SAFETY VECTORS AFTER THE FIRST ROUND

node address	0000	0001	0010	0011	0100	0101	0110	0111
safety vector	(1, 1, 1, 1)	(1, 0, 1, 1)	(1, 0, 1, 1)	(0, 0, 0, 0)	(0, 0, 0, 0)	(1, 1, 1, 1)	(0, 0, 0, 0)	(1, 0, 1, 1)
node address	1000	1001	1010	1011	1100	1101	1110	1111
safety vector	(1, 1, 1, 1)	(0, 0, 0, 0)	(1, 1, 1, 1)	(1, 0, 1, 1)	(1, 1, 1, 1)	(1, 1, 1, 1)	(1, 1, 1, 1)	(1, 1, 1, 1)

(C) SAFETY VECTORS AFTER THE SECOND ROUND

node address	0000	0001	0010	0011	0100	0101	0110	0111
safety vector	(1, 1, 0, 1)	(1, 0, 1, 1)	(1, 0, 1, 1)	(0, 0, 0, 0)	(0, 0, 0, 0)	(1, 1, 0, 1)	(0, 0, 0, 0)	(1, 0, 1, 1)
node address	1000	1001	1010	1011	1100	1101	1110	1111
safety vector	(1, 1, 1, 1)	(0, 0, 0, 0)	(1, 1, 1, 1)	(1, 0, 1, 1)	(1, 1, 1, 1)	(1, 1, 1, 1)	(1, 1, 1, 1)	(1, 1, 1, 1)

(D) SAFETY VECTORS AFTER THE THIRD (FINAL) ROUND

To relate the safety vector concept to the safety level concept proposed in [22], we first give the safety level definition and review its relevant properties.

DEFINITION 3 [22]. *The safety level of a faulty node is zero. For a nonfaulty node u, let $(S_0, S_1, S_2, ..., S_{n-1})$, $0 \le S_i \le n$, be the nondecreasing safety level sequence of node u's n neighboring nodes in an n-cube, such that $S_i \le S_{i+1}$, $0 \le i < n-1$. The safety level of node u, $S(u)$, is defined as: If $(S_0, S_1, S_2, ..., S_{n-1}) \ge (0, 1, 2, ..., n-1)$[1], then $S(u) = n$, else if $(S_0, S_1, S_2, ..., S_{k-1}) \ge (0, 1, 2, ..., k-1) \wedge (S_k = k-1)$ then $S(u) = k$.*

The safety level of a nonfaulty node is recursively defined in terms of its neighbors' safety levels. In a hypercube with faulty links, two end nodes of each faulty link are treated as faulty nodes. An n-safe node is called a *safe node* and all the other nodes (faulty and nonfaulty) are *unsafe*. A faulty node has a safety level of zero (zero-safe). An iterative algorithm similar to GS can be used to calculate the safety level of each node in an n-cube by assuming that all nonfaulty nodes are n-safe initially. It has been proven [22] that, for any faulty n-cube, $n-1$ rounds of information exchange are sufficient. Fig. 1 shows the safety level of each node (an integer inside each cycle) in a faulty four-cube as well as the corresponding safety vector. Based on the safety level definition, safety levels of nodes that have two (or more) faulty neighbors are changed to one after the first round, as in the case for nodes 0001, 0010, 0111, 1011. That is, the effect of zero-safe status of faulty nodes propagates first to their neighbors, then to the neighbors' neighbors, and so on. After the second round, for instance, the safety levels of nodes 0000 and 0101 are changed to two (as shown in Fig. 2), because each node has two one-safe neighbors and one faulty neighbor. The safety level of each node remains stable after two rounds and it represents the final safety level of the corresponding node. Note that throughout the calculation process, the safety level of each node is updated at most once. More specifically, a k-safe node's status is updated at the kth round.

There are several relevant properties [22] of safety level summarized as follows:

PROPERTY 1. *In an n-cube with no more than $n-1$ faulty nodes and no faulty link, each nonfaulty but unsafe node has a safe neighbor.*

PROPERTY 2. *If the safety level of a node is k $(0 < k \le n)$, then there is at least one Hamming distance path from this node to any node within k-Hamming-distance.*

Obviously, the safety vector model provides more information than the safety level model does. In Fig. 1, the safety vector associated with 0000 is (1, 1, 0, 1). However, node 0000 has a safety level two, which means that node 0000 can reach any node within two-Hamming-distance through a minimal path and it is not guaranteed that there exists a minimal path to any node which is three-Hamming-distance away (otherwise the safety level would be at least three). The safety level two (of node 0000) does not tell whether there exists a minimal path from node 0000 to any node which is four-Hamming-distance away.

1. $seq_1 \ge seq_2$ if and only if each element in seq_1 is larger than or equal to the corresponding element in seq_2.

The following theorem reveals the relationship between the safety vector and the safety level of a node in a given n-cube. For a given safety vector $(u_1, u_2, ..., u_n)$, we attach an extra bit $u_{n+1} = 0$.

THEOREM 4. *Assume that k is the safety level of node u in an n-cube and* $(u_1, u_2, ..., u_n, u_{n+1})$ *is the safety vector of node u, then*

$$k \le min\{i \mid u_{i+1} = 0\}$$

PROOF. We prove this theorem by induction on k, the safety level of node u. If $k = 0$, $k \le \min\{i \mid u_{i+1} = 0\}$ always holds true. When $k = 1$, there are at least two faulty neighbors, i.e., the first bit of both vectors associated with two faulty nodes is 0. Based on the definition of safety vector, the second bit of u's safety vector must be 0, i.e., $u_2 = 0$. Since node u is nonfaulty, $u_1 = 1$ based on the definition of safety vector. Assume that this theorem holds true for $k < l$; that is, if $S(u) = k < l$, $k \le \min\{i \mid u_{i+1} = 0\}$. The safety vector can be represented as $(\underbrace{1,1,\ldots,1}_{\ge k}0**,\ldots,*)$, where $*$ is a don't-care bit which can be either 0 or 1. When $S(u) = k = l$ and based on the safety level definition, the nondecreasing neighbor sequence $(S_0, S_1, S_2, ..., S_n)$ satisfies: $(S_0, S_1, S_2, ..., S_{l-1}) \ge (0, 1, 2, ..., l-1)$ and $S_l = l - 1$. Using the induction assumption, we have the following safety vectors of node u's neighbors:

$$S_0 \ge 0 \leftrightarrow (\underbrace{1,1,\ldots,1}_{\ge 0}0**,\ldots,*)$$

$$S_1 \ge 1 \leftrightarrow (\underbrace{1,1,\ldots,1}_{\ge 1}0**,\ldots,*)$$

$$\ldots \leftrightarrow \ldots$$

$$S_{l-1} \ge l-1 \leftrightarrow (\underbrace{1,1,\ldots,1}_{\ge l-1}0**,\ldots,*)$$

$$S_l \ge l-1 \leftrightarrow (\underbrace{1,1,\ldots,1}_{\ge l-1}0**,\ldots,*)$$

$$S_{l+1} \ge l-1 \leftrightarrow (\underbrace{1,1,\ldots,1}_{\ge l-1}0**,\ldots,*)$$

$$\ldots \leftrightarrow \ldots$$

$$S_{n-1} \ge l-1 \leftrightarrow (\underbrace{1,1,\ldots,1}_{\ge l-1}0**,\ldots,*)$$

By performing a bitwise addition of safety vectors of node u's neighbors, we have

$$(b_1(\ge n-1), b_2(\ge n-2), ..., b_{l-1}(\ge n-(l-1)),$$
$$b_l(\ge n-(l-1)), ..., b_n(\ge n-(l-1)).$$

Based on the safety vector definition, the safety vector of u is:

$$(u_1, u_2, \ldots, u_n) = \underbrace{1,1,\ldots,1}_{\ge l}0**,\ldots,*)$$

That is, $\min\{i \mid u_{i+1} = 0\} \ge l$. □

Clearly, if node u is n-safe under the safety level model, then the corresponding safety vector $(u_1, u_2, ..., u_n) = (1, 1, ..., 1)$. To show that the safety vector model is more powerful than the safety level model, we need to show at least one case such that $k < \min\{i \mid u_{i+1} = 0\}$. Let's consider a five-cube with seven faulty nodes: 01101, 01110, 10001, 10100, 10101, 11000, 11001. Based on the safety level definition, we have 3 as node 00000's safety level (five neighbors' safety levels are 2, 5, 2, 2, 1 along dimensions 1, 2, 3, 4, 5, respectively). Applying the GS algorithm, we have (1, 1, 1, 1, 1) as node 00000's safety vector, i.e., node 00000 is safe. Therefore, the safety vector model covers a larger set of safe nodes than the safety level model does.

THEOREM 5. *In an n-cube, assume that W is the number of faulty nodes and V is the number of faulty links. If* $W + 2V < n$, *then every nonfaulty node that does not have an adjacent faulty link has a safe neighbor.*

PROOF. Based on the definition of safety vector, each nonfaulty node that has a faulty adjacent link is assigned a vector (0, 1, 1, ..., 1). If we strengthen the condition by treating such a node as faulty, i.e., it has (0, 0, 0, ..., 0) as its safety vector, then the system is converted to the one with faulty nodes only. Because each faulty link has only two end nodes and $W + 2V < n$, the converted hypercube has fewer than n faulty nodes. Based on Property 2 of safety level, each nonfaulty node has at least one safe neighbor. Since $k \le \min\{i \mid u_{i+1} = 0\}$ based on Theorem 4, where $(u_1, u_2, ..., u_n)$ is the safety vector in the converted hypercube and the extra bit $u_{n+1} = 0$, when $k = n$ $(u_1, u_2, ..., u_n)$ is clearly (1, 1, ..., 1), i.e., safe. Again, because the safety vector (0, 1, 1, ..., 1) is strictly higher (in terms of degree of safety) than the safety vector (0, 0, 0, ..., 0), any safe node in the converted hypercube must be safe in the original hypercube. □

COROLLARY. *In an n-cube without link fault, assume that W is the number of faulty nodes. If* $W < n$, *then each nonfaulty node has a safe neighbor.*

Theorem 5 and its corollary will be used to access the applicability of the routing algorithm discussed in the next section.

4 Reliable Routing Using Safety Vectors

In this section, we propose an optimal routing algorithm and a suboptimal routing algorithm in faulty hypercubes based on the safety vector associated with each node. This scheme is an adaptive one in which each intermediate node (including the source node) routes a message adaptively based on its safety vector, its neighbors' safety vectors, and the relative distance to the destination node. Optimality is ensured when either the source node or one of its neighbors meets certain safety conditions decided by the Hamming distance between the source and destination nodes. Optimality can be decided at the source node. At each intermediate node (including the source node), the selection of a successor, among preferred neighbors, is based on the safety vectors of these neighbors.

4.1 Basic Idea

Suppose that source node s, with safety vector $(s_1, s_2, ..., s_n)$, intends to forward a message to a node which is k-Hamming-

distance away. The optimality is guaranteed if the kth bit of its safety vector is 1 ($s_k = 1$) or one of its preferred neighbors' (along dimension i) $(k-1)$th bit is 1, i.e., $s^{(i)}_{k-1} = 1$, $i \in \{1, 2, ..., n\}$. Routing starts by forwarding the message to a preferred neighbor where the $(k-1)$th bit of its safety vector is one, and this node, in turn, forwards the message to one of its preferred neighbors which has 1 in the $(k-2)$th bit of its safety vector, and so on. If the optimality condition fails but there exists a spare neighbor which has 1 in the $(k+1)$th bit of its safety vector, the message is first forwarded to this neighbor and, then, the optimal routing algorithm is applied. In this case, the length of the resultant path is the Hamming distance plus two. We call this result suboptimal.

The selection between the optimal and the suboptimal algorithms can be decided at the source node using the following information:

1) The Hamming distance of source (s) and destination (d) nodes. This can be done by computing $H(s, d) = |s \oplus d|$. The preferred neighbor set and the spare neighbor set are obtained based on $s \oplus d$.
2) The safety vector of source nodes: $(s_1, s_2, ..., s_n)$.
3) The safety vector of the neighboring node along the ith dimension: $\left(s^{(i)}_0, s^{(i)}_1, s^{(i)}_2, \ldots, s^{(i)}_n\right)$. $s^{(i)}_0$ is an extra bit introduced to simplify the routing algorithm. $s^{(i)}_0 = 1$ except for the case when s and $s^{(i)}$ are two end nodes of a faulty link.

In addition, a *navigation vector*, $N = s \oplus d$, is the relative address between the source and destination nodes. This vector is determined at the source node and it is passed to a selected neighbor after resetting or setting the corresponding bit of N. Upon receiving a routing message, each intermediate node first calculates its preferred and spare neighbors based on the navigation vector associated with the message. If this intermediate node is $(k+1)$-Hamming-distance away from the destination node (this distance can be determined based on the number of 1s in the navigation vector), a preferred neighbor which has 1 in the kth bit of its safety vector is selected. When a node receives a message with an empty navigation vector, it identifies itself as the destination node by terminating the routing process and keeping a copy of this message.

Note that, at the source node, if both conditions for optimal and suboptimal routing fail, the proposed algorithm cannot be applied. This failure state can be easily detected at source node. The cause of failure could be either too many faults in the neighborhood or a network partition. However, based on Theorem 5, when $W + 2V < n$ and the source node is nonfaulty with no adjacent faulty link, the proposed routing process never fails.

4.2 Routing Algorithms

The routing process consists of two parts: UNICASTING_AT_SOURCE_NODE is applied at the source node to decide the type of routing process and perform the first routing step. UNICASTING_AT_INTERMEDIATE_NODE is used at an intermediate node. After the first routing step, since both optimal and suboptimal routing algorithms select a Hamming distance path to route the message to the destination node, there is no need to distinguish the type of routing process in UNICASTING_AT_INTERMEDIATE_NODE.

Algorithm UNICASTING_AT_SOURCE_NODE
begin
  $N = s \oplus d$; $H = |s \oplus d|$;
  /* **calculate navigation vector N and Hamming distance H** */
  if $s_H = 1 \vee \exists i(s^{(i)}_{H-1} = 1 \wedge N(i) = 1)$
    /* **the Hth bit of the safety vector is one or the (H − 1)th bit of the safety vector of a preferred neighbor is one** */
  then OPTIMAL_UNICASTING:
    send $(m, N^{(i)})$ to $s^{(i)}$, where $s^{(i)}_{H-1} = 1$ and $N(i) = 1$
    /* **send message** m **to preferred neighbor** $s^{(i)}$**, where the (H − 1)th bit of its safety vector is one, together with N after resetting bit** i */
  else if $\exists i(s^{(i)}_{H+1} = 1 \wedge N(i) = 0)$
    /* **the (H + 1)th bit of a spare neighbor's safety vector is one** */
    then SUBOPTIMAL_UNICASTING:
      send $(m, N^{(i)})$ to $s^{(i)}$, where $s^{(i)}_{H+1} = 1$
      /* **send message** m **to spare neighbor** $s^{(i)}$ **where the (H + 1)th bit of the safety vector is one, together with N after setting bit** i */
    else failure
end.

Algorithm UNICASTING_AT_INTERMEDIATE_NODE
begin
  {at any intermediate node u with message m and navigation vector N}
  if $N = 0$ /* **the navigation vector is empty** */
  then stop /* **the currect node is the destination node** */
  else send $(m, N^{(i)})$ to $u^{(i)}$, where $u^{(i)}_{H-1} = 1$ and $N(i) = 1$
    /* **send message** m **to preferred neighbor** $u^{(i)}$**, where the (H − 1)th bit is one, together with N after resetting bit** i */
end.

The following theorem relates the length of a routing path, neighbors' safety vectors, and the Hamming distance between the source and destination nodes.

THEOREM 6. *Suppose that the Hamming distance between the source and destination nodes is k for a given unicasting. When the kth bit of the safety vector of the source node is* 1 *or there is a preferred neighbor of the source node which has* 1 *at the* $(k-1)$*th bit of its safety vector, optimality is guaranteed using the proposed routing process. When there is a spare neighbor of the source node which has* 1 *at the* $(k+1)$*th bit of its safety vector, then suboptimality is guaranteed.*

The proof of this theorem is straightforward based on Theorem 3 and the proposed routing process.

COROLLARY. *In the worst case, the length of any path generated from the proposed routing process is no more than* $n + 1$ *in an n-cube.*

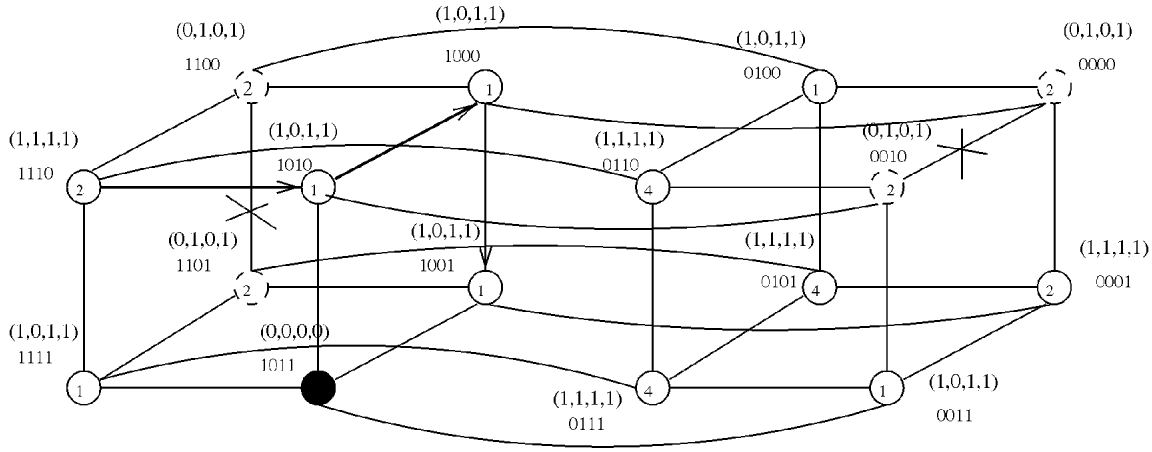

Fig. 3. A faulty four-cube with one faulty node and two faulty links.

Based on the above discussion, if the routing process is feasible, it is either optimal or suboptimal and it is independent of the number of faults in the n-cube. Note that the suboptimal routing generates a path of length $k + 2$ between two nodes that are k-Hamming-distance apart. For two nodes that are n-Hamming-distance apart, all the neighbors of the source node are preferred neighbors. Therefore, only optimal routing is used. The worst case occurs when the suboptimal routing algorithm is applied to two nodes that are $(n - 1)$-Hamming-distance apart. The length of the resultant routing path is $(n - 1) + 2 = n + 1$.

4.3 Examples

Consider the example of Fig. 1 with safety vectors, as shown in Table 1d. Suppose that the source node is 0001 with safety vector (1, 0, 1, 0), which means that a Hamming distance path is guaranteed for any destination node which is one- or three-Hamming-distance away. A Hamming distance path may or may not exist for a destination node which is two- or four-Hamming-distance away, depending on neighbors' safety vectors. For example, if the destination node is 1100, then $H = |1100 \oplus 0001| = |1101| = 3$. Therefore, a Hamming distance path exists. The source node adaptively selects a preferred neighbor which has 1 at the second bit of its safety vector. In this case, nodes 0101 and 0000 are both eligible. Assume that node 0000 is selected to which the routing message, together with $N = 1101 \oplus 0001 = 1100$, is forwarded. Node 0000 forwards the message to one of its preferred neighbors which has 1 at the first bit of its safety vector. Clearly, only node 1000 (the neighbor along dimension four) is eligible. Once node 1000 receives the message, together with $N = 0100$, it sends the message, together with $N = 0000$, to the destination node (the neighbor along three as indicated in $N = 0100$). The selected path from $s_1 = 0001$ to $d_1 = 1100$ is shown in Fig. 1.

To forward a message to a destination which is two-Hamming-distance away from source node 0001, node 0001 has to check if one of preferred neighbors has its first safety bit set to guarantee an optimal routing. For example, if the destination node is 1101, then it has two preferred neighbors: 0101 and 1001. Because the first bit of 0101's safety vector is 1, the message can be forwarded to 1001 via 0101. Note that each node has a copy of all its neighbors' safety vectors (a by-product of the GS algorithm), the optimality is decided at the source node without further information exchange between neighbors. If the destination node is 1011, optimal routing does not exist, since both preferred neighbors 1001 and 0011 (of source node 0001) are faulty. Fig. 1 also shows a possible path from source $s_2 = 1110$ to destination $d_2 = 0001$.

Fig. 3 shows the assignment of safety vectors and safety levels in a faulty four-cube with two faulty links (1100, 1101) and (0000, 0010) and one faulty node 1011. Node 1100's safety level is two, although it is considered faulty (zero-safe) by node 1101, which is the other end node of faulty link (1100, 1101). Note that the safety level of node 1110 is only two in Fig. 3, which means that optimality is not guaranteed to forward a message from 1110 to a node which is more than two-Hamming-distance away. Actually, to forward a message from 1110 to 1001, a suboptimal path (with a length of Hamming distance plus two) has to be used, because all of its preferred nodes' safety levels are lower than two. Using the safety vector model, node 1110 has a safety vector (1, 1, 1, 1), so there exists a Hamming distance path to any destination in the cube. For example, if 1001 is the destination node, source node 1110 sends the message to one of two preferred neighbors, 1100 or 1010, that has one at the second bit of its safety vector. Assume that node 1010 is selected which in turn forwards the message to one of its preferred neighbors which has one at the first bit of its safety vector. The path constructed is $1110 \rightarrow 1010 \rightarrow 1000 \rightarrow 1001$, as shown in Fig. 3.

The proposed algorithm can also be used in disconnected hypercubes. If the source node tries to send a message to a k-Hamming-distance destination node, which is in another (disconnected) subcube, it can detect such an infeasible routing based on its safety vector and neighbors'. More specifically, a k-Hamming-distance routing is infeasible if the kth bit of its safety vector is 0, the $(k - 1)$th bit of the safety vectors of all its preferred neighbors is 0, and the $(k + 1)$th bit of the safety vectors of all its spare neighbors is 0. Routing between two connected nodes in a disconnected hypercube will be treated the same as in a connected hypercube. Note that most existing routing algorithms do not address the routing issue in disconnected networks. Some existing algorithms may livelock or simply fail because the source may

initiate a routing process without knowing the destination is unreachable.

5 Discussion

5.1 Updating Safety Vectors

There are several ways to keep safety vector information up-to-date.

1) *Demand-driven*. The GS algorithm is applied only when a node detects an inaccurate safety vector (caused by occurrence or recovery of faults in the neighborhood) during a routing process. When a node recovers from a failure and becomes healthy, it will not cause disruption. However, when a new fault occurs, an on-going routing process might either continue without any effect or be blocked temporarily and rerouted from the current node after the update of safety vectors.
2) *Periodic*. Each node exchanges safety information periodically with its neighbors. Note that this approach does not adapt its activity to the failure rate of nodes and links. For example, all (or most) exchanges are wasted when all (or most) of nodes and links' status remain stable.
3) *State-change-driven*. A node initiates the GS algorithm whenever it detects a status change of a neighboring node or link. In this case, the GS algorithm can be implemented asynchronously as in the demand-driven approach.

The selection among different models depends on the application and the frequency of fault occurrence. Both demand-driven and state-change-driven approachs are passive in nature; that is, no update occurs when there is no status change. The periodic approach is active in nature in the sense that the safety status of each node is updated periodically even in the absence of new faults.

Let's use the demand-driven approach as an example. This approach is the most optimistic and, therefore, the least expensive to implement. To simply our discussion, we consider only node faults and use the four-cube in Fig. 1 as an example. When one or more new faults occur, an on-going routing process will not be affected unless one of the new faults is along the selected path. Even when one of the new faults blocks the selected path at an intermediate node, say node u, it may still use other available preferred neighbors. In this case, all preferred neighbors are blocked by faults, but there exists a safe spare neighbor (such a neighbor always exists provided the number of fault nodes is less than n in an n-cube). In either case, we have two possible ways to update safety vectors:

1) update after the completion of the on-going routing process, or
2) update immediately by blocking the on-going routing process at node u.

Because each routing process carries only the relative address (of the current and destination nodes), the routing process can resume immediately after the update of safety vectors (an application of the GS algorithm).

Consider a routing process $(s_2, d_2) = (1110, 0001)$ with message m_2 in the four-cube of Fig. 1. Assume that a new fault 0101 occurs and node 1001 recovers (and becomes healthy) when m_2 is at intermediate node 1101. Suppose that node 0001 detects a new fault at neighbor 0000; it immediately starts the GS algorithm and blocks the routing process (approach 2 is used). After GS is completed, the safety vectors associated with nodes 1001 and 0101 become (1, 1, 1, 1) and (1, 1, 0, 1), the routing process resumes at nodes 1101, and m_2 is forwarded to destination node 0001 via node 1001 (instead of node 0101).

The above approach still works when new faults occur in an update interval. If a new fault occurs during the ith round of GS, GS completes the update of the ith bit and will then update the $(i + 1)$th bit of each safety vector. If $i = 0$, this fault occurs before the update. If $i = n$, this fault occurs after the update. Both cases have been discussed earlier. If $1 \le i \le n - 1$, this fault occurs during the update process of GS. Clearly, all the bits before the ith bit have been updated before the occurrence of this new fault and, hence, they may or may not be accurate in representing fault distribution. The ones after the ith bit are more accurate, because the information about this new fault has been included in the calculation. (Note that such bits may still be inaccurate because they depend on other bits that may not be accurate.) To study the effect of a new fault on safety vectors of other nodes in the neighborhood, we have the following result:

THEOREM 7. *A new fault will not affect (by resetting) the first k bits in the safety vector of a node that is k-Hamming-distance away from this fault.*

The proof of this theorem is straightforward based on the following fact: A new fault will not affect a routing process if it is not along any minimal path between the source and destination nodes. That is, if the Hamming distance between the source and destination nodes is k, this new fault will not have any effect if it is at least k-Hamming-distance away from the source node. Clearly, a fault will have no effect on the node which is n-Hamming-distance away.

5.2 Adaptivity Analysis

To study the degree of adaptivity, let's define a *safety matrix* associated with each node u. An $n \times n$ safety matrix, A_u, is defined as a collection of neighbors' safety vectors in an n-cube, where the ith row of the safety matrix corresponds to the neighbor's safety vector along dimension i. For example, the safety matrix of node $u = 0001$ in the four-cube of Fig. 1 is the following:

$$A_u = \begin{pmatrix} 1 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{pmatrix}$$

To determine the adaptivity of intermediate node u (including the source node) with respect to destination node d, the safety matrix of u, A_u, is first derived. Because neighbors' safety vectors are all available after an application of GS, there is no need for a separate process to calculate A_u. We mask out rows that correspond to spare neighbors. The degree of adaptivity is defined as the summation of 1s in the $(H(u, d) - 1)$th column in the masked A_u. For example, suppose $u = 0001$ and $d = 1100$, $u \oplus d = 1101$ and $H(u, d) = 3$, i.e., the second row of A_u will be masked out (by setting all

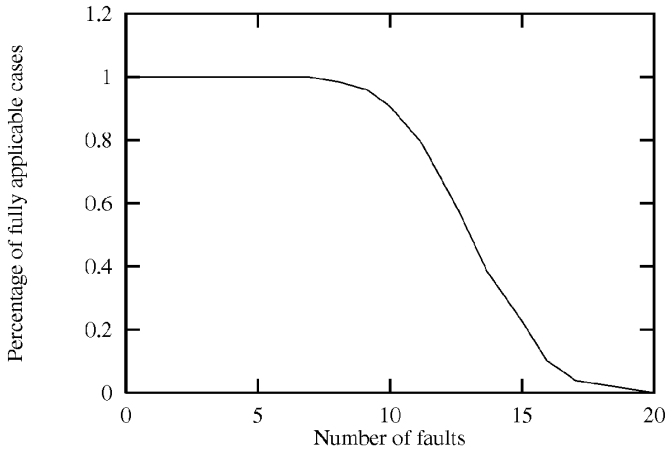

Fig. 4. Average percentage of fully applicable 7-cubes for different numbers of faults under different distributions of faults, source, and destination nodes.

bit values zero) and the number of 1s in the second column will be the degree of adaptivity. Intuitively, at intermediate node $u = 0001$, there are two choices of next node to forward a message to destination 1100. Note that at an intermediate node, any neighbor that satisfies the safety requirement (for optimal or suboptimal routing) is an eligible next forwarding node. However, we can select one with the highest safety degree. By doing so, we maximize the adaptivity and the probability of successful routing should more faults occur in the remaining routing process.

To determine the application range of the proposed scheme in terms of degree of fault tolerance, we consider the following three cases for a given number of faults and the dimension size:

1) *Fully applicable*. The proposed method can be applied to any combination of the source and destination nodes under any fault distribution.
2) *Partially applicable*. The proposed method can be applied to certain combinations of the source and destination nodes under certain fault distributions.
3) *Inapplicable*. The proposed method cannot be used in any combination of the source and destination nodes under any fault distribution.

Theorem 5 and its corollary show the upper bound of faults to ensure fully applicability under any distribution of faults, source, and destination nodes. For example, when the number of faulty nodes is less than n in an n-cube with no link fault, the proposed algorithm is fully applicable under any fault distribution. This is a very conservative approach, because certain fault distributions (which make the proposed method inapplicable) rarely occur. In other words, the proposed method is still fully applicable for certain cases (of fault distributions) even when the number of faults exceeds the upper bound. This situation can be modeled by the probability of fully applicable cases under certain fault distributions and this probability is a function of the number of faults. For example, when the number of faulty nodes reaches 2^{n-1} in an n-cube, the probability of fully applicable cases under certain fault distributions is not zero. Actually, it is no less than $n \Big/ \binom{2^n}{2^{n-1}}$. Because any faulty n-cube is still fully applicable if all 2^{n-1} faulty nodes reside in an $(n-1)$-cube. Clearly, there are n such assignments by selecting different dimensions to split an n-cube into two $(n-1)$-subcubes. Note that inapplicable cases are virtually nonexistent unless all the nodes in the n-cube are disconnected by faults, because any two connected nodes can still transfer messages between them. The full applicability of the routing process based on the safety vector model is captured in the following definition.

DEFINITION 4. *An n-cube is fully applicable if and only if, for every nonfaulty and unsafe node u, there exists at least one neighbor that is safe.*

In other words, if an n-cube is fully applicable, our scheme can be applied to any unicasting and obtain either an optimal or suboptimal result. A simulation has been conducted to obtain the average percentage of fully applicable seven-cubes for different numbers of faults under different distributions of faults, source, and destination nodes. Results are shown in Fig. 4. We can see that when the number of faults in seven-cubes reaches nine, the percentage is still close to one hundred. Note that when an n-cube is not fully applicable, it is still partially applicable, i.e., the proposed scheme is still applicable for certain unicasting. The applicability of the proposed routing scheme can be easily determined at each source node through a feasibility check.

5.3 Feasibility Check

Feasibility check determines the applicability of the proposed routing scheme. Actually, based on UNICASTING_AT_SOURCE_NODE, the checking process for a k-Hamming-distance routing can be simply expressed as:

Algorithm FEASIBILITY_CHECK
{at the source node}
if the $(k-1)$th bit of a preferred neighbor's safety vector is 1 or the $(k+1)$th bit of a spare neighbor's safety vector is 1
then it is feasible to use the proposed routing process
else it is infeasible.

6 EXTENSIONS

There are many variations of the hypercube topology proposed in the literature to improve selected, desirable properties. In general, the safety vector concept can be extended to other cube-based multicomputers provided certain properties of the hypercube topology are still maintained. Here, we illustrate our approach using generalized hypercubes. Extension to other cube-based architecture, such as cube-connected-cycles, is discussed in [21].

The generalized hypercube interconnection [2] is based on a mixed radix number system (as opposed to the binary system used in regular binary hypercubes) and this technique results in a variety of hypercube structures for a given number of processors, depending on the desired diameter of the network.

Let N be the total number of processors and be represented as a product of m_is, $m_i > 1$ for $1 \le i \le n$; that is, $N = m_n \times m_{n-1} \times ... \times m_1$. Each node corresponds to an n-vector address $(u_n, u_{n-1}, ..., u_1)$, where $0 \le u_i \le m_i - 1$. The *generalized n-dimensional hypercube* (GH_n) is defined as follows: Two nodes are linked by an edge if they differ in exactly one coordinate.

The safety vector of neighbors along a dimension, say i, of node u (analogous to the safety vector of the neighbors along this dimension in a regular hypercube) is a bitwise logical AND of all the neighbors' safety vectors ($m_i - 1$ in total) along this dimension, i.e., the bitwise minimum of safety vectors of all the nodes that have the same address as node u except at dimension i.

DEFINITION 4.

- *The safety vector of a faulty node is* (0, 0, ..., 0). *For a nonfaulty node u, if node u is an end node of a faulty link, then node u's safety vector is* (0, 0, ..., 0) *from the view of the other end node adjacent to the same faulty node.*
- $\left(u_1^{(i)}, u_2^{(i)}, \ldots, u_n^{(i)}\right)$ *is the bitwise minimum of safety vectors among all the nodes that have the same coordinates as node u except at dimension i.*
- *Base:*

$$u_1 = \begin{cases} 0 & \text{if } u \text{ is an end node of a faulty link} \\ 1 & \text{otherwise}. \end{cases}$$

- *Inductive steps*:

$$u_k = \begin{cases} 0 & \text{if } \sum_{1 \le i \le n} u_{k-1}^{(i)} \le n - k \\ 1 & \text{otherwise}. \end{cases}$$

The GLOBAL_STATUS (GS) algorithm can be easily extended for the generalized hypercube. The only change is the calculation of each bit in the safety vector. The bitwise minimum of safety vectors of all the nodes in the same dimension can be obtained in one step because all these nodes are completely connected. Therefore, the EXTENDED_GLOBAL_STATUS (EGS) algorithm requires a total of $2(n-1)$ rounds to obtain the safety status of each node in GH_n. This is because in the all-port model, where each node can send messages to and receive messages from all its neighbors in one step, bitwise minimization can be done in one step with all the relevant nodes directly connected. (Note that a round of message exchange followed by a local update is considered as one step.)

Algorithm EXTENDED_GLOBAL_STATUS (EGS)
begin
  forall $u \in GH_n$
    if u is an end node of a faulty link **then** $u_1 = 0$ **else** $u_1 = 1$;
  for $k = 2$ **step** 1 **to** n
    begin
      $u_{k-1}^{(i)} = \min\{a'_{k-1} \mid$ nodes u' and u differ only in the ith coordinate and $(u'_1, u'_2, \ldots, u'_n)$ is the safety vector of $u'\}$;
      forall $u \in GH_n$
        if $\sum_{1 \le i \le n} u_{k-1}^{(i)} \le n - k$ **then** $u_k = 0$ **else** $u_k = 1$
    end
end.

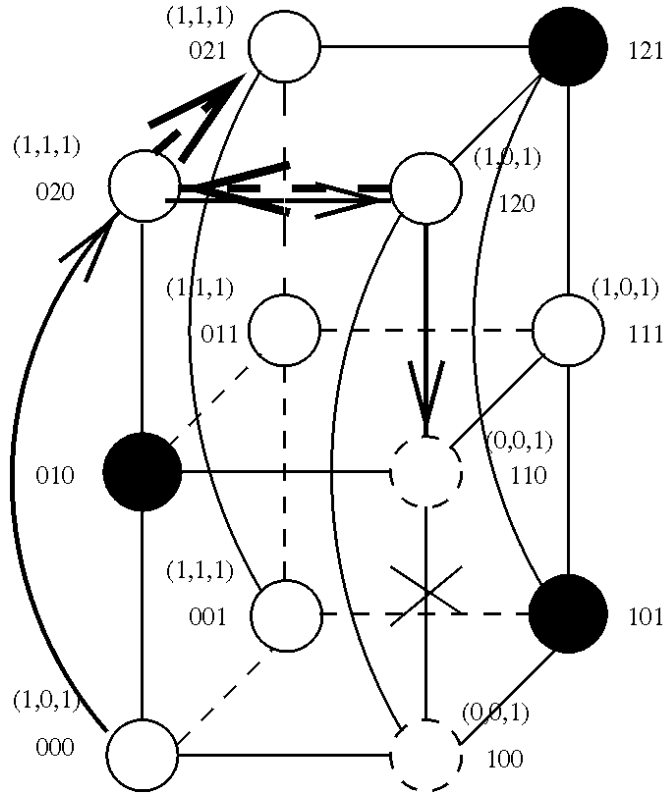

Fig. 5. A $2 \times 2 \times 3$ GH_3 with three faulty nodes and one faulty link.

Fig. 5 shows a $2 \times 2 \times 3$ GH_3 with three faulty nodes 121, 101, and 010 and one faulty link (100, 110). Applying the EGS algorithm to this GH_3, we obtain the safety vector for each node, as shown in Fig. 5.

The proposed routing process can be easily extended for the generalized hypercube. Dimensions are still classified as preferred and spare. Among neighbors ($m_k - 1$ in total) along a preferred dimension k, only one node has the same ith coordinate as the one in the destination node and this node is called a *preferred neighbor*. All the other neighbors along this preferred dimension are called *semi-preferred neighbors*. Any neighbors along a spare dimension are still called *spare neighbors*. For example, assume that the source node is 300 and the destination node is 321 in the $4 \times 3 \times 2$ GH_3 shown in Fig. 5. With respect to source node 300, dimension three is a spare one and dimensions two and one are preferred dimensions. Among neighbors along dimension two, node 310 is a semi-preferred neighbor and node 320 is a preferred one.

In the extended routing scheme, the selection of a neighbor is based on the following priority: preferred, semi-preferred, and spare. For example, if the source node is 120 and the destination node is 021, their distance is two. The safety vector of node 120 is (1, 0, 1) and its second bit is 0. Therefore, the optimal algorithm cannot be directly applied. However, node 020, one of the preferred neighbors, is safe. A minimal path can still be constructed, as shown in Fig. 5. Consider another routing example between source node 000 and destination node 110 separated by two-Hamming-distance. If the first bit of both preferred nodes 100 and 010 is zero, the optimal algorithm cannot be applied. The other two neighbors 020 (semi-preferred) and 001 (spare) are both safe

and node 020 has a higher priority. Once the routing message is forwarded to node 020, the optimal routing is followed. The length of the resultant path is the Hamming distance plus one (see Fig. 5). If a spare neighbor is selected, the length of the resultant path is the Hamming distance between the source and destination nodes plus two.

Theorem 3 in Section 3 still holds for generalized hypercubes. Note that in GH_n, two nodes are directed connected if and only if there addresses differ in exactly one coordinate. Therefore, the distance between two nodes are still measured by the number of different coordinates.

THEOREM 3'. *Assume that $(u_1, u_2, ..., u_n)$ is the safety vector associated with node u in any given GH_n. If $u_k = 1$, then there exists at least one minimal path from node u to any node which is exactly k-Hamming-distance away from node u.*

The proof of this theorem follows directly from the definition of GH_n and the proof of Theorem 3.

7 Conclusions

We have proposed an adaptive fault-tolerant routing scheme for cube-based multicomputers. This scheme uses limited global information captured by an n-bit safety vector associated with each node in an n-cube. Safety vectors can be calculated through a simple $(n - 1)$-round of information exchange among neighboring nodes. The source node can easily decide to perform either an optimal or suboptimal routing algorithm, based on its safety vector, its neighbors' safety vectors, and the Hamming distance between the source and destination nodes. It can also identify cases when minimal paths are blocked by faults and when a routing process tries to forward a message to a disconnected subcube. Possible extensions of the safety vector concept to other cube-based multicomputers have also been discussed.

The proposed routing scheme could be used together with other heuristic and/or greedy routing algorithms [12], [20], such as *randomized routing* and *depth-first routing*. Such a combination is especially efficient and useful in a system with many faults that result in a relatively small percentage of safe nodes. When safety degrees of the source node and its neighbors are too low to use the proposed routing scheme, a heuristic or greedy routing approach can be applied until an intermediate node with a sufficiently high safety status is reached. Then, the proposed routing scheme is used to guide the message to the destination node through a minimal path.

The safety vector concept can be potentially used to implement reliable collective communication. Collective communication operations, as defined by the *Message Passing Interface* standard [1], are fundamental in many applications and have been implemented in many parallel languages. They are used to distribute, gather, and exchange data, to perform global computation operation, and to perform processes synchronization. Examples of collective communication operations are multicasting, broadcasting, scatter, gather, all-to-all, complete exchange, reduction, and prefix sum [13].

The application of safety vectors in multicasting is straightforward. The key issue is how each intermediate node u forwards a set of destination nodes to its appropriate neighboring nodes. To solve this issue, the address summation, a bit-wise summation of all the destination nodes, is first calculated which represents the distribution of destination nodes along different dimensions. If a destination node can be forwarded to more than one preferred node, a priority among preferred neighbors is defined based on either safety degrees of preferred neighbors or values of address summation along these preferred dimensions. Safety-level-based multicasting and address-sum-based multicasting developed in [24] can be easily extended to safety-vector-based multicasting and address-sum-based multicasting, respectively. Because a safety vector normally offers more precise information than a safety level does, the corresponding multicast algorithm is expected to perform better. Note that in such an extension, the two end nodes of a faulty links should be treated as special faulty nodes (called *marked faulty nodes*) by all the other nodes. During a multicast process, no marked faulty node should be used as an intermediate node and, hence, no faulty links will be used in a resultant multicast tree. That is, all the marked faulty nodes will appear as leaves in the tree. Therefore, in the last step, the multicast message can still be forwarded to all these nodes. The use of safety vectors to implement other types of collective communication operations in a faulty cube-based multicomputer will be our future work.

Acknowledgments

The author gratefully acknowledges the three anonymous referees for providing helpful comments which improved the quality of the paper. A preliminary version of this paper appeared in the Proceedings of the 16th International Conference on Distributed Computing Systems.

References

[1] *MPI: A Message-Passing Interface Standard*. Message Passing Interface Forum, May 1994.
[2] L.N. Bhuyan and D.P. Agrawal, "Generalized Hypercube and Hyberbus Structures for a Computer Network," *IEEE Trans. Computers*, vol. 32, no. 4, pp. 323-333, Apr. 1984.
[3] Y.M. Boura and C.R. Das, "Fault-Tolerant Routing in Mesh Networks," *Proc. 1995 Int'l Conf. Parallel Processing*, pp. I 106-I 109, 1995.
[4] M.S. Chen and K.G. Shin, "Adaptive Fault-Tolerant Routing in Hypercube Multicomputers," *IEEE Trans. Computers*, vol. 39, no. 12, pp. 1,406-1,416, Dec. 1990.
[5] M.S. Chen and K.G. Shin, "Depth-First Search Approach for Fault-Tolerant Routing in Hypercube Multicomputers," *IEEE Trans. Parallel and Distributed Systems*, vol. 1, no. 2, pp. 152-159, Apr. 1990.
[6] G.M. Chiu and S.P. Wu, "A Fault-Tolerant Routing Strategy in Hypbercube Multicomputers," *IEEE Trans. Computers*, vol. 45, no. 2, pp. 143-156, Feb. 1996.
[7] *NCUBE 6400 Processor Manual*. NCUBE Company, 1990.
[8] P.T. Gaughan and S. Yalamanchili, "Adaptive Routing Protocols for Hypercube Interconnection Networks," *Computer*, vol. 26, no. 5, pp. 12-24, May 1993.
[9] J.M. Gordon and Q.F. Stout, "Hypercube Message Routing in the Presence of Faults," *Proc. Third Conf. Hypercube Concurrent Computers and Applications*, pp. 251-263, Jan. 1988.

[10] Y. Lan, "A Fault-Tolerant Routing Algorithm in Hypercubes," *Proc. 1994 Int'l Conf. Parallel Processing*, pp. II 163-II 166, Aug. 1994.
[11] T.C. Lee and J.P. Hayes, "A Fault-Tolerant Communication Scheme for Hypercube Computers," *IEEE Trans. Computers*, vol. 41, no. 10, pp. 1,242-1,256, Oct. 1992.
[12] F.T. Leighton, *Introduction to Parallel Algorithms and Architectures: Arrays, Trees, Hypercubes*. Morgan Kaufmann, 1992.
[13] P.K. McKinley, Y.J. Tasi, and D.F. Robinson, "Collective Communication in Wormhole-Routed Massively Parallel Computers," *Computer*, vol. 28, no. 12, pp. 39-50, Dec. 1995.
[14] L.M. Ni and P.K. McKinley, "A Survey of Routing Techniques in Wormhole Networks," *Computer*, vol. 26, no. 2, pp. 62-76, Feb. 1993.
[15] F.P. Preparata and J. Vuillemin, "The Cube-Connected Cycles, a Versatile Network for Parallel Computation," *Comm. ACM*, pp. 30-39, May 1981.
[16] C.S. Raghavendra, P.J. Yang, and S.B. Tien, "Free Dimension–An Effective Approach to Achieving Fault Tolerance in Hypbercubes," *Proc. 22nd Int'l Symp. Fault-Tolerant Computing*, pp. 170-177, 1992.
[17] J. Rattler, "Concurrent Processing: A New Direction in Scientific Computing," *Proc. AFIPS Conf.*, vol. 54, pp. 157-166, 1985.
[18] Y. Saad and M.H. Schultz, "Data Communication in Hypercubes," Technical Report YALEU/DCS/RR-428, Dept. of Computer Science, Yale Univ., June 1985.
[19] H. Sullivan, T. Bashkow, and D. Klappholz, "A Large Scale, Homogeneous, Fully Distributed Parallel Machine," *Proc. Fourth Ann. Symp. Computer Architecture*, pp. 105-124, Mar. 1977.
[20] L. Valiant, "A Scheme for Fast Parallel Communication," *SIAM J. Computing*, vol. 34, no. 1, pp. 350-361, May 1982.
[21] J. Wu, "Reliable Communication in Cube-Based Multicomputers Using Safety Vectors," Technical Report TR-CSE-95-24, Dept. of Computer Science and Eng., Florida Atlantic Univ., Apr. 1995.
[22] J. Wu, "Unicasting in Faulty Hypercubes Using Safety Levels," *IEEE Trans. Computers*, vol. 46, no. 2, pp. 241-247, Feb. 1997.
[23] J. Wu and E.B. Fernandez, "Broadcasting in Faulty Hypercubes," *Proc. 11th Symp. Reliable Distributed Systems*, pp. 122-129, Oct. 1992.
[24] J. Wu and K. Yao, "Fault-Tolerant Multicasting in Hypercubes Using Limited Global Information," *IEEE Trans. Computers*, vol. 44, no. 9, pp. 1,162-1,166, Sept. 1995.

Jie Wu received the BS degree in computer engineering in 1982 and the MS degree in computer science in 1985, both from the Shanghai University of Science and Technology, and the PhD degree in computer engineering from Florida Atlantic University, Boca Raton, Florida, in 1989. During 1985-1987, he taught at the Shanghai University of Science and Technology. Since August 1989, he has been with the Department of Computer Science and Engineering at Florida Atlantic University, where he is an associate professor and the director of computer science and engineering graduate programs. Dr. Wu has published in *IEEE Transactions on Software Engineering*, *IEEE Transactions on Computers*, *IEEE Transactions on Parallel and Distributed Systems*, *Journal of Parallel and Distributed Computing*, and *Concurrency: Practice and Experience*. His research interests are in the area of fault-tolerant computing, interconnection networks, Petri net applications, and software engineering.

Dr. Wu serves on the editorial board of the *International Journal on Computers and Applications*. He served on the program committees of the 1996 and 1998 IEEE International Conference on Distributed Computing Systems. He will be the program cochair of the 12th ISCA International Conference on Parallel and Distributed Computing Systems in 1999 and the local arrangements chair of the Fifth IEEE International Symposium on High-Performance Computer Architecture in 1999. Dr. Wu is also a co-guest editor of a special issue of *IEEE Transactions on Parallel and Distributed Systems* on "Challenges in Designing Fault-Tolerant Routing in Networks." He is a recipient of the 1996-1997 Researcher of the Year Award at Florida Atlantic University. Dr. Wu is a senior member of the IEEE and a member of the ACM.