Optimal Fault-Tolerant Routing in Hypercubes Using Extended Safety Vectors

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

Feng Gao, Zhongcheng Li, and Yinghua Min
Center for Fault-Tolerant Computing, CAD Lab
Institute of Computing Technology, Chinese Academy of Sciences
Beijing, P.R. China

Abstract

[1] *Reliable communication in cube-based multicomputers using the extended safety vector concept is studied in this paper. In our approach, each node in a cube-based multicomputer of dimension n is associated with an extended safety vector of n bits, which is an approximated measure of the number and distribution of faults in the neighborhood. In the extended safety vector model, each node knows fault information within distance-2 and fault information outside distance-2 is coded in a special way based on the coded information of its neighbors. The extended safety vector of each node can be easily calculated through $n-1$ rounds of information exchanges 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. In addition, the extended safety vector can be used as a navigation tool to direct a message to its destination through a minimal path. Simulation results show a significant improvement in terms of optimal routing capability in a hypercube with faulty links using the proposed model, compared with the one using the original safety vector model.*

1. Introduction

Many recent experimental and commercial multicomputers use direct-connected networks with the grid topology. The binary hypercube [8] is one of the popular grid structures. Several research prototypes and systems have been built in the past two decades, including NCUBE-2, Intel iPSC, and the Connection Machine. The more recently built SGI Origin 2000 uses a variation of the hypercube topologies.

Efficient interprocessor communication is a key to the performance of a multicomputer. *Unicasting* is a one-to-one communication between two nodes, one is called the source node and the other the destination node. With the rapid progress in VLSI and hardware technologies, the size of computer systems has increased tremendously and the probability of processor failure has also increased. As a result, building a reliable multicomputer has become one of the central issues, especially in the communication subsystem which handles all interprocessor communications. Among different routing (unicast) schemes, the classical e-cube routing is simple to implement and provides high throughput for uniform traffic; however, it cannot handle even simple node or link faults due to its nonadaptive routing. Adaptive and fault-tolerant routing protocols have been the subject of extensive research in recent years ([3], [4], [7]). A general theory of fault-tolerant routing is discussed in [2].

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, an approach of this type can be more cost effective than the ones based on global information [9] or local information ([1], [5]).

One simple but ineffective approach is to use distance-k information in which each node knows the status of all components within Hamming-distance-k (or simple distance-k). However, optimality cannot be guaranteed, as a routing process could possibly go to either a state where all mini-

[1]This work was supported in part by NSFC of P. R. China under grant No. 69703001 and by NSF grant CCR 9900646.

mal 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 distance-k information.

Another approach is based on the *coded fault information*, where each node has the exact information of adjacent nodes and information of other nodes are coded in a special way. Then an optimal/suboptimal routing algorithm is proposed based on the coded information associated with each node. The following is a summary of different coding methods in an n-cube, all of them are primarily designed to cover node faults.

- Lee and Hayes' [6] *safe* and *unsafe* node concept. A nonfaulty node is unsafe if and only if there are at least two unsafe or faulty neighbors. Therefore, each node is labeled (coded) faulty, unsafe, or safe.

- Wu and Fernandez' [12] *extended safe node* concept by relaxing certain conditions of Lee and Hayes' definition. Each node is still labeled faulty, unsafe, or safe. However, a different definition is given: A nonfaulty node is unsafe if and only if there are two faulty neighbors or there are at least three unsafe or faulty neighbors.

- Wu's *safety level* [11] concept where each node is assigned with a safety level k, $0 \leq k \leq n$. A node with a safety level $k = n$ is called safe and a faulty node is assigned with the lowest level 0. Therefore, there are $n + 1$ possible labels for a node in the safety level model.

- Wu's *safety vector* [10] concept where each node is associated with a binary vector. The bit value of the kth bit corresponds to the routing capability to nodes that are distance-k away. The safety vector is a refinement of the safety level model.

The effectiveness of a coding method is measured by the following: (1) How fast fault information can be collected (coded) at each node. (2) How accurate the coded fault information representing the real fault distribution in terms of optimal routing capability.

Both safe/unsafe and extended safe/unsafe models require $O(n^2)$ rounds of information exchanges in an n-cube to label (code) all the nodes. Both safety level and safety vector need only $O(n)$ rounds of information exchanges. The order, in terms of accurately representing fault information, is the following: safe/unsafe, extended safe/unsafe, safety level, and safety vector. The safety vector is the latest model that has a merit of simplicity and wide-range of fault coverage. However, this model is still relatively inefficient in handling of link faults. Basically, a link fault is considered by other nodes as node fault(s) by treating two end nodes of the link faulty. Each end node of a faulty link treats the other one faulty, but it does not consider itself faulty. This overly conservative approach generates many faulty nodes that severely diminishes the routing capability of the system.

In this paper, we propose a new coding methodology. It is assumed that each node has precise information of fault distribution within a given distance d. Fault information outside distance d is coded, like the one used in safety vector. We select $d = 2$ as an example and the corresponding model is called *extended safety vector*. Simulation results show a significant improvement using the proposed model in terms of optimal routing capability in a hypercube with faulty links, compared with the one using the original safety vector model. We also show that the selection of $d = 2$ is a right one and its results stay very close to the one using global fault information, i.e., $d = n$.

2. Preliminaries

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 a link 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 $u(i) \in \{0,1\}$, $1 \leq i \leq n$, 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, $1101^{(3)} = 1001$. 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 connected depending on the number and location of faults. A path connecting two nodes s and d is called 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, the above 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 ob-

tained by using links at each of these $H(s,d)$ preferred dimensions in some order.

Safety Level and Safety Vector. Let us first review the concepts of safety level and safety vector. Safety level and safety vector are scalar and vector numbers associated with each node in a given n-cube, respectively. They provide coded information about fault information in the neighborhood.

Definition 1 [11]: *The safety level of a faulty node is* 0. *For a nonfaulty node* u, *let* $(sl_0, sl_1, sl_2, ..., sl_{n-1})$, $0 \leq sl_i \leq n$, *be the nondescending safety level sequence of node* u*'s* n *neighboring nodes in an n-cube, such that* $sl_i \leq sl_{i+1}$, $0 \leq i < n-1$. *The safety level of node* u, $sl(u)$, *is defined as: if* $(sl_0, sl_1, sl_2, ..., sl_{n-1}) \geq (0, 1, 2, ..., n-1)$[2], *then* $sl(u) = n$, *else if* $(sl_0, sl_1, sl_2, ..., sl_{k-1}) \geq (0, 1, 2, ..., k-1) \wedge (sl_k = k-1)$ *then* $sl(u) = k$.

In the above definition, it is assumed that all faults are node faults. To extend this definition to cover link faults, both end nodes of a faulty link have to be assigned a safety level of 0 in order to be consistent with the original safety level definition. The safety vector concept is a refinement of the safety level concept by providing routing capability to destinations at different distances. More specifically, each node u in an n-cube is assigned with a safety vector $sv(u) = (u_1, u_2, ..., u_n)$.

Definition 2 [10]: *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* k*th bit:*

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

In the safety level (vector) model, a node in an n-cube is said to be *safe* if its safety level is n $((1, 1, ..., 1))$; otherwise, it is *unsafe*. Two properties related to safety levels and safety vectors are as follows:

Property 1: *If the safety level of a node is* k $(0 < k \leq n)$, *then there is at least one Hamming distance path from this node to any node within distance-*k.

Property 2: *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 that is exactly distance-*k *away.*

[2] $seq_1 \geq seq_2$ if and only if each element in seq_1 is larger than or equal to the corresponding element in seq_2.

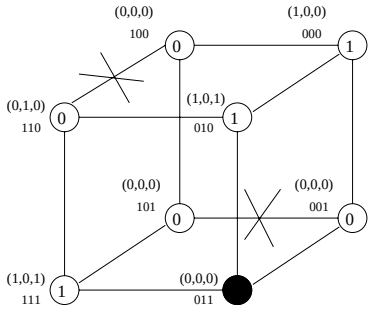

Figure 1. An example of a faulty 3-cube with its safety level and safety vector assignments.

Based on the above two properties, it is clear that a safe node in both models can reach any destination node (which is within Hamming distance n, the diameter of the cube) through a minimal path. The safety vector model is an improvement based on the safety level model. It can provide more and accurate information about the number and distribution of faults in an n-cube. However, the safety vector concept still cannot effectively present faulty link information. Actually, it is not clear that an efficient coding method exists under the assumption that each node has only neighbor information.

Fig. 1 shows an example of a 3-cube with one faulty node and two faulty links. In this example, the safety level of each node is either 0 or 1, i.e., a source node can only send a message to its neighbors. Clearly, by inspection, the safety level information is not accurate. For example, nodes 110, 101, and 111 can send a message to any nodes through a minimal path that are distance-2 or -3 away. This problem is partially resolved in the safety vector model, where the safety vectors associated with nodes 110 and 111 are (0,1,0) and (1,0,1), respectively. Nodes 001, 100, and 101 still have the lowest safety level (0,0,0). The reason that node 101 has a 0-bit at the 2nd bit of its safety vector is that it has two neighbors 100 and 001 with both 0-bit as the 1st bit of their safety vectors. However, the two corresponding faulty links (100, 110) and (101, 001) do not span on the same dimension, and hence, these two faulty nodes will not block all the minimal paths initiated from node 101 to a node that is distance-2 away.

Based on the above analysis, the direction of each fault (especially link fault) is needed to provide accurate information about fault distribution. However, this approach will dramatically increase the memory space requirement and the coding complexity. A compromise is therefore needed.

3. Extended Safety Vectors

Proposed Model. In this section, we propose a new approach to code fault information. In general, a good coding method is generated on the soundness of its base. In all existing approaches, the base is based on neighbor information only. For both safety level and safety vector models, the above method is proved to be effective for node faults, but not for link faults. Our approach here consists of the following two steps (see Fig. 2):

1. Each node knows the exact fault information within distance-d.
2. Fault information about nodes that are outside distance-d is coded in a special way.

In this paper, we show an application of the proposed model on $d = 2$; that is, each node knows fault information within distance-2. Information about faults that are more than distance-2 away are coded. We show that $d = 2$ is sufficient to handle link faults and there is no need to select a large d (this will be confirmed by our simulation results later). This model is called *extended safety vector*, where the first two bits of an extended safety vector (for a node) represent accurate fault information and other bits k are coded based on the $(k-1)$th bit of its neighbors'. Note that the regular safety vector model is a special case of this approach where $d = 1$. Results of our simulation show a dramatic improvement of this approach over the safety vector model in handling link faults.

Extended Safety Vectors. Let $esv(u) = (u_1, u_2, ..., u_n)$ be the extended safety vector of node u and $esv(u) = (u_1^{(i)}, u_2^{(i)}, ..., u_n^{(i)})$ be the extended safety vector of node $u^{(i)}$, u's neighbor along dimension i. We have the following inductive definition of extended safety vector $esv(u)$.

Definition 3: *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:* $u_1 = 1$ *if node* u *can reach any neighbor, i.e.,*

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

- *Base for the second bit:* $u_2 = 1$ *if node* u *can reach any nonfaulty and faulty nodes that are two hops away through a minimal path (a decision process will be discussed later); otherwise,* $u_2 = 0$.
- *Inductive definition for the* k*th bit, where* $k \geq 3$*:*

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

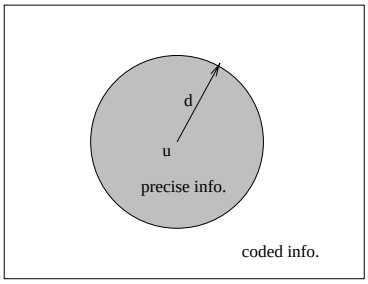

Figure 2. A new approach to code information.

In the extended safety vector model, in addition to the information of adjacent links and nodes, each node has complete information about adjacent links of its neighbors. $u_1 = 1$ (or $u_2 = 1$) indicates the existence of a minimal path to a nonfaulty nodes that are one hop (or two hops) away. For the case of $u_1 = 0$ (or $u_2 = 0$), such a minimal path may or may not exist, but for a given source-destination pair (that are one or two hops away) its existence can be easily verified based on distance-2 information. We use coded information for destinations that are more than two hops away. Therefore, for the case of $u_k = 0$, with $k > 2$, the actual existence of a minimal path for a given source-destination pair cannot be verified, that is, $u_k = 1$ provides a sufficient condition for the existence of a minimal path to a distance-k node, but it is not a necessary condition.

To determine u_2, node u needs to keep *faulty paths of length 2* initiated from node u, i.e., a path along which there exists at least one faulty link or node. A faulty path $(u, u^{(i)}, (u^{(i)})^{(j)})$ can be simply represented by a dimension sequence (i, j). Clearly, an adjacent faulty link or node along dimension i can be represented as (i, c_i), where $c_i = \{1, 2, ..., n\} - \{i\}$. That is, any path $(u, u^{(i)}, (u^{(i)})^{(k)})$, where $k \in c_i$, is a faulty path of length 2. If both adjacent link and node along dimension i are healthy, but there are adjacent faulty links along dimensions in $c_i^{'}$, where $c_i^{'}$ is a subset (including empty set) of c_i, the corresponding faulty paths can be represented as $(i, c_i^{'})$. Note that information about $(i, c_i^{'})$ is passed from node $u^{(i)}$ to node u. In general, each node u in an n-cube has exactly n pairs of $(i, c_i^{'})$, denoted as $f(u) = \{(i, c_i^{'}) : i \in \{1, 2, ..., n\}\}$. Some $(i, c_i^{'})$'s correspond to healthy paths, where $c_i^{'} = \{\}$ is an empty set.

Theorem 1: $u_2 = 0$ *for node* u *if and only if there exist* $(i, c_i^{'})$ *and* $(j, c_j^{'})$ *in* $f(u)$ *such that* $\{i\} \cap c_j^{'} \neq \phi$ *and* $\{j\} \cap c_i^{'} \neq \phi$, *where* $\phi = \{\}$ *is an empty set.*

Proof: There are two node-disjoint paths from node u to another node v that is distance-2 away. Suppose these two nodes "span" on dimensions i and j. Clearly, a path of length 2 from node u to node v is (i, j) and another is (j, i).

Calculating extended safety vectors:

1. In the first round, node u determines u_1 based on the status of its adjacent links, and then, exchanges adjacent link and node status with all its neighbors'.
2. In the second round, node u constructs $f(u)$ which is a list of faulty paths of length 2 based on the information collected in the first round. u_2 is determined based on $f(u)$, and then, exchanges u_2 with $u_2^{'}$s of all its neighbors.
3. In the kth round ($3 \leq k < n$), node u determines u_k based on u_{k-1}'s collected in the previous round, and then, exchanges u_k with $u_k^{'}$s of all its neighbors.
4. In the nth round, node u determines u_n based on u_{n-1}'s collected in the previous round.

u cannot reach v if and only if (i, j) is in $(i, c_i^{'})$ (i.e., that path is faulty) and (j, i) is in $(j, c_j^{'})$. In this case, we have $\{i\} \cap c_j^{'} \neq \phi$ and $\{j\} \cap c_i^{'} \neq \phi$. ■

In the example of Fig. 1, if $u = 111$, then $f(u) = \{(1, \{2\}), (2, \{3\}), (3, \{1, 2\})\}$. Based on the above theorem, the second bit u_2 of the safety vector associated with node 111 is 1. Clearly, bits u_1 and u_2 can be determined through 2 rounds of information exchanges among neighboring nodes and u_3, u_4, ..., u_n each needs one round. Therefore, the extended safety vector $(u_1, u_2, ..., u_n)$ of node u in an n-cube can be calculated through $n-1$ rounds of information exchanges among neighboring nodes.

Theorem 2: *The extended safety vector of each node in an n-cube can be determined through $n-1$ rounds of information exchanges between adjacent nodes.*

4. Examples and Properties

Fig. 3 shows an example of a faulty 4-cube with two faulty nodes 0001 and 1011 and two faulty links (0000, 0010) and (1100, 1101). The safety vector of each node is shown under both the safety vector and extended safety vector (on top) models. The safety level of each node is placed inside each node. Nodes 0001 and 1011 have an extended safety vector $(0, 0, 0, 0)$, nodes 0010, 1100, and 1101 have an extended safety vector $(0, 1, 1, 1)$, 0011 has (1,0,1,1) and 0000 has (0,0,1,1), and the remaining nodes have a safety vector $(1, 1, 1, 1)$.

In the following we show several properties related to extended safety vectors.

Theorem 3: *Assume that $(u_1, u_2, ..., u_n)$ is the extended 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 distance-k away.*

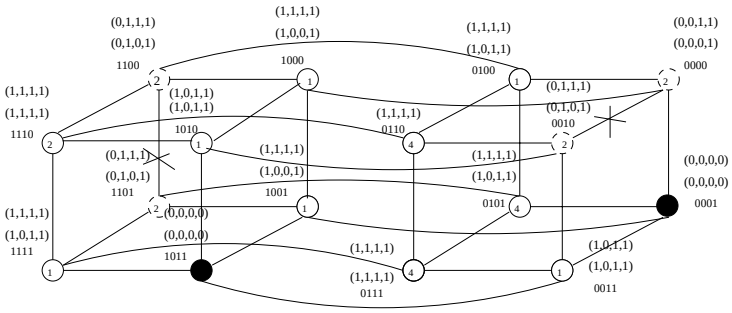

Figure 3. A faulty 4-cube with two faulty nodes and two faulty links.

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. If $u_2 = 1$ (where $k = 2$), based on the definition of u_2, there is a minimal path to any destination that is distance-2 away from the source. 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 distance-l away. When $k = l + 1$, if $u_{l+1} = 1$ then $\sum_{1 \leq i \leq n} u_l^{(i)} > n - (l + 1)$, which means that there are at most l neighbors which have 0 at the lth bit of their safety vectors. Therefore, among $l + 1$ preferred neighbors, there is at least one neighbor, say node v, that has its lth 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 distance-l 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 distance-$(l + 1)$ away. ■

Next we show that the extended safety vector is better than the regular safety vector in terms of accurately representing fault information (Figure 4 is such an example). Consider a vertex u in an n-cube with safety vector $sv(u) = (u_1, u_2, ..., u_n)$ and extended safety vector $esv(u) = (u_1^{'}, u_2^{'}, ..., u_n^{'})$, the extended safety vector is said to *cover* the safety vector at node u, if $u_k^{'} \geq u_k$ for all $1 \leq k \leq n$. Intuitively, if $ev(u)$ covers $v(u)$ at the kth bit, then the routing based on the extended safety vector has at least the same routing capability as one based on the safety vector to all destinations that are distance-k away.

Theorem 4: *For any given faulty n-cube, $esv(u)$ covers $sv(u)$ for any node u in the cube.*

Proof: We assume that a general node in a given cube is represented as u, with $esv(u) = (u_1^{'}, u_2^{'}, ..., u_n^{'})$ and $sv(u) = (u_1, u_2, ..., u_n)$ as extended safety vector and safety vector, respectively. We prove the theorem by induction on k in bit u_k for all nodes in the cube. When $k = 1$, u_1 has the same definition as $u_1^{'}$ for all nodes. Clearly, $u_1^{'} \leq u_1$. When $k = 2$, based on Property 2, $u_2 = 1$ means that there is a minimal path to any node that is distance-2 away. On the other hand, $u_2^{'} = 1$ if and only if there is a minimal path to any node that is distance-2 away. Hence, if $u_2 = 1$ then $u_2^{'} = 1$. The reverse condition normally does not hold. Therefore, $u_2^{'}$ covers u_2. Assume that the theorem holds for $k = l > 2$, i.e., $u_l^{'(i)} \geq u_l^{(i)}$ for all l. When $k = l + 1$, $u_{l+1} = 1$ if $\sum_{1 \leq i \leq n} u_l^{(i)} > n-(l+1)$ and $u_{l+1}^{'} = 1$ if $\sum_{1 \leq i \leq n} u_l^{'(i)} > n - (l + 1)$. Based on the fact that $u_l^{'(i)} \geq u_l^{(i)}$ for all i, where $i \in \{1, 2, ..., n\}$, $\sum_{1 \leq i \leq n} u_l^{'(i)} > n-(l+1)$ implies $\sum_{1 \leq i \leq n} u_l^{(i)} > n-(l+1)$, i.e., $u_{l+1}^{'} = 1$ implies $u_{l+1} = 1$. ■

5. Fault-Tolerant Routing

Basic Idea. The routing algorithm is similar to the one in [10]. Suppose that source node s, with safety vector $(s_1, s_2, ..., s_n)$, intends to forward a message to a node that is distance-k away. $(s_1^{(i)}, s_2^{(i)}, ..., s_n^{(i)})$ is the safety vector of neighbor $s^{(i)}$. 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., $s_{k-1}^{(i)} = 1$, $1 \leq i \leq n$. 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 that 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 that has one 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.

Routing Algorithms. The routing process consists of two parts: *unicasting_at_source_node* is applied at the source node to decide the type of the routing algorithm and to perform the first routing step. *unicasting_at_intermediate_node* is used at an intermediate node. In the proposed routing process, a *navigation vector*, $N = s \oplus d$, is used which 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 (using $H^{(i)}$) the corresponding bit i in 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 distance-$(k+1)$ away from the destination node (this distance can be determined based on the number of 1's 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 by 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 the source node. The cause of failure could be either too many faults in the neighborhood or a network partition.

Algorithm unicasting_at_source_node
begin
  $N = s \oplus d$; $k = |s \oplus d|$;
  if $s_k = 1 \vee \exists i(s_{k-1}^{(i)} = 1 \wedge N(i) = 1) \vee$
    ($k = 1$ (or 2) $\wedge$ a healthy 1-hop (or 2-hop) path
     along dimension i exists)
  then optimal_unicasting:
      send $(m, N^{(i)})$ to $s^{(i)}$, where $s_{k-1}^{(i)} = 1 \wedge N(i) = 1$
  else if $\exists\, i(s_{k+1}^{(i)} = 1 \wedge N(i) = 0)$
      then suboptimal_unicasting:
          send $(m, N^{(i)})$ to $s^{(i)}$, where $s_{k+1}^{(i)} = 1$
      else failure
end.

Algorithm unicasting_at_intermediate_node
begin
  {at any intermediate node u with message m and
   navigation vector N}
  if $N = 0$
  then stop
  else send $(m, N^{(i)})$ to $u^{(i)}$, where $u_{k-1}^{(i)} = 1 \wedge N(i) = 1$
end.

---

6. Performance Evaluation

The simulation study focuses on the following three aspects. (1) Percentage of optimal/suboptimal routing. (2) Comparison of safety vector and extended safety vector in terms of routing capability. (3) Performance results when $d = k$ (other than $k = 2$), i.e., each node knows the exact fault information that is within distance-k.

Percentage of optimal routing is measured by the probability of an optimal routing using the proposed approach for two randomly selected source and destination nodes. Again, an optimal routing to a distance-k destination is possible if

$s_k = 1$ for the source node or $s^{(i)}_{k-1} = 1$ for the source node's preferred neighbor along dimension i. In addition, suboptimal routing is feasible, if $s^{(i)}_{k+1} = 1$ for the source node's spare neighbor along dimension i. When the source and destination nodes are separated by 1- or 2-hop, optimal routing can be decided directly from the distance-1 and distance-2 information at the source node. Note that a minimal path may exist even when s_1 and s_2 are both zero.

Routing capability of safety vector and extended safety vector is compared mainly under the above two measures. Tables 1 and 2 show simulation results for 8-cubes and 10-cubes, respectively. Each table contains three subtables for three different distributions of faults: (a) represents cases of all faults being node faults; (b) for half faults being node faults and the other half being link faults; and (c) for all faults being link faults. Within each cube, for a given number of faults, these faults are randomly generated based on the specificed distribution of link and node faults. We selected 100 different fault distributions for each case. For a given fault distribution, we randomly selected 200,000 source and destination pairs. Percentage of the actual optimal routing is also reported (in the second column of each subtable). This also corresponds to cases when global fault information is given, i.e., $d = n$. Percentage of optimal routing, when distance-3 information is given, is shown in the third column of each subtable.

Based on results in Tables 1 and 2, the percentage of optimal routing under the extended safety vector model (when $d = 2$) stays very close to the one with global information (when $d = n$) for all cases. That is, the model for $d = 2$ is sufficient. Note that when all faults are node faults, the safety vector and the extended safety vector models are the same. However, as the percentage of link faults increases, the results for the safety vector model deteriorate quickly, especially for large numbers of faults. For example, when there are 75 link faults (no node faults) in a 10-cube, the percentage of optimal routing is only 35.82 percentage. For the extended safety vector model, the percentage of optimal routing remains high, especially when there is a high percentage of link faults. The summation of percentages of optimal (Op) and suboptimal (SubOp) routing corresponds to the percentage of the applicability of the corrsponding approach, which is either safety vector ($d = 1$) or extended safety vector ($d = 2$).

7. Conclusions

We have proposed a new coding method of limited global fault information in an n-cube. First each node collects precise fault information within distance-d, and then, fault information about nodes that are more than distance-d is coded in a special way. A model, called extended safety vector, has been proposed which is extended from the safety level and safety vector models to better handle link faults. The extended safety level model has been used to achieve optimal and suboptimal routing in an n-cube. A simulation study has been conducted based on different selections of d and results have shown a significant improvement under the proposed model over the safety vector model in handling link faults, even for a small value of d such as $d = 2$.

References

[1] M. S. Chen and K. G. Shin. Adaptive fault-tolerant routing in hypercube multicomputers. *IEEE Trans. on Computers*. 39, (12), Dec. 1990, 1406-1416.

[2] J. Duato. A theory of fault-tolerant routing in wormhole networks. *IEEE Trans. Parallal and Distributed Syst.* 8, (8), Aug. 1998, 790-802.

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

[4] Q. P. Gu and S. Peng. Unicast in hypercubes with large number of faulty nodes. *IEEE Trans. Parallal and Distributed Syst.* to appear.

[5] Y. Lan. A fault-tolerant routing algorithm in hypercubes. *Proc. of the 1994 International Conference on Parallel Processing*. August 1994, III 163 - III 166.

[6] T.C. Lee and J.P. Hayes. A fault-tolerant communication scheme for hypercube computers. *IEEE Trans. on Computers*. 41, (10), Oct. 1992, 1242-1256.

[7] C. S. Raghavendra, P. J. Yang, and S. B. Tien. Free dimensions - an effective approach to achieving fault tolerance in hypercubes. *Proc. of the 22nd International Symposium on Fault-Tolerant Computing*. 1992, 170-177.

[8] Y. Saad and M. H. Schultz. Topological properties of hypercubes. *IEEE Transactions on Computers*. 37, (7), July 1988, 867-872.

[9] S. B. Tien and C. S. Raghavendra. Algorithms and bounds for shortest paths and diameter in faulty hypercubes. *IEEE Transactions on Parallel and Distributed Systems*. Vol. 4, No. 6, June 1993, 713-718.

[10] J. Wu. Reliable communication in cube-based multicomputers using safety vectors. *IEEE Transactions on Parallel and Distributed Systems*. 9, (4), April 1998, 321-334.

[11] J. Wu. Unicasting in faulty hypercubes using safety levels. *IEEE Transactions on Computers*. 46, (2), Feb. 1997, 241-247.

[12] J. Wu and E. B. Fernandez. Broadcasting in faulty hypercubes. *Proc. of 11th Symposium on Reliable Distributed Systems*. Oct. 1992, 122-129.

# of faults	Op (d=n)	Op (d=3)	Safety Vec.		Ext. Safety Vec.	
			Op	SubOp	Op	SubOp
6	99.99	99.99	99.99	0.006	99.99	0.006
10	99.98	99.98	99.97	0.030	99.97	0.030
15	99.96	99.90	99.81	0.187	99.81	0.187
20	99.91	99.75	99.03	0.832	99.03	0.832
22	99.89	99.65	99.31	1.372	98.31	1.372
25	99.86	99.43	96.37	2.583	96.37	2.853
28	99.80	99.09	93.05	3.987	93.05	3.987
30	99.77	98.75	90.74	4.950	90.74	4.950

(a) When all faults are node faults in 8-cubes

# of faults	Op (d=n)	Op (d=3)	Safety Vec.		Ext. Safety Vec.	
			Op	SubOp	Op	SubOp
6	99.99	99.98	99.97	0.030	99.98	0.020
10	99.98	99.95	99.82	0.176	99.95	0.050
15	99.96	99.89	99.16	0.782	99.88	0.122
20	99.93	99.80	95.81	3.005	99.70	0.289
22	99.92	99.74	93.41	4.353	99.61	0.380
25	99.90	99.66	87.58	6.467	99.30	0.661
28	99.87	99.56	79.28	8.500	98.91	1.003
30	99.85	99.46	72.49	9.313	98.45	1.344

(b) When half faults are node faults in 8-cubes

# of faults	Op (d=n)	Op (d=3)	Safety Vec.		Ext. Safety Vec.	
			Op	SubOp	Op	SubOp
6	99.98	99.96	99.90	0.097	99.96	0.039
10	99.97	99.91	99.50	0.487	99.91	0.090
15	99.95	99.82	97.30	2.316	99.82	0.175
20	99.92	99.71	88.30	6.624	99.70	0.299
22	99.91	99.65	82.41	8.432	99.65	0.347
25	99.90	99.56	68.76	10.11	99.55	0.450
28	99.88	99.47	58.38	10.39	99.45	0.552
30	99.87	99.39	52.79	10.63	99.35	0.645

(c) When all faults are link faults in 8-cubes

Table 1. Percentage of optimal and suboptimal routing under different models.

# of faults	Op (d=n)	Op (d=3)	Safety Vec.		Ext. Safety Vec.	
			Op	SubOp	Op	SubOp
8	99.99	99.99	99.99	0.0003	99.99	0.0003
15	99.99	99.99	99.99	0.001	99.99	0.001
30	99.99	99.99	99.98	0.019	99.98	0.019
40	99.99	99.98	99.91	0.087	99.91	0.087
50	99.99	99.95	99.55	0.422	99.55	0.422
55	99.98	99.93	99.13	0.736	99.13	0.736
60	99.99	99.99	98.22	1.386	98.22	1.386
65	99.98	99.87	96.91	2.217	96.91	2.217
70	99.97	99.83	93.83	3.685	93.83	3.685
75	99.97	99.77	90.08	5.180	90.08	5.180

(a) When all faults are node faults in 10-cubes

# of faults	Op (d=n)	Op (d=3)	Safety Vec.		Ext. Safety Vec.	
			Op	SubOp	Op	SubOp
8	99.99	99.99	99.99	0.001	99.99	0.001
15	99.99	99.99	99.99	0.008	99.99	0.003
30	99.99	99.98	99.88	0.117	99.98	0.014
40	99.99	99.98	99.33	0.612	99.97	0.026
50	99.99	99.97	96.91	2.337	99.94	0.056
55	99.99	99.96	94.05	3.872	99.92	0.082
60	99.99	99.95	88.34	6.032	99.88	0.114
65	99.98	99.94	81.76	7.713	99.83	0.172
70	99.98	99.93	71.86	9.035	99.74	0.248
75	99.98	99.91	61.32	9.635	99.54	0.426

(b) When half faults are node faults in 10-cubes

# of faults	Op (d=n)	Op (d=3)	Safety Vec.		Ext. Safety Vec.	
			Op	SubOp	Op	SubOp
8	99.99	99.99	99.99	0.004	99.99	0.002
15	99.99	99.99	99.97	0.022	99.99	0.006
30	99.99	99.99	99.62	0.367	99.98	0.019
40	99.99	99.97	97.01	2.341	99.97	0.032
50	99.99	99.95	86.01	6.977	99.95	0.049
55	99.99	99.94	75.59	9.078	99.94	0.057
60	99.99	99.93	63.41	9.981	99.93	0.066
65	99.98	99.92	50.81	9.885	99.92	0.078
70	99.98	99.91	42.67	9.417	99.91	0.089
75	99.98	99.90	35.82	8.791	99.90	0.099

(c) When all faults are link faults in 10-cubes

Table 2. Percentage of optimal and suboptimal routing under different models.