Optimal and Reliable Communication in Hypercubes Using Extended Safety Vectors [1]

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

Feng Gao
Advanced Computer Architecture Lab.
University of Michigan
Ann Arbor, MI 48105

Zhongcheng Li and Yinghua Min
CAD Lab
Institute of Computing Technology
The Chinese Academy of Science
Beijing, P.R. China

[1]This work was supported in part by NSFC of P. R. China under grants 69703001 and 90104006 and by NSF of US under grants CCR 9900646, CCR 0329741, ANI 0073736, and EIA 0130806.

Abstract

We propose 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. Specifically, in our approach, each node in a cube-based multicomputer of dimension n is associated with an extended safety vector of n bits. 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, and the extended safety vector of each node can be easily calculated through $n-1$ rounds of information exchanges among neighboring nodes. Therefore, each extended safety vector is an approximated measure of the number and distribution of faults in the neighborhood. 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. 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 as in the extended safety vector model where $d = 2$.

Index Terms: *Fault-tolerant routing, generalized hypercubes, multicomputers, reliable communication, safety vectors.*

ACRONYMS

OP	Optimal
SubOP	Sub-optimal
VLSI	Very Large Scale Integration
MIMD	Multiple Instructions and Multiple Data

NOTATION

Q_n	n-dimensional hypercubes (n-cubes)
u, v	Nodes u and v
s	Source node
t	Destination node
m	Message
$u^{(i)}$	Neighbor of u along dimension i
$sl(u)$	Safety level of u
$sv(u)$	Safety vector of u
$esv(u)$	Extended safety vector of u
$f(u)$	A set of faulty paths of length 2 initiated from node u
$H(s,t)$	Hamming distance between s and t
$\oplus$	Exclusive OR operation
$N = s \oplus t$	Navigation vector

1 Introduction

Many experimental and commercial multicomputers use direct-connected networks with the grid topology. The binary hypercube is one of the popular grid structures. An *n-dimensional hypercube* (n-cube) 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. Several research prototypes and systems have been built in the past two decades, including NCUBE-2 [2], Intel iPSC [10], and the Connection Machine [6]. 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. Bypassing faulty components (nodes/links) in a system when routing a message to the destination is an important aspect of reliable communication. Adaptive and fault-tolerant routing protocols have been the subject of extensive research ([4], [5], [9]). A general theory of fault-tolerant routing is discussed in [3].

Limited-global-information-based routing is a compromise between local-information-based and global-information-based approaches. In a local-information-based approach, each node makes its decision based on local information; that is, fault information in the neighborhood. In a global-information-based approach, each node makes its decision based on global information. In a limited-global-information-based approach, each node makes its decision based on limited amount of global fault information. Depending on how limited global information is defined and collected, 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). By optimal routing, we mean that the unicast message reaches each destination through a minimal path (i.e., the length of each path is equal to the Hamming distance between the source and destination). Therefore, an approach of this type can be more cost effective than the ones based on global information [11] or local information ([1], [7]).

One simple but ineffective limited-global-information-based approach is to use distance-d information in which each node knows the status of all components within distance-d. 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 distance-d information.

Another limited-global-information-based 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' [8] *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 labelled (coded) faulty, unsafe, or safe.

- Wu and Fernandez' [14] *extended safe node* concept by relaxing certain conditions of Lee and Hayes' definition. Each node is still labelled 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. Xiang [15] applied the Lee and Hayes' model to each $(n-1)$-cube, generating n safety status for each node.

- Wu's *safety level* [13] 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* [12] 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 represents 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. 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 safety vector can be used as a navigation tool to direct a message to its destination through a minimal path. 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 diminish the routing capability of the system. Note that there are many types of link faults. One type is the network link fault and another type is the network interface fault, e.g., a failure in a particular network port in the router.

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 as 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 choice and its results stay very close to that of using global fault information, i.e., $d = n$. It is assumed the hypercube multicomputer is operated in an asynchronous MIMD mode. The extended safety vector at each node is determined based on those of its neighbors through asynchronous exchanges and updates.

This paper is organized as follows: Section 2 defines some notation and preliminaries. The safety level and safety vector models are also reviewed. Section 3 proposes the concept of extended safety vector. Section 4 illustrates this concept through several examples and providees several related properties. Section 5 presents an optimal/suboptimal unicasting algorithm based on the extended safety vector concept. Section 6 shows results of our simulation study.

2 Preliminaries

Hypercubes. An n-cube (Q_n) is a graph having 2^n nodes labelled 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 disconnected 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 t is equal to the Hamming distance between their binary addresses, denoted by $H(s,t)$. Symbol $\oplus$ denotes the bitwise exclusive OR operation on binary addresses of two nodes. Clearly, $s \oplus t$ has value 1 at $H(s,t)$ bit positions corresponding to $H(s,t)$ distinct dimensions. These $H(s,t)$ dimensions are called *preferred dimensions* and the corresponding nodes are termed *preferred neighbors*. The remaining $n - H(s,t)$ 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 $H(s,t)$ preferred dimensions in some order. For example, suppose $s = 0101$ and $t = 1011$, then $s \oplus t = 0101 \oplus 1011 = 1110$. 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 $t = 1011$ 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. The above path can be simply represented as (0101, 0001, 1001, 1011).

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 number associated with each node in a given n-cube, respectively. They provide coded information about fault information in the neighborhood.

Definition 1 [13]: *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 nondecreasing 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* $(sl_0, sl_1, sl_2, ..., sl_{n-1}) \geq (0, 1, 2, ..., n-1)$[1], *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 $sv(u) = (u'_1, u'_2, ..., u'_n)$. Assume that $sv(u^{(i)}) = (u'^{(i)}_1, u'^{(i)}_2, ..., u'^{(i)}_n)$ is the safety vector $u^{(i)}$, u's neighbor along dimension i.

Definition 2 [12]:

- *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:*

$$u'_1 = \begin{cases} 0 & \textit{if node } u \textit{ is an end-node of a faulty link} \\ 1 & \textit{otherwise} \end{cases}$$

- *Inductive definition for the kth bit:*

$$u'_k = \begin{cases} 0 & \textit{if } \sum_{1 \leq i \leq n} u'^{(i)}_{k-1} \leq n-k \\ 1 & \textit{otherwise} \end{cases}$$

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

Note that in the safety vector model, faulty links are considered and treated as follows: Each end node of a faulty link treats the other one faulty, but it does not consider itself faulty. This faulty link model is called conservative faulty link model.

In the safety level (vector) model, a node in an n-cube is said to be *safe* if its safety level is n (i.e., $(1, 1, ..., 1)$); 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 Hamming-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 Hamming-distance-k away.

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 accurater information about the number and distribution of faults in an n-cube. In terms of the number of safe nodes, for a given cube, it contains at least the same number of safe nodes under the safety vector model as the one under the safety level model. Both safety vectors and safety levels are calculated through $n - 1$ rounds of information exchanges among neighboring nodes. An optimal unicasting 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 the safety vector model can also be used in disconnected hypercubes by distinguishing infeasible routing from feasible ones. In [12], it is shown that the safety vector concept can be extended to other cube-based multicomputers, such as generalized hypercubes. 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. Node 111 should be able to send a message to node 100 through a shortest path (111 - 101 - 100). When node 101 with an adjacent faulty link is treated as faulty (labelled 0), such a shortest path cannot be identified. Therefore, a conservative node labelling scheme is more likely to disable some shortest paths. In fact, nodes 110, 101, and 111 can send a message to any nodes through a minimal path that are distance-2 or -3 away. Ideally, these nodes should be labelled a safety level of 2. This problem is partially resolved in the

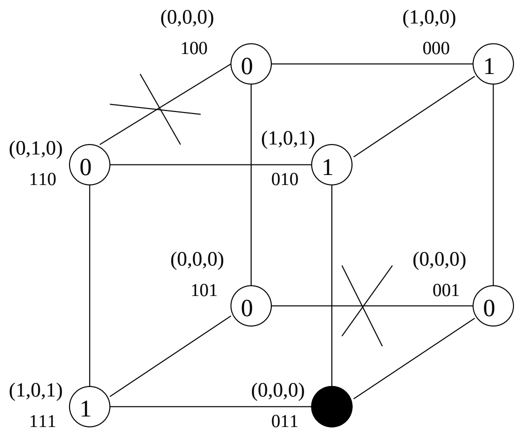

Figure 1: An example of a faulty 3-cube with its safety level and safety vector assignments.

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 paper, 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

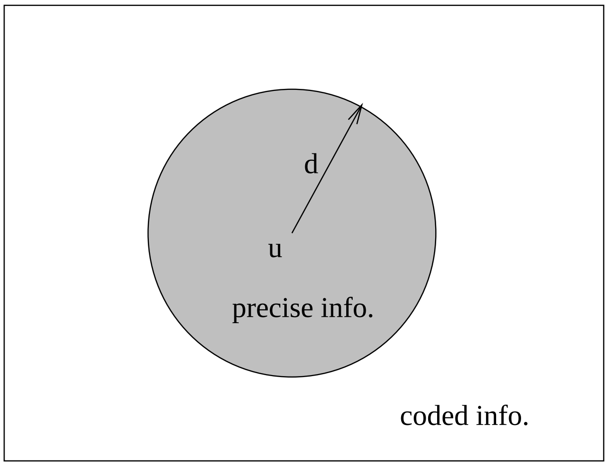

Figure 2: A new approach to code information.

coded. We show that $d = 2$ is sufficient to handle link faults and there is no need to select a larger d (this will be confirmed by our simulation results later). This approach is called *extended safety vectors*, 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 $d = 1$. However, the regular safety vector model cannot accurately represent a fault link with one end node that is adjacent to the current node and the other end node distance-2 away. The location of such a faulty link can be precisely located in the model where $d = 2$. 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.,*

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

- *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 kth bit, where* $k \geq 3$*:*

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

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 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.

Note that the complexity of the safety vector and the extended safety vector models are the same in term of the number of rounds of message exchanges. The only difference is in the second round for determining the second bit. In the extended safety vector model, each node exchanges the adjacent link set of each neighbor. Therefore, the size of that particular message is $O(n^2)$. The size of that particular message in the regular safety vector model is $O(n)$.

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). 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 $\{j\} \cap c_i^{'} \neq \phi$. ■

Fig. 3 shows the u_2 value of node u for four sample fault distributions. 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. 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

In the example of Fig. 1, the extended safety vectors for nodes 0 to 7 are (1,1,1), (0,1,1), (1,1,1), (0,0,0), (0,1,1), (0,1,1), (0,1,1), and (1,1,1), respectively. Fig. 4 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 $(1, 1, 1, 1)$. Table 1 shows a round-by-round calculation of the safety vector and extended safety vector of each node in Fig. 4. Note that for this example, a 3-round (including the initial assignment) of information exchanges is needed for the safety vector model and a 2-round of information exchanges is needed for the extended safety vector model.

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 Hamming-distance-k 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. 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

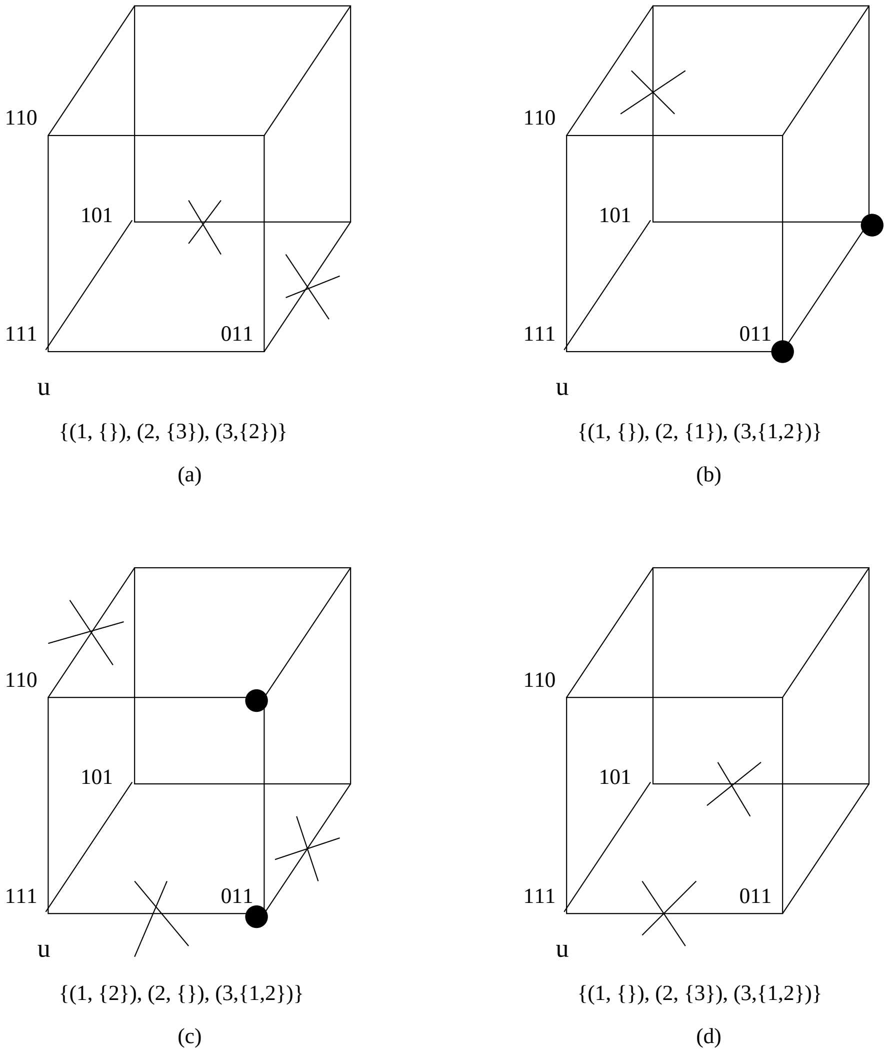

Figure 3: The u_2 value for different fault distributions: (a) $u_2 = 0$, (b) $u_2 = 1$, (c) $u_2 = 1$, and (d) $u_2 = 0$.

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.

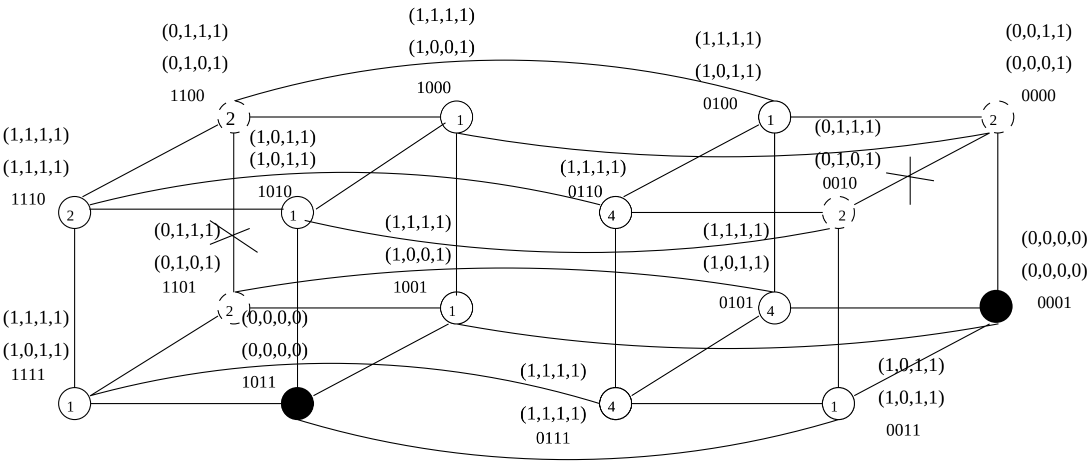

Figure 4: A faulty 4-cube with two faulty nodes and two faulty links.

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

(a) Initial extended safety vector assignments

node address	0000	0001	0010	0011	0100	0101	0110	0111
safety vec.	(0,$\underline{0}$,1,1)	(1,1,1,1)	(0,1,1,1)	(1,$\underline{0}$,1,1)	(1,$\underline{0}$,1,1)	(1,1,1,1)	(1,1,1,1)	(1,1,1,1)
extended safety vec.	(0,$\underline{0}$,1,1)	(1,1,1,1)	(0,1,1,1)	(1,$\underline{0}$,1,1)	(1,1,1,1)	(1,1,1,1)	(1,1,1,1)	(1,1,1,1)
node address	1000	1001	1010	1011	1100	1101	1110	1111
safety vec.	(1,$\underline{0}$, 1, 1)	(1,$\underline{0}$,1,1)	(1,$\underline{0}$,1,1)	(0,0,0,0)	(0,1,1,1)	(0,1,1,1)	(1,1,1,1)	(1,$\underline{0}$,1,1)
extended safety vec.	(1,1, 1, 1)	(1,1,1,1)	(1,1,1,1)	(0,0,0,0)	(0,1,1,1)	(0,1,1,1)	(1,1,1,1)	(1,1,1,1)

(b) Extended safety vectors after the first round

node address	0000	0001	0010	0011	0100	0101	0110	0111
safety vec.	(0,0,$\underline{0}$,1)	(1,1,1,1)	(0,1,$\underline{0}$,1)	(1,0,1,1)	(1,0,1,1)	(1,0,1,1)	(1,1,1,1)	(1,1,1,1)
extended safety vec.	(0,0,1,1)	(1,1,1,1)	(0,1,1,1)	(1,0,1,1)	(1,1,1,1)	(1,1,1,1)	(1,1,1,1)	(1,1,1,1)
node address	1000	1001	1010	1011	1100	1101	1110	1111
safety vec.	(1,0,$\underline{0}$,1)	(1,0,$\underline{0}$,1)	(1,0,1,1)	(0,0,0,0)	(0,1,$\underline{0}$,1)	(0,1,$\underline{0}$,1)	(1,1,1,1)	(1,0,1,1)
extended safety vec.	(1,1,1,1)	(1,1,1,1)	(1,1,1,1)	(0,0,0,0)	(0,1,1,1)	(0,1,1,1)	(1,1,1,1)	(1,1,1,1)

(c) Extended safety vectors after the second round

Table 1: The calculation of the safety vector and extended safety vector of each node in the 4-cube of Fig. 4.

Hamming distance path from node u to any node which is exactly Hamming-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 d, which is Hamming-distance-l away. Connecting the link from node u to node v to the path originated from node v to destination node d, we construct a Hamming distance path from node u to destination node w which is Hamming-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 $esv(u)$ covers $sv(u)$ 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 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 [12]. Suppose that source node s, with safety vector $(s_1, s_2, ..., s_n)$, intends to forward a message to a node that is Hamming-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*, $N = s \oplus t$, 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 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.

Note that if the kth bit of the extended safety vector of the source node is 0, it may still be possible to find a minimal path for a distance-k destination, as long as one of its preferred neighbors has its $(k-1)$th bit set. In the example of Fig. 3, suppose node 1000 sends a message to its distance-2 destination 1101. The regular safety vector of node 1000 is (1,0,0,1) and the extended safety vector is (1,1,1,1). Clearly, there is no problem to find a minimal path under the extended safety vector model. The regular safety vector model can still find a minimal path in this case, since one of preferred neighbors of the source has its 1st bit set. That is, the message reaches the destination via node 1001 (with a safety vector of (1,0,0,1)).

For $k = 1$, source s and destination t are neighbors, as long as their connecting link is healthy, optimal routing is still possible even when $s_1 = 0$ in the extended safety vector of the source. Similarly for $k = 2$, optimal routing is decided based on distance-1 and distance-2 information (rather than the extended safety vector), optimal routing is possible if a healthy 2-hop path exists from s to t (even

Algorithm **unicasting_at_source_node**

begin

$N = s \oplus t$; $k = |s \oplus t|$;

if $s_k = 1 \vee \exists i(s^{(i)}_{k-1} = 1 \wedge N(i) = 1) \vee$

$(k = 1$ (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 $s^{(i)}_{k-1} = 1$ and $N(i) = 1$

else **if** $\exists\ i(s^{(i)}_{k+1} = 1 \wedge N(i) = 0)$

then suboptimal_unicasting:

send $(m, N^{(i)})$ to $s^{(i)}$, where $s^{(i)}_{k+1} = 1$

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 $u^{(i)}_{k-1} = 1$ and $N(i) = 1$

end.

when $s_2 = 0$ and $s_1^{(i)} = 0$ for all neighbors).

Consider another routing example with source 1000 and destination 0011 (which is distance-3 away from the source). A minimal path can be easily determined under the extended safety vector model, since source 1000 is a safe node. However, a minimal path cannot be found using the regular safety vector model. Because the 3rd bit of the safety vector of the source is 0, in addition, the 2nd bit of the safety vectors of all its preferred neighbors (1010, 0100, and 1001) is 0. This example also demonstrates that the extended safety vector model is strictly more powerful than the regular safety vector model.

6 Performance Evaluation

Simulation study focuses on the following four aspects.

1. Percentage of optimal/suboptimal routing.
2. Comparison of safety vector and extended safety vector in terms of routing capability.
3. Percentage of safe nodes and safe neighbors.
4. 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 $s_{k-1}^{(i)} = 1$ for the source node's preferred neighbor along dimension i. In addition, suboptimal routing is feasible, if $s_{k-1}^{(i)} = 1$ 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.

When a source node is safe, it indicates the existence of an optimal routing from the source to any destination node. When a neighbor is safe, at least there exists a suboptimal routing from the source to any destination node.

Routing capability of safety vector and extended safety vector is compared mainly under the above two measures. Tables 2 and 3 show simulation results for 8-cubes and 10-cubes, respectively. We did not include large cubes with a dimension size over 10, because most commercial systems do not scale over dimension 10. In addition, we can deduce results for large cubes from those of 8-cubes and 10-cubes as will be shown later. 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 specified 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 2 and 3, 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 fault) in a 10-cube, the percentage of optimal routing is only 35.8212 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 columns under "Total" represent the summation of percentages of optimal and suboptimal routings, i.e., the percentage of the applicability of the proposed approach. It is expected that fault tolerance capability increases as the dimension of the hypercube increases; that is, for the same number of faults, the percentage of optimal and suboptimal routing increases as the dimension increases. In conclusion, the conservative faulty link model used in the regular safety vector model does not handle faculty links effectively, especially when there is a large number of faults; whereas the proposed extended safety vector can handle faulty links effectively without increasing the complexity of the original safety vector model.

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] NCUBE Company. *NCUBE 6400 Processor Manual.* 1990.

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

#of faults	OP Exist ($d=n$)	OP ($d=3$)	Safety Vector			Extended Safety Vector		
			OP ($d=1$)	SubOP	Total	OP ($d=2$)	SubOP	Total
6	99.9944	99.9938	99.9937	0.0063	100.00	99.9937	0.0063	100.00
10	99.9840	99.9756	99.9697	0.0303	100.00	99.9697	0.0303	100.00
15	99.9565	99.9016	99.8082	0.1873	99.9954	99.8082	0.1873	99.9954
20	99.9137	99.7470	99.0317	0.8318	99.8635	99.0317	0.8318	99.8635
22	99.8941	99.6538	99.3094	1.3715	99.6809	98.3094	1.3715	99.6809
25	99.8569	99.4278	96.3706	2.5825	98.9532	96.3706	2.8525	98.9532
28	99.7978	99.0941	93.0474	3.9874	97.0348	93.0474	3.9874	97.0348
30	99.7746	98.7512	90.7403	4.9496	95.6899	90.7403	4.9496	95.6899

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

#of faults	OP Exist ($d=n$)	OP ($d=3$)	Safety Vector			Extended Safety Vector		
			OP ($d=1$)	SubOP	Total	OP ($d=2$)	SubOP	Total
6	99.9982	99.9794	99.9697	0.0303	100.00	99.9804	0.0196	100.00
10	99.9766	99.9509	99.8236	0.1755	99.9991	99.9496	0.0504	100.00
15	99.9563	99.8928	99.1624	0.7822	99.9447	99.8781	0.1219	100.00
20	99.9285	99.8021	95.8135	3.0050	98.8185	99.7043	0.2892	99.9935
22	99.9176	99.7421	93.4052	4.3530	97.7582	99.6111	0.3799	99.9910
25	99.8973	99.6577	87.5787	6.4668	94.0455	99.3008	0.6606	99.9614
28	99.8742	99.5601	79.2779	8.4955	87.7735	98.9081	1.0032	99.9113
30	99.8539	99.4647	72.4938	9.3132	81.8070	98.4505	1.3436	99.8539

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

#of faults	OP Exist ($d=n$)	OP ($d=3$)	Safety Vector			Extended Safety Vector		
			OP ($d=1$)	SubOP	Total	OP ($d=2$)	SubOP	Total
6	99.9804	99.9626	99.9035	0.0965	100.00	99.9608	0.0392	100.00
10	99.9656	99.9110	99.5036	0.4865	99.9901	99.9098	0.0902	100.00
15	99.9453	99.8240	97.2949	2.3160	99.6109	99.8246	0.1754	100.00
20	99.9211	99.7073	88.2978	6.6243	94.9221	99.7008	0.2992	100.00
22	99.9132	99.6486	82.4090	8.4315	90.8405	99.6528	0.3472	100.00
25	99.8992	99.5624	68.7555	10.1083	78.8638	99.5505	0.4495	100.00
28	99.8820	99.4651	58.3818	10.3869	68.7714	99.4485	0.5515	100.00
30	99.8678	99.3881	52.7934	10.6309	63.4243	99.3521	0.6447	99.9968

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

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

#of	OP Exist	OP	Safety Vector			Extended Safety Vector		
faults	$(d = n)$	$(d = 3)$	OP $(d = 1)$	SubOP	Total	OP $(d = 2)$	SubOP	Total
8	99.9997	99.9998	99.9997	0.0003	100.00	99.9997	0.0003	100.00
15	99.9991	99.9988	99.9989	0.0011	100.00	99.9989	0.0011	100.00
30	99.9956	99.9908	99.9814	0.0186	100.00	99.9814	0.0186	100.00
40	99.9927	99.9799	99.9107	0.0868	99.9975	99.9107	0.0868	99.9975
50	99.9869	99.9508	99.5489	0.4216	99.9705	99.5489	0.4216	99.9705
55	99.9839	99.9347	99.1311	0.7355	99.8665	99.1311	0.7355	99.8665
60	99.9793	99.9903	98.2246	1.3860	99.6106	98.2246	1.3860	99.6106
65	99.9765	99.8648	96.9108	2.2168	99.1185	96.9108	2.2168	99.1185
70	99.9710	99.8296	93.8286	3.6851	97.5137	93.8286	3.6851	97.5137
75	99.9665	99.7669	90.0849	5.1798	95.2647	90.0849	5.1798	95.2647

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

#of	OP Exist	OP	Safety Vector			Extended Safety Vector		
faults	$(d = n)$	$(d = 3)$	OP $(d = 1)$	SubOP	Total	OP $(d = 2)$	SubOP	Total
8	99.9991	99.9987	99.9981	0.0019	100.00	99.9987	0.0013	100.00
15	99.9981	99.9967	99.9923	0.0077	100.00	99.9967	0.0033	100.00
30	99.9951	99.9887	99.8797	0.1186	99.9982	99.9863	0.0137	100.00
40	99.9929	99.9793	99.3337	0.6121	99.9458	99.9743	0.0257	100.00
50	99.9898	99.9664	96.9056	2.3372	99.2428	99.9443	0.0557	100.00
55	99.9875	99.9591	94.0471	3.8721	97.9192	99.9171	0.0818	99.9989
60	99.9861	99.9500	88.3401	6.0322	94.3723	99.8849	0.1144	99.9993
65	99.9842	99.9376	81.7614	7.7127	89.4741	99.8259	0.1724	99.9983
70	99.9815	99.9271	71.8642	9.0349	80.8990	99.7423	0.2484	99.9907
75	99.9791	99.9142	61.3216	9.6352	70.9568	99.5413	0.4258	99.9672

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

#of	OP Exist	OP	Safety Vector			Extended Safety Vector		
faults	$(d = n)$	$(d = 3)$	OP $(d = 1)$	SubOP	Total	OP $(d = 2)$	SubOP	Total
8	99.9984	99.9976	99.9954	0.0046	100.00	99.9975	0.0025	100.00
15	99.9970	99.9936	99.9785	0.0215	100.00	99.9937	0.0063	100.00
30	99.9938	99.9908	99.6188	0.3669	99.9857	99.9807	0.0193	100.00
40	99.9913	99.9681	97.0059	2.3409	99.3468	99.9679	0.0320	100.00
50	99.9887	99.9520	86.0142	6.9771	92.9913	99.9510	0.0491	100.00
55	99.9877	99.9438	75.5854	9.0781	84.6625	99.9432	0.0568	100.00
60	99.9862	99.9334	63.4148	9.9812	73.3960	99.9345	0.0655	100.00
65	99.9847	99.9231	50.8087	9.8852	60.6939	99.9220	0.0780	100.00
70	99.9837	99.9136	42.6686	9.4173	52.0859	99.9115	0.0885	100.00
75	99.9823	99.9018	35.8212	8.7914	44.6126	99.9012	0.0988	100.00

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

Table 3: Percentage of optimal and suboptimal routing under different models.

[4] 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.

[5] Q. P. Gu and S. Peng. Unicast in hypercubes with large number of faulty nodes. *IEEE Trans. Parallal and Distributed Syst.* 10, (10), Oct. 1999, 964-975.

[6] W.D. Hills. *The Connection Machine.* MIT Press, Cambridge, MA, 1985.

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

[8] 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.

[9] 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.

[10] J. Rattler. Concurrent processing: A new direction in scientific computing. *Proc. of AFIPS Conference.* Vol. 54, 1985, 157-166.

[11] 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.

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

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

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

[15] D. Xiang. Fault-tolerant routing hypercube multicomputers using local safety information. *IEEE Transactions on Parallel and Distributed Systems.* Vol.12, No. 9, 2001, 942-951.