Reliable Communication in Cube-Based Multicomputers Using Safety Vectors

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

Abstract

Reliable communication in cube-based multicomputers (including disconnected ones) using the safety vector concept is studied in this paper. In the proposed approach each node in a cube-based multicomputer of dimension n is associated with a safety vector of n binary numbers which is an approximated measure of the number and distribution of faults in the neighborhood. The safety vector of each node in an n-dimensional hypercube 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 the destination. An extended deadlock-free unicasting using virtual channels is also introduced.

1 Introduction

With its numerous attractive features, the binary hypercube has been one of the topological structures for multicomputers. Efficient interprocessor communication is a key to the performance of a cube-based multicomputer. *Unicasting* is a one-to-one communication between a source and a destination. Unicasting in fault-free hypercubes and its variations has been extensively studied ([8], [9]). As the number of processors in a cube-based multicomputer increases, the probability of processors failure also increases. There has been a number of fault-tolerant unicasting schemes proposed in previous work ([2], [3], [4], [7]). Most existing models assume that each node knows either only the neighbors' status (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 sub-optimal) and adaptive routing can be designed based on such information. An optimal routing is also called minimal routing that forwards a message to the destination through a shortest path. A routing is adaptive if it can use alternative paths between the source and the destination, making more efficient use of network bandwidth and providing resilience to failure.

Normally, a global-information-based model can obtain an optimal or suboptimal result; however, it requires a complex process to collect global information. Such global information is presented in a tabular format and it is not easy to use. Local-information-based models use a weaker but a more reasonable assumption; however, local information can only be used to achieve local optimization and most of approaches based on this model are heuristic in nature. Therefore, the length of a routing path is unpredictable in general and global optimization, such as time and traffic in routing, is impossible. *Limited-global-information-based* routing is a compromise between local-information- 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.

Lee and Hayes [4] proposed the concept of safe node 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. A routing algorithm was proposed based on the status of each node and can route a message via a path of length no larger than two plus the Hamming distance between the source and the destination as long as the hypercube is not fully unsafe. A similar safety node concept has been used recently to represent convex-type fault blocks in a 2-dimensional mesh [1]. Wu and Fernandez [13] ex-

1063-6927/96 $5.00 © 1996 IEEE
Proceedings of the 16th ICDCS

tended the Lee and Hayes' safe node concept by relaxing certain conditions and hence increased the size of the safe nodes 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 safety level concept [12] 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, $0 \leq k \leq n$. 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. When a faulty n-cube has fewer than n faulty nodes, the unicasting algorithm based on safety level ensures an optimal unicasting (generating an optimal path) or suboptimal unicasting (generating a path with a length of Hamming distance between the source and the destination plus two). However, the safety level concept has the following two pitfalls: (1) Suppose the safety level of a node is k, it only tells that there exists a Hamming distance path to any node within k Hamming distance. There is no information about the existence of Hamming distance path to nodes that are more than k distance away. (2) The safety level concept applies to hypercubes with faulty nodes but it is rather inefficient to cover link faults.

The *safety vector* concept proposed in this paper can effectively include faulty links information and provide more accurate information about number and distribution of faults in the system. Basically, each node in an n-dimensional hypercube is associated with a binary vector called safety vector which can be easily calculated through $n-1$ rounds of information exchange among neighboring nodes. An optimal one-to-one routing 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 the destination. Routing based on safety vector can also be used in disconnected hypercubes, where nodes in a hypercube are disjointed (into two or more parts). In [11], we also show that the safety vector concept can be extended to other cube-based multicomputers such as generalized hypercubes.

Deadlock may happen when multiple nodes send their unicast messages simultaneously. In order to maintain dynamic adaptivity while ensure deadlock freeness, we propose two extensions: one uses $n+1$ *virtual channels* and other one adapts from any existing fully adaptive minimal routing for fault-free n-cube that uses $k(n)$ virtual channels by adding one extra virtual channel.

We make the following assumptions for the techniques used 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 an algorithm to be perfect. We do assume that each node knows exactly the safety status of all its neighbors and can distinguish a faulty link from a faulty node.

2 Notation and Preliminaries

The n-dimensional hypercube (or 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 a has an address $a_n a_{n-1} \cdots a_1$ with $a_i \in \{0,1\}$, $1 \leq i \leq n$, and a_i is called the ith bit (also called the ith dimension) of the address. We denote node a^i the neighbor of a along dimension i. Symbol $\oplus$ denotes the bitwise exclusive-or operation on binary addresses of two nodes. For example, $1101 \oplus 0110 = 1011$. $a \oplus e^i$ represents setting or resetting the ith bit of node a. For example, $1101 \oplus e^3 = 1001$. The distance between two nodes s and d is equal to the Hamming distance between their binary addresses, denoted by $H(s,d)$.

A path connecting two nodes s and d is termed *optimal path* if its length is equal to the Hamming distance between these 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*. An optimal path can be obtained by using links at each of these $H(s,d)$ preferred dimensions in some order.

The challenge of designing a routing algorithm in a faulty hypercube is to find a feasible routing without sacrificing adaptivity. In the 4-cube shown in Figure 1, suppose that node 1111 is the source and 0010 is the destination. In the absence of fault, the maximum adaptivity at source 1111 is $H(1111, 0010) = 3$. However, node 1011 cannot be used as an intermediate node in this case. Therefore, the maximum adaptively at source 1111 with respect to the destination 0010 is 2. That is, the message can be forwarded to either 1110 or 1011. To find maximum adaptivity, we need a simple but effective way of representing fault distribution in the neighborhood. The safety vector concept provides such a mechanism.

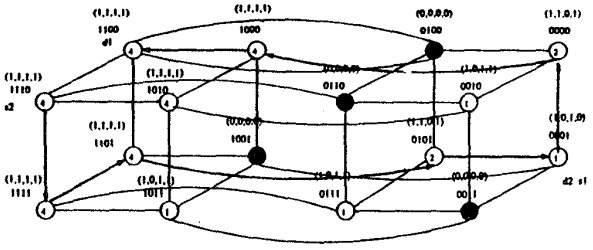

Figure 1: A 4-cube with four faulty nodes

3 The Safety Vector and its Properties

In the proposed approach, fault information is captured in a safety vector of n binary numbers, $(a_1, a_2, ..., a_n)$, associated with each node a in an n-cube. The safety vector associated with a node is an approximated measure of the number and distribution of faults in the neighborhood, rather than just the number of faults. The safety degree of node a is *strictly higher* than the safety degree of node b if and only if $(a_1, a_2, ..., a_n) \geq (b_1, b_2, ..., b_n)$ and there exists a k such that $a_k > b_k$. Note that strictly higher is a partial order, not all the pairs of safety vectors can be ordered. A faulty node is associated with $(0, 0, ..., 0)$ which corresponds to the lowest degree of safety, while a node with $(1, 1, ..., 1)$ as its safety vector has the highest safety degree and is called *safe node.*

Definition 1: *The safety vector of a faulty node is* $(0, 0, ..., 0)$. *For a nonfaulty node a, assume that* $(a_1, a_2, ..., a_n)$ *is a's safety vector and* $(a_1^{(i)}, a_2^{(i)}, ..., a_n^{(i)})$ *is the safety vector of node a's neighbor along dimension i. If node a 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 a, and*

$$a_1 = \begin{cases} 0 & \text{if node } a \text{ is an end-node of a faulty link} \\ 1 & \text{otherwise} \end{cases}$$

and

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

The safety vector defined in Definition 1 ensures the following property: if the kth bit of the safety vector of a node is one, then randomly select k neighbors, there exists at least one neighbor which has 1 in the $(k-1)$th bit of its safety vector. This property will be used as a basis of the proposed routing algorithm.

Note that there are different views of the safety status of an end node of a faulty link. Two end nodes of a faulty link cannot reach each other and 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.

```
Algorithm global_status (GS)
begin
    forall a ∈ Q_n
        if a is an end node of a faulty link
        then a_1 = 0 else a_1 = 1;
    for k = 2 step 1 to n
        begin
            forall a ∈ Q_n
            collects all the (k − 1)th bits, a_{k−1}^{(i)},
            of neighbors safety vectors
                if Σ_{1≤i≤n} a_{k−1}^{(i)} ≤ n − k
                then mark a_k as 0
                else mark a_k as 1
        end
end.
```

Theorem 1: *For any given faulty hypercube, there is one and only one way to assign safety vectors to nodes that satisfies the safety vector definition.*

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 ith bit depends on all the $(i-1)$th bits of neighbors' safety vectors. Clearly, all the ith bits can be fixed once all the $(i-1)$th bits are determined. Because all the 1st bits are fixed, there is one possible assignment of all the other bits. □

The proof of Theorem 1 also suggests a simple way to identify 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 ith bit at the $(i-1)$th round. We assume that all nonfaulty nodes without adjacent faulty links have $(1, 1, 1, ..., 1)$ and nonfaulty nodes with adjacent faulty links have $(0, 1, 1, ..., 1)$ as their initial safety vectors. All faulty nodes have $(0, 0, 0, ..., 0)$ as their safety vector. In the absence of faults, the safety vector associated each node is (1,1,1,..,1), i.e., there is no need to calculate the safety vector in this case.

Figure 1 also shows the safety vectors obtained by applying GS to the 4-cube. In this case, 3 rounds are used. The safety vector $(1, 1, 0, 1)$ associated with

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 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 $k \leq \min\{j|a_{j+1} = 0\}$ based on Theorem 4, where $(a_1, a_2, ..., a_n)$ is the safety vector in the converted hypercube, when $k = n$, $(a_1, a_2, ..., a_n)$ 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-dimensional hypercube without link fault, assume that W is the number of faulty nodes. If $W < n$ then each nonfaulty node has a safe neighbor.*

4 Routing Using Safety Vectors

The basic idea used in the optimal routing is as follows: suppose that the source 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 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 spare neighbor which 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 followed. In this case, the length of the resultant path is Hamming distance plus two. We call this result *suboptimal.* The selection of the optimal and the suboptimal algorithms can be decided locally at the source node, using the following information: (1) The Hamming distance of the source (s) and the destination (d). This can be done by computing $H(s,d) = |s \oplus d|$. The preferred neighbor sets and the spare neighbor sets are obtained based on $s \oplus d$. (2) The safety vector of the source node $(s_1, s_2, ..., s_n)$. (3) The safety vectors of the neighboring nodes of the source node ($s^{(i)}_1$, $s^{(i)}_2$, ..., $s^{(i)}_n$) along dimension i.

A *navigation vector*, $N = s \oplus d$ is used which is the relative address between the source and the destination. This vector can be determined at the source node and it is passed to a selected neighbor after resetting or setting the corresponding bit of N. At each intermediate node, a preferred neighbor which is k distance away from the destination and which has 1 in the kth bit of its safety vector is selected. Each intermediate node calculates its preferred and spare neighbors upon receiving the unicast message with a navigation vector. A unicasting completes when N in the navigation vector becomes zero; that is, each bit is zero. Note that, at the source node, if both conditions for optimal and suboptimal unicasting fail, the proposed algorithm cannot be applied. This failure state can be easily detected at the source node.

The routing algorithm consists of two parts: one for the source node (unicasting_at_source_node) and the other one for each intermediate node (unicasting_at_intermediate_node). We use the following notation: source node is denoted as s with safety vector $(s_1, s_2, ..., s_n)$, neighbors' safety vectors $(s^{(i)}_0, s^{(i)}_1, s^{(i)}_2, ..., s^{(i)}_n)$ along dimension i, routing message m, and destination d. $s^{(i)}_0$ is an extra bit introduced to simplify the algorithm. $s^{(i)}_0 = 1$ except for the case that nodes s and s^i are two end nodes of a faulty link.

Algorithm unicasting_at_source_node
begin
  $N = s \oplus d$; $H = |s \oplus d|$;
  if $s_H = 1 \vee \exists i(s^{(i)}_{H-1} = 1 \wedge N(i) = 1)$
  then optimal_unicasting:
    send $(m, N \oplus e^i)$ to s^i, where $s^{(i)}_{H-1} = 1$
  else if $\exists\ i(s^{(i)}_{H+1} = 1 \wedge N(i) = 0)$
    then suboptimal_unicasting:
    send $(m, N \oplus e^i)$ to s^i, where $s^{(i)}_{H+1} = 1$
  else failure
end.

Algorithm unicasting_at_intermediate_node
begin
  {at any intermediate node a with unicast message m, navigation vector N, and relative distance H}
  if $N = 0$ **then** stop
  else send $(m, N \oplus e^i)$ to a^i,
    where $a^{(i)}_{H-1} = 1$ and $N(i) = 1$
end.

Theorem 6: *Suppose that the Hamming distance between the source and the destination is k for a given unicasting. When the kth bit of the safety vector of*

node 0000 indicates the existence of an optimal path to any other node, except nodes that are 3 Hamming distance away.

Theorem 2: *Given any faulty 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 $(a_1, a_2, ..., a_n)$ is the safety vector associated with node a in any faulty n-cube. If $a_i = 1$ then there exists at least one Hamming distance path from node a to any node which is exactly i Hamming distance away from node a.*

Proof: We prove this theorem by induction on i. If $a_1 = 1$ (where $i = 1$) there is no adjacent faulty link. Clearly node a can reach any neighboring nodes, faulty and nonfaulty. Assume that this theorem holds for $i = k$, i.e., if $a_k = 1$ there exists at least one Hamming distance path from node a to any node which is exactly k Hamming distance away. When $i = k+1$, if $a_i = 1$ then $\sum_{1 \le j \le n} a_k^{(j)} > n - (k+1)$, which means that there are at most k neighbors which have 0 at the kth bit of their safety vectors. Therefore, among $k+1$ preferred neighbors, there is at least one neighbor, say node b where the kth bit of its safety vector is one. Based on the induction assumption, there is at least one Hamming distance path from node b to any node which is k Hamming distance away a. Because we can always find such a node b, for any given destination which is $k+1$ Hamming distance away from node a there exists at least one Hamming distance path from node a to this destination. □

To relate the safety level concept proposed earlier to the safety vector concept, we first given the safety level definition and its relevant properties.

Definition 2 [12]: *The safety level of a faulty node is 0. For a nonfaulty node a, let $(S_0, S_1, S_2, ..., S_{n-1})$, $0 \le S_i \le n$, be the nondecreasing safety level sequence of node a'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 a is defined as: if $(S_0, S_1, S_2, ..., S_{n-1}) \ge (0, 1, 2, ..., n-1)$[1], then $S(a) = n$ else if $(S_0, S_1, S_2, ..., S_{k-1}) \ge (0, 1, 2, ..., k-1) \wedge (S_k = k-1)$ then $S(a) = k$.*

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 iterative algorithm similar to GS can be used to calculate the safety level of each node in an n-cube. We assume that all nonfaulty nodes in a Q_n have n as their initial safety levels. It has been proved [12] that for any faulty n-cube, $n-1$ rounds of information exchange are sufficient. Figure 1 also shows the safety level of each node. Based on the safety level definition, the safety levels of all the nodes that have two (or more) faulty neighbors will be changed to 1 after the first round, as in the case for nodes 0001, 0010, 0111, 1011. That is, the effect of 0-safe status of faulty nodes will first propagate to their neighbors, then neighbors' neighbors and so on. Obviously, the safety vector model provides more information than the safety level model. In Figure 1, the safety vector associated with node 0000 is (1,1,0,1). However, node 0000 has a safety level 2 which means that node 0000 can reach any node within 2 Hamming distance away through an optimal path and it is not guaranteed that there exists an optimal path to any node which is 3 Hamming distance away (otherwise the safety level would be at least 3). The safety level 2 (of node 0000) does not tell whether there exists an optimal path from node 0000 to any node which is 4 Hamming distance away. There are several interesting properties [12] of safety level summerized as follows:

Property 1: *In a faulty n-cube with no more than $n-1$ faulty nodes, 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.*

The following theorem reveals the relationship between the safety vector and the safety level. The proof of this theorem is rather involved and can be found in [11].

Theorem 4: *Assume that k is the safety level of node a in a faulty n-cube and $(a_1, a_2, ..., a_n)$ is the safety vector of node a, then $k \le \min\{j | a_{j+1} = 0\}$. If node a is safe then $(a_1, a_2, ..., a_n) = (1, 1, ..., 1)$.*

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 $k < \min\{j | a_{j+1} = 0\}$. Let's consider a 5-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. Applying the GS algorithm, we have $(1, 1, 1, 1, 1)$ as node 00000's safety vector, i.e., node 00000 is safe.

Theorem 5: *In an n-dimensional hypercube, 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 a adjacent faulty*

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

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 unicasting algorithm. When there is a spare neighbor of the source node which has 1 at the $(k+1)$th bit of the safety vector, then suboptimality is guaranteed.

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

Corollary: *In the worst case, the length of any path generated from the proposed unicasting algorithm is no more than $n+1$, where n is the dimension of the hypercube.*

Consider the example of Figure 1. Suppose that the source node is 0001 with safety vector $(1,0,1,0)$, which means that the Hamming distance path is guaranteed to any node which is one or three Hamming distance away. Hamming distance path may or may not exist to a node two or four 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 one at the 2nd bit of its safety vector. In this case nodes 0101 and 0000 are both eligible. Assume node 0000 is selected to which the routing message together with $N = 1101 \oplus 0001 = 1100$ are forwarded. Node 0000 in turn forwards the message to one of it's preferred neighbors which has one at the 1st bit of its safety vector. Clearly, only node 1000 (the neighbor along dimension 4) is eligible. Once node 1000 receives the message together with $N = 0100$, it sends the message to the destination node (the neighbor along 3) as indicated in $N = 0100$. The selected path from $s_1 = 0001$ to $d_1 = 1101$ is shown in Figure 1.

Figure 2 shows the assignment of safety vectors and levels in a faulty 4-cube with two faulty links $110-$ (connecting two end nodes 1100 and 1101) and $00-0$ and one faulty node 1011. Note that the safety level of node 1110 is only 2 in Figure 2 which means that optimality is not guaranteed to forward a message from 1110 to a node which is more than 2 Hamming distance away. Actually, to forward a message from 1110 to 1001 a detour path (with a length of Hamming distance plus two) has to be used, because all of its preferred nodes' safety levels are lower than 2. Using the safety vector model, node 1110 has a safety vector (1,1,1,1) indicating a safe node, so there exists a Hamming distance path to any nodes in the hypercube. For example, if node 1001 is the destination node, the source sends the message to one of two preferred neighbors, 1100 or 1010, that has one at the 2nd bit of its safety vector. Assume node 1010 is selected which in turn forwards the message to one of its preferred neighbor which has one at the 1st bit. The path constructed is $1110 \rightarrow 1010 \rightarrow 1000 \rightarrow 1001$ as shown in Figure 2.

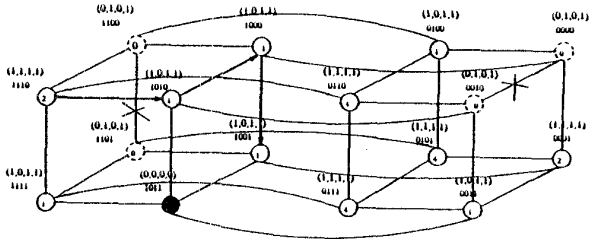

Figure 2: The safety vector and level assignment of a faulty 4-cube with faulty links

The proposed algorithm can also be used in disconnected hypercubes. If a source tries to send a message to a k-distance destination which is disconnected from the source, the source can detects such an infeasible routing based on its safety vector and neighbors'. More specifically, a 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 regular connected hypercube.

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 unicasting. (2). *Periodic*: each node exchanges safety information periodically with its neighbors. (3) *State-change-driven*: a node initiates the GS algorithm whenever it detects that a neighboring node or link fails (or recovers).

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 Figure 1 as our example. When one or more new faults occur, the message(s) in transit will not be affected unless one of the new faults is along the selected path from a source to a destination. Even when one of the new faults blocks the selected path at an intermediate node, say a, the current message at node a may still use other available preferred neighbors. In case all the 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 a system with no link fault, see the Corollary of Theorem 5). In either case, we have two possible ways to update safety vectors: (a) update after the completion of transmitting the current message (at node a), or (b) update immediately by blocking the current message at node a. In both case, we might still need to block other messages (include the one at node a) in transit. Because each message carries only the relative address (between the current node and the destination), the blocked messages can resume immediately after the completion of GS as new messages to be routed.

Consider the example of Figure 1 with two messages in transit: $(s_1, d_1) = (0001, 1100)$ with message m_1 and $(s_2, d_2) = (1110, 0001)$ with message m_2, assume that a new fault 0000 occurs when m_2 is at intermediate node 1101 and m_1 is just about to start. Node 0001 detects a new fault at neighbor 0000 and immediately blocks all the messages m_1 and m_2 (approach (b) is used here). After the update algorithm GS is completed, both m_1 and m_2 resume at nodes 0001 and 1101, respectively, as two new messages.

To determine the application range of the proposed method in terms of degree of fault tolerance, we consider two cases: fully applicable under certain fault distributions and fully applicable under any fault distribution. Theorem 5 and its corollary show the upper bound of faults to ensure fully applicability under any fault distribution. 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 in an n-cube. This is a very conservative approach, because certain fault distributions (which make the proposed method inapplicable) rarely occur. In other words, the proposed method still works for most cases (of fault distributions) even when the number of faults exceeds the upper bound (as in the 4-cube of Figure 1).

Feasibility checking at the source node determines whether it is applicable to use the proposed unicasting method. Actually, based on the unicasting_at_source_node the checking process for a H distance routing can be simply expressed as:

{at the source node}
if the $(H-1)$th bit of a preferred neighbor's safety vector is one or the $(H+1)$th bit of a spare neighbor's safety vector is one
then it is feasible to use the proposed algorithm
else it is unfeasible.

B. Deadlock freeness

Deadlock may happen when multiple nodes send their unicast messages simultaneously. Intuitively, deadlock occurs when messages traveling in the system develop *dependency loops* among themselves that prevent further movement. To avoid deadlock, we need to prevent the circular waiting situation. Deadlock avoidance also depends on the switching mechanism used. Here we use *wormhole routing* as an example. With wormhole routing, a message is decomposed into flits which are spread out among several channels as the message moves. Deadlock can be prevented by restricting the combination in which the input and output channels are connected.

Restricting the combination of input and output channels can be done in two ways: (a) restrict the use of channels in the existing network, and (b) "expand" the existing network by dividing each physical channel into multiple *virtual channels* [6] and then restrict the use of virtual channels in the virtual network. Type (a) approach corresponds to partially adaptive routing (where certain physical connections cannot be used). To maintain the degree of adaptivity, virtual channels are used which expand the existing network. The expanded network is then restricted to ensure the absence of dependence circle among virtual channels and at the same time all the possible physical channel connections are still fully used. In this way, adaptivity is preserved. The objective of the proposed routing algorithm is to maximize adaptivity in the presence of faults; therefore, type (b) approach is used.

Based on the Corollary of Theorem 6, the path length for the unicast message cannot be longer than $n+1$ in the proposed algorithm. Therefore, we can divide each physical channel into $n+1$ virtual channels number from 1 to $n+1$ in an n-dimensional hypercube. The $n+1$ levels of virtual channels are used in an increasing order in the n-cube. That is, the level-1 channels are used for the first hop in the routing. In general the level-k channels are used for the kth hop in the routing. The proposed routing is obviously deadlock-free with this implementation.

Because our dynamic adaptive routing uses a subset or the whole set (when there is no faults in the neighborhood) of the shortest paths used in a fully adaptive minimal routing. The following theorem shows that any results on fully adaptive minimal routing can be used in our case.

Theorem 7: *If a fully adaptive minimal routing algorithm for a fault-free n-cube uses $k(n)$ virtual channels, our proposed routing algorithm can be adapted that to ensure deadlock freenes using $k(n)+1$ virtual*

channels.

Proof: We number virtual channels used in the given fully adaptive minimal routing algorithm (called A) from 1 to $k(n)$. Normally $k(n)$ is a function of n, the dimension of the hypercube. The extra virtual channel is numbered $k(n)+1$. In the suboptimal routing, the first step is a detour and we use the level-$(k(n)+1)$ virtual channels. In all the other steps in both optimal and suboptimal routing, the selection of virtual channels will strictly follow the given fully adaptive minimal routing algorithm. Note that, at each intermediate node our proposed algorithm uses a subset of preferred neighbors used in algorithm A for the fault-free case. Because algorithm A is deadlock-free, there will be no cycle in the channel dependence graph of the proposed routing algorithm among level-1 to level-$k(n)$ virtual channels. Also, level-$k(n)+1$ channels are only used in the first step of the suboptimal routing. That is, only level-$(k(n)+1)$ virtual channels wait for channels among level-1 to level-$k(n)$. Therefore, there will still be no dependency loop among all the channels from level-1 to level-$(k(n)+1)$. □

5 Conclusions

We have proposed an adaptive unicasting algorithm for cube-based multicomputers. The algorithm uses limited global information captured by a safety vector of binary numbers associated with each node. The safety vector can be calculated through a simple $(n-1)$-round of information exchanges among neighboring nodes in an n-cube. A source node can easily decide to perform either an optimal or a suboptimal unicasting, based on its safety vector, its neighbors' safety vectors, and the Hamming distance between the source and the destination,

The proposed routing algorithm can be used together with other heuristic and/or greedy routing algorithms [10], such as *randomized routing* and *depth-first routing*. Such combination is especially efficient and useful in a system with many faults which result in a relatively small percentage of safe nodes in the system. When the source and neighbors' safety levels are too low to use the proposed routing algorithm, a heuristic or greedy routing algorithm is used until an intermediate node with a sufficiently high safety status is reached; then the proposed routing algorithm is used to guide the message to the destination through an optimal path. The use of safety vectors to implement reliable collective communication operations [5] will be our future work.

References

[1] Y. M. Boura and C. R. Das. Fault-tolerant routing in mesh networks. *Proc. of 1995 International Conferenc on Parallel Processing.* 1995, I 106 - I 109.

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

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

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

[5] P. K. McKinley, Y. J. Tasi, and D. F. Robinson. Collective communication in wormhole-routed massively parallel computers. *Computer.* 28, (12), 1995, 39-50.

[6] L. M. Ni and P. K. McKinley. A survey of routing techniques in wormhole networks. *Computer.* 26, (2), Feb. 1993, 62-76.

[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. Data communication in hypercubes. Tech. Rep. YALEU/DCS/RR-428, Dept. Comput. Sci., Yale Univ., June 1985.

[9] H. Sullivan, T. Bashkow, and D. Klappholz. A large scale, homogeneous, fully distributed parallel machine. *Proc. of 4th Annual Symposium on Computer Architecture.* March 1977, 105-124.

[10] L. Valiant. A scheme for fast parallel communication. *SIAM Journal on Computing.* 34, (1), May 1982, 350-361.

[11] J. Wu. Reliable communication in cube-based multicomputers using safety vectors. TR-CSE-95-24, Dept. of Computer Science and Engineering, FLorida Atlantic University, April 1995.

[12] J. Wu. Unicasting in faulty hypercubes using safety levels. *Proc. of the 1995 International Conference on Parallel Processing.* 1995, III 133- III 136.

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