Fault-Tolerant Communication in Cube-Based Multiple-Bus Systems

Jie Wu

Department of Computer Science and Engineering

Florida Atlantic University

Boca Raton, FL 33431

jie@cse.fau.edu

Abstract

We consider routing in multiple-bus systems with faulty nodes and buses. A cube-based multiple-bus system is used as an example to demonstrate our approach. In such a system, processors are partitioned into two sets based on the bipartite graph definition, one set is called a processor set and the other one becomes a bus set. A fault-tolerant routing scheme is proposed based on limited global information of fault distribution captured by an integer called safety level associated with each processor and bus. A safety level is an approximated measure of the number and distribution of faulty components in the neighborhood. Each processor (also called node) is considered as an active entity while each bus is viewed as a passive one. The safety level of each bus is calculated by one of its adjacent nodes. Optimal routing between two nodes is guaranteed if the safety level of the source node is no less than the Hamming distance between the source and destination nodes. The proposed routing scheme can also be used in disconnected cube-based multiple-bus systems, where nodes are disjointed. The feasibility of optimal or suboptimal routing can by easily determined at the source node by comparing its safety level, its neighbors' safety levels, together with the Hamming distance between the source and destination nodes. The proposed scheme is the first attempt to address fault-tolerant communication in a multiple-bus system.

Index Terms: *Fault tolerance, hypercubes, multiple-bus systems, reliable communication, routing*

1 Introduction

Interconnecting a large number of nodes is a major problem for the designer of a multicomputer system. There are many different and often conflicting performance and reliability considerations in choose a particular type of interconnection networks for a multicomputer system. *Multiple-bus architectures* ([6], [7]) can be cost effective due to their low cost, high communication bandwidth and graceful degradation in the presence of faults. In a multiple-bus network, nodes are connected through multiple shared buses.

Recently, the *hypergraph model* [11] has been used to relate the multiple-bus network to the traditional point-to-point network. In a hypergraph, a hyperlink (a bus in a multiple-bus network) connects two or more nodes. Therefore, hypergraph-based networks (also called *hypernetworks*) include the point-to-point network as a subclass. A multiple-bus system can be considered as a special implementation of the hypernetwork. Although multiple-bus systems were extensively studied ([2] and [5]), together with design issues and implementation aspects, very few papers exploited routing capability of such systems, especially in a faulty environment. In a multicomputer system with a large amount of nodes, fault tolerance capabilities become more important, including the one for one-to-one routing.

Efficient one-to-one routing (or simply routing) is a key to the performance of a multiple-bus network. Communication in fault-free multicomputer systems has been extensively studied. There have been many fault-tolerant routing algorithms proposed in various literature. However, almost all of them are based on the point-to-point network model. Most of them assume that each node knows either only neighbors' status or status of all the nodes. A model that uses the former assumption is called *local-information-based*, while a model that uses the later assumption is called *global-information-based.* Normally, a global-information-based model can obtain an optimal or suboptimal result; however, it requires a separate process to collect global information. In general, global information is presented in a tabular format and it is not easy to use. The local-information-based model uses a weaker but a more reasonable assumption; however, local information can only be used to achieve local optimization and most 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 different. A routing algorithm that uses *limited global information* is a compromise between local-information- and global-information-based approaches. Because this type of information is easy to update and maintain and the optimality is still preserved, it is more cost effective than the others.

In this paper, we study fault-tolerant routing in a multiple-bus network using limited global information. We use the cube-based multiple-bus system defined in this paper as an example. In [9], Wu proposed a novel concept called *safety level*, which is an integer associated with each node in a hypercube system. The safety level, a special type of limited global information, is a concise representation of the distribution of faulty nodes. Basically, each node in an n-dimensional hypercube (also called n-cube) is assigned a safety level k, where $0 \leq k \leq n$, and this node is called k-safe. A k-safe node indicates that there exists at least one Hamming distance path (i.e. the optimal path) from this node to any node within k Hamming distance. The safety level of each node can be calculated using a simple $(n-1)$-round iterative algorithm which is independent of the number and distribution of faults in the n-cube. Optimal routing between two nodes is guaranteed if the safety level of the source node is no less than the Hamming distance between these two nodes. The feasibility of optimal or suboptimal routing can by easily determined at the source node by comparing its safety level, its neighbors' safety levels, together with the Hamming distance between the source and destination nodes.

Optimal routing in multiple-bus networks is more complex than traditional point-to-point networks. In a multiple-bus network, each intermediate node (including the source node) has to select a bus among adjacent buses and then select an adjacent node (that shares the selected bus) as the next forwarding node. Nodes in a multiple-bus system are adjacent if they share a common bus. In fault-tolerant routing using the safety level concept, since each bus is a passive entity that cannot actively collect safety information of adjacent nodes, each node needs to know safety levels of adjacent buses and nodes; that is, each node needs to know safety status of neighbors (buses) and all the connected nodes (through these buses). In this study, we show how safety levels are calculated in cube-based multiple-bus systems and propose an optimal and a suboptimal fault-tolerant routing algorithms. Two fault models are used in this study, one considers only bus faults and the other includes both bus and nodes faults (the case with node faults only can be considered as a special case of the latter model).

Although the safety level as a special type of limited global information is used only for hypercubes and its variants, the approach used in collecting safety status of neighboring nodes can be applied to general systems and can be used in collecting other types of limited global information. Also, we have shown that the safety level concept can be used to achieve optimization or suboptimization in *collective communication operations* such as multicasting [10] and broadcasting [8] in a regular faulty hypercube. Collective communication operations, defined by the *Message Passing Interface* (MPI) standard [1], are fundamental in many applications and have been identified in many parallel languages. We believe that our approach could be a promising one in achieving fault-tolerant routing, including fault-tolerant collective communication, in general multiple-bus systems.

This paper is organized as follows: Section 2 defines some notation and preliminaries, where cube-based multiple-bus systems are defined. The concept of safety level and its application in achieving optimal and suboptimal fault-tolerant routing in regular hypercubes are reviewed. Section 3 proposes a method which determines the safety level of each node in a cube-based multiple-bus system. An optimal and a suboptimal routing algorithms using the safety level concept are proposed for cube-based multiple-bus systems. We show that the proposed algorithms can also be used in disconnected cube-based multiple-bus systems, where nodes are disconnected. Section 4 summaries our results and discusses some future work.

2 Notation and Preliminaries

2.1 n-dimensional hypercubes

An n-cube, Q_n, is a graph having 2^n nodes labeled from 0 to $2^n - 1$. Two nodes are joined by an edge (also called link) if their addresses, as binary numbers, differ in exactly one bit position. More specifically, every node a has an address $a_n a_{n-1} \cdots a_i \cdots a_1$ with $a_i \in \{0,1\}$, $1 \leq i \leq n$, and a_i is called the ith bit (dimension) of the address. We denote node a^i the neighbor of a along dimension i. Symbol $\oplus$ denotes the bitwise exclusive OR operation. Let $e^k = e_n e_{n-1} \ldots e_k \ldots e_1$ where $e_k = 1$, $1 \leq k \leq n$, and $e_j = 0, \forall j \neq k$. For example $1101 \oplus e^3 = 1001$. Clearly $a \oplus e^i$ represents setting or resetting the ith bit of node a. 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* (also called *Hamming distance 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*. Clearly, an optimal path is obtained by using links at each of these $H(s,d)$ preferred dimensions in some order. For example, suppose $s = 0101$ and $d = 1011$ then $s \oplus d = 0101 \oplus 1011 = 1110$. Therefore, dimensions 2, 3, 4 are preferred dimensions and dimension 1 is a spare dimension. Among neighbors of $s = 0101$, nodes 1101, 0001, and 0111 are preferred neighbors and node 0100 is a spare neighbor. Any path from $s = 0101$ to $d = 1011$ that uses links at dimensions 2, 3, and 4 in some order is an optimal path, e.g., $0101 \rightarrow 0001 \rightarrow 1001 \rightarrow 1011$ is an optimal path between 0101 and 1011.

2.2 Safety levels

The materials in this subsection and the next one come from [8] and [9]. In a given n-cube, the safety level of each node ranges from 0 to n. The safety level associated with a node is an approximation of the number and distribution of faulty nodes in the neighborhood, rather than just the number of faulty nodes. Let $S(a) = k$ be the safety status of node a, where k is referred to as the level of safety, and a is called *k-safe*. A faulty node is 0-safe which corresponds to the lowest level of safety, while an n-safe node (also called a *safe node*) corresponds to the highest level of safety. A node with k-safe status is called *unsafe* if $k \neq n$. Suppose seq_1 and seq_2 are two sequences of integer, $seq_1 \geq seq_2$ if and only if each element in seq_1 is greater than or equal to the corresponding element in seq_2.

Definition 1 [9]: *The safety level of a faulty node is* 0*. For a nonfaulty node a, let $(S_0, S_1, S_2, ..., S_{n-1})$, $0 \leq S_i \leq n$, be the nondecreasing safety level sequence of node a's n neighboring nodes in an n-cube, such that $S_i \leq S_{i+1}$, $0 \leq i < n-1$. The safety level of node a is defined as: If $(S_0, S_1, S_2, ..., S_{n-1}) \geq (0, 1, 2, ..., n-1)$, then $S(a) = n$ else if $(S_0, S_1, S_2, ..., S_{k-1}) \geq (0, 1, 2, ..., k-1) \wedge (S_k = k-1)$ then $S(a) = k$.*

The safety level of a nonfaulty node is recursively defined in terms of its neighbors' safety levels. It has been shown in [9] that for any given faulty hypercube, there is one and only one way to assign safety levels to nodes that satisfies the safety level definition. The iterative algorithm GLOBAL_STATUS (GS) calculates the safety level of each node in n-cube Q_n. We assume that all nonfaulty nodes have n as their initial safety levels.

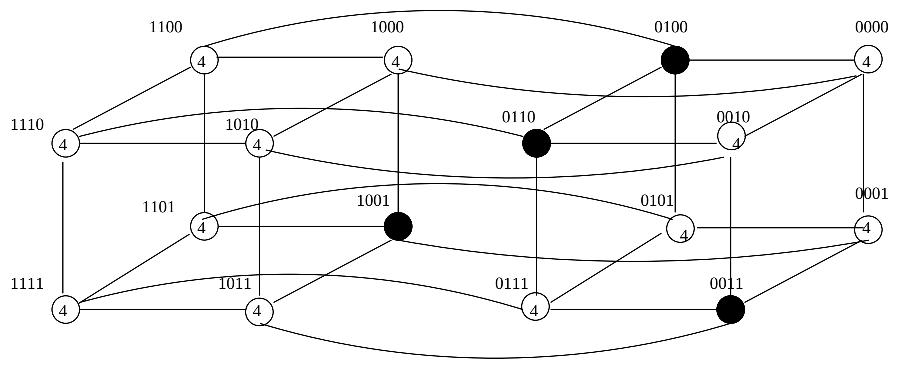
Figure 1 shows the safety level of each node in a faulty 4-cube with four faulty nodes: 0011, 0100, 0110, and 1001, represented as black nodes. Based on the safety level definition, the safety levels of all the nodes that have two (or more) faulty neighbors are changed to 1 after the first round, as in the case for nodes $0001, 0010, 0111, 1011$ in Figure 1 (b). That is, the effect of 0-safe status of faulty nodes first propagate to their neighbors, then neighbors' neighbors and so on. For example, after the second round the safety levels of nodes $0000, 0101$ change to 2 as shown in Figure 1 (c), because each node has two 1-safe neighbors and one faulty neighbor. The safety level of each node remains stable after two rounds and each value represents the final safety level of the corresponding node. It has been proved [8] that for any faulty n-cube, $n-1$ rounds of information exchanges are sufficient; that is, $\Delta = n-1$ in GS. There are several relevant properties of safety level summerized as follows:

Property 1 [8]: *In a faulty n-cube with no more than $n-1$ faulty nodes and with no faulty links, each nonfaulty but unsafe node has a safe neighbor.*

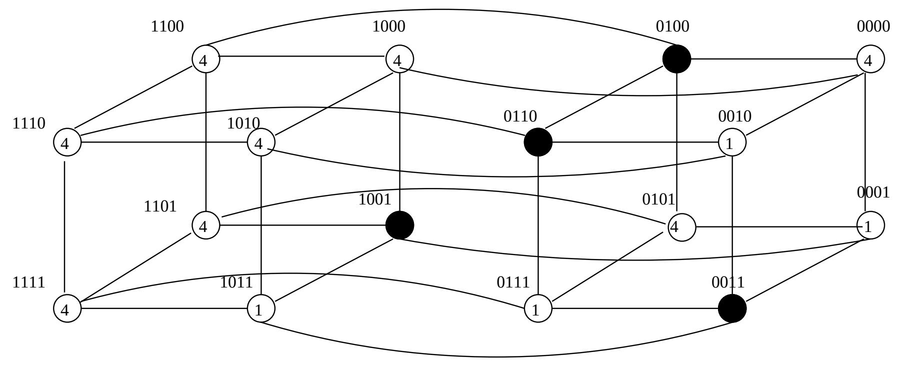
Property 2 [8]: *If the safety level of a node is k ($0 < k \leq n$), then there is at least one Hamming distance path from this node to any node within k Hamming distance.*

```
GLOBAL_STATUS (GS):
{ Initially all nonfaulty nodes are $n$-safe and $round = 1$}
begin
    while round $\leq \Delta$
      parbegin
          NODE_STATUS($a$), $\forall\ a \in Q_n$;
      parend;
      round := round +1;
    end while;
end.
```

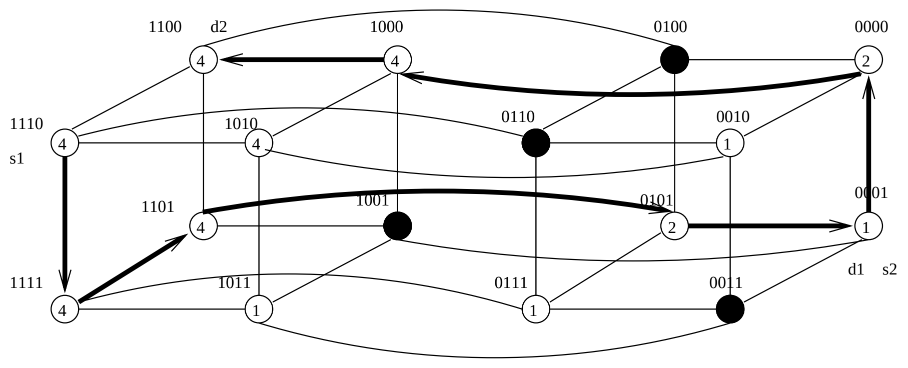
```
NODE_STATUS($a$):
begin
    at node $a$ determine a nondecreasing sequence of neighbors' safety levels $(S_0, S_1, S_2, ..., S_{n-1})$
    if $(S_0, S_1, S_2, ..., S_{n-1}) \geq (0, 1, 2, ..., n-1)$
       then mark $a$ as $n$-safe (or safe);
    if $((S_0, S_1, S_2, ..., S_{k-1}) \geq (0, 1, 2, ..., k-1)) \wedge (S_k = k-1)$
       then mark $a$ as $k$-safe;
end.
```



(a)



(b)



(c)

Figure 1: A 4-cube with four faulty nodes.

2.3 Reliable routing using safety levels

Based on Property 2 and the definition of safety level, it is easy to identify an optimal path if the safety level of the source node is no less than the Hamming distance between the source and destination nodes: A Hamming distance path is generated by selecting a preferred neighbor with the highest safety level at each routing step. If the safety level of the source node is less than the Hamming distance between the source and destination nodes and the number of faults is less than the dimension of the cube, based on Property 1 we can always forward the routing message to a safe neighbor and then initiate a routing from this safe neighbor. Obviously, if the safe neighbor is a preferred neighbor, the resultant path is still optimal; if the safety neighbor is a spare neighbor, the length of the resultant path is the Hamming distance between the source and destination nodes plus two (the corresponding routing algorithm is called suboptimal).

Consider the example in Figure 1 (c), where $s_1 = 1110$ and $d_1 = 0001$ are the source and destination nodes, respectively. A navigation vector, defined as the bitwise exclusive OR of s_1 and d_1, is calculated $N_1 = s_1 \oplus d_1 = 1111$; hence, the Hamming distance $H(s_1, d_1) = 4$. Also, the safety level of the source node s_1 is 4. Therefore, the optimal algorithm is applied. Among preferred neighbors of the source node, nodes $1010, 1100$, and 1111 have a safety level 4 and node 0110 has a safety level 0. A neighbor with the highest safety level, say 1111 along dimension 1, is selected. The navigation vector N_1 is sent together with the routing message after resetting bit 1. At intermediate node 1111, based on the navigation vector 1110, the preferred neighbor set is calculated which is $\{0111, 1011, 1101\}$. Among preferred neighbors, node 1101 has the highest safety level (which is 4); therefore, 1101 is the next intermediate node with navigation vector 1100. At node 1101, preferred neighbor 0101 (with a safety level 2) is selected among two preferred neighbors (the other one is faulty neighbor 1001). At node 0101 with navigation vector 0100, there is only one preferred neighbor which is node 0001. Upon receiving the routing message with navigation vector 0000, node 0001 identifies itself as the destination node and terminates the routing process.

Consider another routing example in the faulty 4-cube of Figure 1 (c), where $s_2 = 0001$ and $d_2 = 1100$ are the source and destination nodes, respectively. In this case, the safety level of the source node (which is 1) is less than the Hamming distance between the source and destination nodes (which is 3). However, there are two preferred neighbors (0000 and 0101) with a safety level of 2 (one less than the Hamming distance between the source and destination nodes). Therefore, optimal routing is still possible by selecting one of these two preferred neighbors, say node 0000. The corresponding routing path is $0001 \rightarrow 0000 \rightarrow 1000 \rightarrow 1100$ as shown in Figure 1 (c).

The optimal and suboptimal routing algorithms can also be applied in faulty n-cubes (including disconnected cubes) with even more than $n-1$ faulty nodes. Optimal routing can be used if the safety level of the source node is no less than the Hamming distance between the source and destination nodes, or one of the preferred neighbors' safety level is no less than the Hamming distance minus one. Suboptimal routing can be applied if one of the spared neighbors' safety level is more than the Hamming distance between the source and destination nodes.

2.4 Cube-based multiple-bus systems

In this section, we propose a multiple-bus system based on the hypercube topology. A multiple-bus system consists of three types of components: nodes, buses, and links. Each link connects one node to one bus. A node can be connected to many buses and vice versa. In a cube-based multiple-bus system, nodes in a hypercube are divided into two disjointed subsets, one is the node set and the other is the bus set. More formally, we represent a hypercube by a bipartite graph $(V(G), E(G))$, where $V(G)$ is the set of nodes and $E(G)$ is the set of links. The node set is partitioned into two subsets (node set and bus set) such that no two nodes in the same subset are adjacent. Links are used to connect nodes and buses.

Figure 2 (a) shows a regular faulty 3-cube and Figure 2 (b) shows the corresponding cube-based multiple-bus system. Without loss of generality, we use nodes with an odd number of 1-bits in their addresses as nodes and nodes with an even number of 1-bits in their addresses as buses. In the example of Figure 2, $\{001, 010, 100, 111\}$ forms a node set where each node is represented by a double cycle and $\{000, 011, 101, 110\}$ constitutes a bus set in which each node is represented by a cycle in Figure 2 (a) and a bus in Figure 2 (b). Again black nodes represent faulty nodes as shown in Figure 2 in which all faulty nodes are in the bus set. A node and a bus are connected via a link iff their addresses differ in exactly one bit. Two nodes are adjacent, i.e., they share a common bus, iff their addresses differ in exactly two bits. Figure 2 (b) also shows the safety level of each node and bus. Each link can be considered as part of the bus it connects. In this way, there are only two types of components: buses and nodes.

There are several interesting topological properties of an n-cube-based multiple-bus system. Most of them can be easily derived from hypercube properties. For example, each node is connected to n buses and each bus is connected to n nodes. If the distance between two adjacent nodes is measured as one unit, then the diameter of an n-cube-based multiple-bus system is $\lfloor \frac{n}{2} \rfloor$.

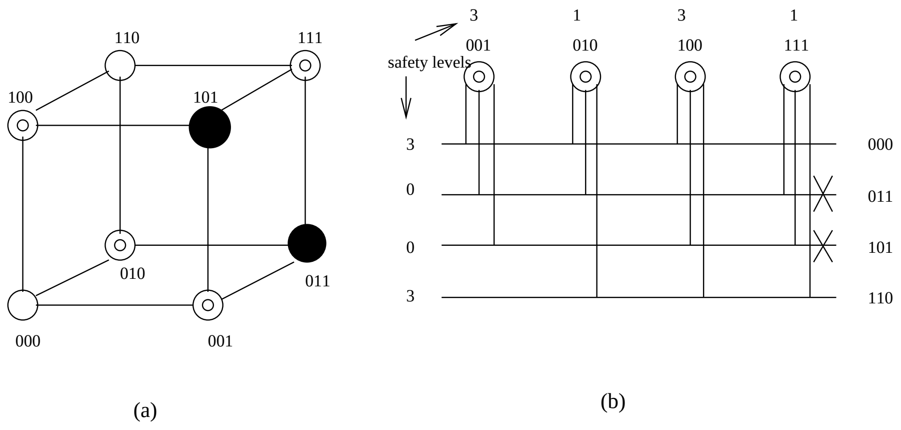


Figure 2: A faulty 3-cube (a) and the corresponding cube-based multiple-bus system (b).

3 Fault-Tolerant Routing in Cube-Based Multiple-Bus Systems

3.1 Collecting safety level information

Fault-tolerant routing in a multiple-bus network is much more complex than the one in a regular point-to-point network for the following reasons.

In a point-to-point network each intermediate node selects one out of n neighbors (in a network with maximum node degree n) as its next forwarding node. For example, in an n-cube, each intermediate node selects one node from its n neighbors. In a multiple-bus network, however, each intermediate node first selects an adjacent bus (out of l connected buses) and then selects an adjacent node (out of m adjacent nodes) along the selected bus. Therefore, in a multiple-bus network in which each node is connected to l buses and each bus is connected to m nodes, each intermediate node has up to $l \times m$ options in selecting the next forwarding node. For example, in an n-cube-based multiple-bus system, $l = m = n$; that is, each intermediate node has an order of $O(n^2)$ options.

Nodes are active entities, i.e., they can store data and make intelligent decisions (such as selecting the next forwarding node). Buses are normally passive entities, i.e., they are just communication media and cannot permanently store data and make (routing) decisions. Selecting an appropriate adjacent bus at each intermediate node is not sufficient, because the selected bus cannot make a decision as to which adjacent node the message should be forwarded to. The passive nature of buses also causes the problem of collecting safety level information. For example, each bus cannot be actively involved in collecting the safety status of its connected nodes in order to decide its own safety status. Therefore, the safety status of each bus has to be calculated and maintained by one (or more) of its connected nodes. To balance workload, each node should be responsible, if possible, for supplying safety level information for the same number of adjacent buses.

We consider first a system with only bus faults. The first step is to calculate safety levels of all the components (buses and nodes). In the cube-based multiple-bus system, we define the concept of *buddy pair* in which a node and a bus with the same address except the first bit form a buddy pair. For example, in the 3-cube as shown Figure 2 (b), $\{000, 001\}, \{010, 011\}, \{100, 101\}, \{110, 111\}$ are four buddy pairs. In calculating and distributing safety level information, each node is responsible for safety level calculation and distribution of its buddy (bus). An extended GS algorithm called EGS is defined, (a, a^1) is a buddy pair where a is a node and a^1 is its buddy bus, i.e., the neighbor along dimension 1 in a regular hypercube. The result of EGS generates a *safety matrix* associated with each node. Each safety matrix of a node contains safety information of nodes and buses within 2 Hamming distance of this node.

EGS needs $O(n)$ rounds of information exchanges by calling EXTENDED_NODE_STATUS(a) and BROADCAST(a), each of which needs $O(n)$ bus cycles, since $O(n)$ messages are transferred on each bus. Therefore, a total of $O(n^2)$ steps are needed for EGS. Applying the EXTENDED_NODE_STATUS algorithm to node 001 of Figure 2, we have $ls[1] = 1$ (001's safety level) and $bs[1] = 3$ (001's buddy 000's safety level). $ls[2] = ls[3] = 0$ are 011 and 101's safety levels. $bs[2] = bs[3] = 3$ are 000's neighbors' safety levels. $S[i, j]$ of a node includes safety levels of the adjacent buses and nodes. That is, safety levels of nodes and buses within 2 Hamming distance. $S[i, j]$ at each node a is determined through BROADCAST(a). Note that in a faulty cube-based multiple-bus system, a node may not be able to obtain safety levels of all its adjacent nodes (through shared buses). In the example of Figure 2, node 001 cannot get the status of node 111, since both shared buses 101 and 011 are faulty. It can be proved that through a two-round broadcasting process, each node can get status of all its adjacent nodes provided the number of faulty buses is less than the dimension of the corresponding cube. The first round is the step (1) of BROADCAST(a) and during the second round, each node broadcasts safety level information, received in the first round, along adjacent buses. However, as we will see later, there is no need to obtain status of all the adjacent nodes. Therefore, step (1) of BROADCAST(a) is sufficient. In $S[i, j]$ of a node a, although the safety level of node a is not directly included in $S[i, j]$, it can be easily derived from its neighbors' safety levels.

Figure 3 (a) and (b) shows two safety matrices associated with nodes 001 and 100 in Figure 2 (b), respectively. Note that the numbers in the first column are safety levels of adjacent buses and numbers in the ith row (except the first column) represent safety status of adjacent nodes (that share the bus along the ith dimension). More specifically, suppose $S[i, j]$ is the safety matrix of node a, $S[i, 1]$ is the

```
EXTENDED_GLOBAL_STATUS (EGS):
{ Initially all nonfaulty nodes are $n$-safe and $round = 1$}
begin
    while round $\leq \lfloor \frac{n}{2} \rfloor$
        parbegin
            EXTENDED_NODE_STATUS($a$), $\forall$ $a$ in the node set;
        parend;
        round := round +1;
    end while;
    parbegin
        BROADCAST(a), $\forall$ $a$ in the node set;
    parend
end.
```

```
EXTENDED_NODE_STATUS($a$):
begin
    initial safety levels of $a$ and its buddy are stored in $ls[1]$ and $bs[1]$;
    (1) send $ls[1]$ to node$(a^i)^1$ along bus $a^i$, $2 \leq i \leq n$;
        /* forward $a$'s safety level to $(a^i)^1$, bus $a^i$'s buddy node */
        broadcast $bs[1]$ along bus $a^1$;
        /* broadcast the safety level of its buddy to buddy's neighboring nodes */
    (2) receive one message from bus $a^i$ $(2 \leq i \leq n)$ and store it in $ls[i]$;
        receive $n-1$ messages from bus $a^1$ and store them appropriately in $bs[i]$ $(1 \leq i \leq n)$;
        /* $ls[i]$ ($bs[i]$) stores $a$'s (resp. $a^1$'s) neighbors' safety status */
    (3) update $ls[1]$ based on $bs[1], ls[2], ls[3], ..., ls[n]$ and $bs[1]$ based on $ls[1], bs[2], bs[3], ..., bs[n]$;
        /* two updates are based on the safety level definition */
end.
```

```
BROADCAST($a$):
begin
    (1) node $a$ broadcasts $ls[1]$ and $bs[1]$ along bus $a^i$, $1 \leq i \leq n$;
    (2) receive broadcast messages and store them in such a way that $S[i,1]$ is the safety level of
        bus $a^i$ and $S[i,j]$ is the safety level of the adjacent node of bus $a^i$ along dimension $j-1$.
        /* $S[i,j]$ $(1 \leq i \leq n, 1 \leq j \leq n+1)$ is the safety matrix associated with node $a$ */
end.
```

	j=1	2	3	4
i=1	3	-	3	3
2	0	3	-	*
3	0	3	*	-

(a)

	j=1	2	3	4
i=1	0	-	1	1
2	3	1	-	3
3	3	1	3	-

(b)

Figure 3: Safety matrices $S[i, j]$ associated with (a) node 001 in Figure 2 (b) with a safety level 1 and (b) node 100 in Figure 2 (b) with a safety level 3.

safety level of the adjacent bus, a^i, along the ith dimension. $S[i, j]$ ($1 \leq i \leq n$, $2 \leq j \leq n + 1$) is the safety level of the adjacent node $(a^i)^{j-1}$ (through the adjacent bus a^i). Clearly when $i = j - 1$, node $(a^i)^{j-1}$ is node a itself. Therefore, we put - (a don't-care symbol) in entries $S[i, j]$ where $i = j - 1$. If there is an adjacent node for which its safety level cannot be determined, we put symbol $*$ in the corresponding entry in $S[i, j]$.

In example of Figure 3, the safety levels of 001 and 100 are 1 and 3 based on the safety levels of their adjacent buses, respectively. In Figure 3 (a), entries $S[3, 3]$ and $S[2, 4]$ are empty (with symbol $*$) since node 001 cannot obtain the safety level of node 111. Clearly, node 100 can reach any other node in one (bus) step. In the worst case, node 001 needs one more step by forwarding the message to the safe adjacent bus 000 along dimension 1. Note that if we remove the first column of a safety matrix, the resultant safety matrix is symmetric with respect to the principal diagonal and this is can be easily shown using the fact that $(a^i)^j = (a^j)^i$. Therefore, we only need to keep either an upper-triangular or lower-triangular matrix to save about half of memory space.

3.2 Routing in cube-based multiple-bus systems with bus faults

The performance of a routing process is measured by the number of buses traversed and this number is called *bus steps* or simplely steps. The minimum number of steps between two nodes s and d is equal to the Hamming distance between their binary addresses divided by two; that is, $\frac{H(s,d)}{2}$. A path with a minimum number of steps is called an *optimal path.* A path with one more step than the minimum number is called a *suboptimal path.* Our approach here ensures that each routing path is either optimal or suboptimal as long as the number of faults does not exceed certain threshold. In the remaining discussion, we consider two fault models: One with only bus faults and the other with both bus and node faults (the case with node faults only is covered by the later case). Note that the routing algorithms are the same for both models, they differ in the way safety levels are calculated.

The routing in the cube-based multiple-bus system can be done in a similar way as in a regular hypercube. Just treat all the buses and nodes as regular nodes in a regular hypercube. Therefore, each adjacent bus (node) is either a preferred or spare neighbor with respect to an intermediate (or source) node depending on the location of the intermediate and destination nodes. For example, bus 000 is a spare neighbor of node 001 if the destination node is 111, while bus 101 is a preferred neighbor of node 001 for the same destination node. The major difference is that each node has to select a neighbor (a bus) and the neighbor's neighbor (a node); that is, an intermediate node first selects a preferred adjacent bus with a highest safety level, and then picks a preferred adjacent node with the highest safety level connected by the selected bus.

Let us examine several routing examples. Suppose nodes 100 and 001 are the source and destination nodes in the example of Figure 2 (b). Since $100 \oplus 001 = 101$, preferred neighbors of node 100 are buses along dimensions 1 and 3, that is, buses 101 and 000. Note that the first column of the safety matrix (Figure 3 (b)) provides all the safety level information of neighboring buses. Among two preferred neighbors (buses), the one along dimension 1 has a safety level 0 and the one along dimension 3 has a safety level 3. Clearly the bus along dimension 3 is selected. To select a preferred adjacent node connected via the selected bus, we only need to examine the safety levels in the corresponding row of the associated safety matrix. In this case, there is only one preferred node (along dimension 1). The resultant routing path is: 100 (node) $\rightarrow$ 000 (bus) $\rightarrow$ 001 (node). As an another example, assume that nodes 001 and 111 are the source and destination nodes, respectively. Since $001 \oplus 111 = 110$, preferred neighbors of node 001 are buses 101 and 011 and the Hamming distance between the source and destination nodes is 2. Unfortunately, both preferred buses are faulty with a safety level of 0, optimal routing is not possible. However, the safety level of the spare bus 000 is 3, which is one more than the Hamming distance; that is, suboptimal is possible. In this case, the routing message is routed to either node 100 or node 010 (both have the same safety level 3) and the destination node 111 is reached with one extra step via bus 110. Therefore, a total of 2 bus steps (4 regular steps) are used in this case. Note that the size of each safety matrix is $O(n^2)$; however, the complexity of the table lookup process at each intermediate step is $O(n)$. Actually, exactly $2n$ operations are needed, where first n operations are used to select the adjacent bus by examining the first column of the safety matrix and another n operations are used to select the adjacent node in the selected row of the safety matrix. Formally, the routing algorithm is described in UNICASTING_AT_SOURCE_NODE and UNICASTING_AT_INTERMEDIATE_NODE, where $S[i, j]$ ($1 \leq i \leq n$ and $1 \leq j \leq n+1$) and ls are the safety matrix and the safety level associated with each node, respectively.

Theorem 1: *If the safety level of the source node is no less than the Hamming distance (H) between the source and destination nodes, then the length of the resultant routing path is $\frac{H}{2}$ bus steps which is optimal.*

The proof can be easily done by using Theorem 5 from [8] and the proposed routing algorithm. The routing algorithm in a cube-based multiple-bus system has the same complexity as in a regular hypercube. Although $2n$ operations are needed in table lookup at each intermediate node, the total number of steps is reduced by half. Actually, two steps in a regular hypercube are combined into one (bus) step in a cube-based multiple-bus system. However, routing in a cube-based multiple-bus system needs more memory space for safety matrices $O(n^2)$ at each node than for safety levels $O(n)$ in a regular hypercube. Again, this is not surprising, since each node in a cube-based multiple-bus system needs to know the status of components within 2 Hamming distance.

When the source node's safety level is less than the Hamming distance between the source and destination nodes, if the total number of bus faults is less than the dimension of the corresponding cube, the source node still has a safe adjacent bus (Property 1), i.e., any destination node is reachable from this bus through an optimal path. Therefore, suboptimality is guaranteed.

Theorem 2: *If there are at most $n-1$ faulty buses in an n-cube-based multiple-bus system, the proposed routing algorithm is at least suboptimal; that is, the length of the resultant path is either $\frac{H}{2}$ or $\frac{H}{2}+1$, where H is the Hamming distance between the source and destination nodes.*

The proof to this theorem can be derived directly from Theorem 1 and Property 1.

3.3 Routing in cube-based multiple-bus systems with both bus and node faults

In this subsection we consider routing in cube-based multiple-bus systems with both bus and node faults. Because buses and nodes are both nodes in a regular hypercube, the safety-level-based routing can be directly applied once the safety level of each node and bus is identified. However, nodes (the active entities) may fail while their buddy buses are still functioning. Then who are responsible for calculating and distributing the safety levels of these buses?

There are two possible solutions to this problem:

- The first solution tries to get around the problem by building buses as active entities. This can be done by a bus controller capable of monitoring the (safety) status of connected nodes and

At source node s with routing message m and destination node d:

```
UNICASTING_AT_SOURCE_NODE:
    $N = s \oplus d$; $H = H(s,d)$;
    /* calculate the navigation vector and the Hamming distance */
    if $(C1 : ls \geq H) \vee (C_2 : \exists i(S[i,1] \geq H-1 \ \wedge N(i) = 1))$
          / * the safety level of the source node is at least equal to
          the Hamming distance or a preferred neighbor's safety level
          is at least equal to the Hamming distance minus one */
    then OPTIMAL_UNICASTING:
          send $(m, N \oplus e^i \oplus e^j)$ to the adjacent node $(s^i)^j$ via the adjacent bus $s^i$, where
          $S[i,1] = \max_{1 \leq k \leq n}\{S[k,1] | N(k) = 1\}$ and $S[i,j] = \max_{2 \leq k \leq n+1}\{S[i,k] | N(k) = 1\}$
          /* send message $m$ to the preferred adjacent node $(s^i)^j$ via the
          preferred adjacent bus $s^i$ together with N after resetting bits $i$ and $j$ */
    else  if $C_3 : \exists\ i(S[i,1] \geq H+1 \wedge N_i = 0)$
          /* one spare neighbor's safety level is at least
          equal to the Hamming distance plus one */
          then SUBOPTIMAL_UNICASTING:
          send $(m, N \oplus e^i \oplus e^j)$ to the adjacent node $(s^i)^j$ via the adjacent bus $s^i$, where
          $S[i,1] = \max_{1 \leq k \leq n}\{S[k,1] | N(k) = 1\}$ and $S[i,j] = \max_{2 \leq k \leq n+1}\{S[i,k] | N(k) = 1\}$
    else failure
```

At any intermediate node a with routing message m and navigation vector N:

```
UNICASTING_AT_INTERMEDIATE_NODE
    if $N = 0$ /* the navigation vector is empty */
    then stop /* the currect node is the destination node */
    else  send $(m, N \oplus e^i \oplus e^j)$ to the adjacent node $(a^i)^j$ via the adjacent bus $a^i$, where
          $S[i,1] = \max_{1 \leq k \leq n}\{S[k,1] | N(k) = 1\}$ and $S[i,j] = \max_{2 \leq k \leq n+1}\{S[i,k] | N(k) = 1\}$
```

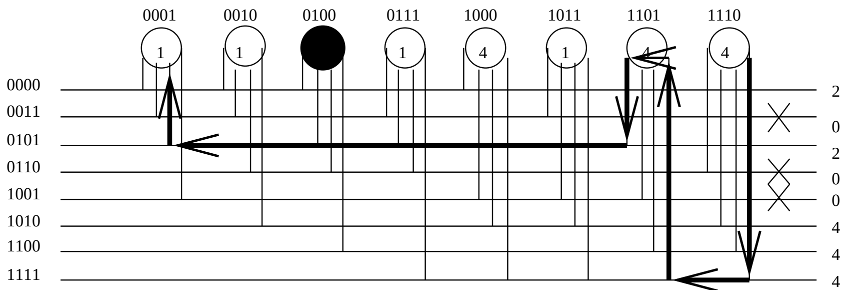


Figure 4: A 4-cube-based multiple-bus system.

store safety information. In this way, each bus can calculate and maintain its own safety status and the *GS* algorithm can be directly applied without modification.

- The second solution still treats each bus as a passive entity. When the buddy node of a bus fails, an *election* process is carried out among all the nodes connected through this bus. The elected node is the new buddy of the bus.

We focus here on the second solution since the first one is straightforward. Election [4] is a general style of computation in parallel/distributed systems, in which one node from the node set is selected to perform a particular task. In the routing example, after a node failure occurs, it is often necessary to reorganize buddy pairs so that one healthy node can take over the buddy of the new faulty node. A simple broadcast along buses that connect the faulty node can be designed to elect a new buddy node based on *a priori* priority among connected nodes by these buses. Also, each adjacent node is informed (through another round of broadcast) the id of the newly elected buddy.

Because each node is connected to more than one bus, it is possible that one node is selected as a new buddy for many buses, while another one is the buddy for just one bus. That is, each node may have different workloads of calculating safety levels of adjacent buses. Thus it is desirable to have an algorithm to modify buddy pairs in case of node failures while still retain the original features of load sharing among nodes. The concept of *preferred list* [3] can be used here: Each bus is equipped with a backup list of its most preferred buddy nodes. Whenever its current buddy node fails, its next most preferred node will be the new buddy. The preferred lists are designed in such a way that the chance for each healthy node being selected as the new buddy of a bus is almost the same. A detailed discussion and implementation of the preferred list concept can be found in [3].

Table 1 shows an example of preferred lists of all the buses in a 4-cube, where bit sequences at the first row represent bus addresses and integers are preferred node addresses in decimal. Nodes in a

	j=1	2	3	4	5
i= 1	4	-	4	1	1
2	4	4	-	4	0
3	4	1	4	-	1
4	0	1	0	1	-

(a)

	j=1	2	3	4	5
i= 1	4	-	4	4	0
2	4	4	-	1	1
3	0	4	1	-	1
4	2	0	1	1	-

(b)

Figure 5: Safety matrices associated with nodes (a) 1110 and (b) 1101 of Figure 4.

preferred list are ordered by preference. For example, the preferred list of bus 0000 is node 1 (0001). The rest of the nodes in the buddy set of bus 0000 (in the decreasing preference order) are: nodes 2 (0010), 4 (0100), and 8 (1000). Figure 4 shows the cube-based multiple-bus system of Figure 1. In this case, buses 0011, 0110, and 1001 are faulty and node 0100 is faulty. Based on Table 1, node 0100 (4) will be replaced by node 0111 (7) as the new buddy of bus 0101 and all the other buddy pairs remain unchanged. Once buddy pairs are formed, EGS can be easily modified to identify safety levels of all the components (nodes and buses). Figure 4 shows a routing process from node 1110 to node 0001 based on safety levels. Figure 5 shows two safety matrices associated with nodes 1110 and 1101 in Figure 4, respectively. Because $1110 \oplus 0001 = 1111$, all adjacent buses of node 1110 are preferred ones and three of them have the same safety level of 4 (see Figure 5 (a)). Assume that the bus along dimension 1 is selected, by inspecting the first row of Figure 5 (a), the adjacent node along dimension 2 has the highest safety level out of three possible choices (along preferred dimensions 2, 3, and 4). That is, node 1101 is the selected intermediate node that forwards the routing message. At node 1101, because $1101 \oplus 0001 = 1100$, only dimensions 3 and 4 are preferred ones. Obviously, the adjacent bus along dimension 4 (with safety level 2) is selected based on the first column of Figure 5 (b) and the corresponding preferred adjacent node is the destination node 0001.

Both Theorems 1 and 2 still apply to cube-based multiple-bus systems with both bus and node faults (the number of bus faults is now changed to the number of bus and node faults). The proof is straightforward.

Table 1: Preferred lists of buses in a 4-cube.

preference	0000	0011	0101	0110	1001	1010	1100	1111
I	1	2	4	7	8	11	13	14
II	2	1	7	4	11	8	14	13
III	4	7	1	2	13	14	8	11
VI	8	11	13	14	1	2	4	7

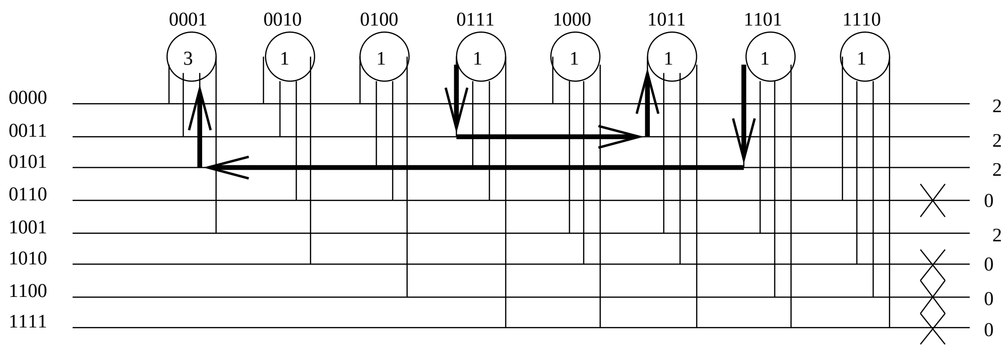


Figure 6: A disconnected 4-cube-based multiple-bus system.

3.4 Disconnected cube-based multiple-bus systems

In this section, we show that the proposed routing scheme can be applied to various faulty cube-based multiple-bus systems, including disconnected ones. To our best knowledge, our approach is the first one that addresses routing in disconnected multiple-bus systems.

We illustrate this approach by considering a disconnected cube-based multiple-bus example. Figure 6 shows a disconnected cube-based multiple-bus system with four faulty buses: 0110, 1010, 1100, and 1111. Clearly, any routing initiated at node 1110 will fail. The source node detects this by checking its adjacent bus' safety levels and its own safety level. However, routing is possible if it is initiated from the other part of the partition. For example, in a routing process with the source node $s_1 = 1101$ and the destination node $d_1 = 0001$, the Hamming distance between the source and destination nodes is 2 and the safety level of the source node is 2. Therefore, optimal routing is possible and the corresponding path is shown in Figure 6. In another example, where $s_2 = 0111$ and $d_2 = 1011$, although the safety level of the source node (which is 1) is less than the Hamming distance (which is 2), the preferred bus 0011's safety level is 2, which is more than the the Hamming distance minus one. Again optimal routing is possible in this case (see Figure 6). When the destination node is 1110, any routing will fail and this can be easily detected at the source node, say 0111. At node 0111, the safety level of the source node is 1, which is less than the Hamming distance $H(0111, 1110) = 2$ (condition C_1 of the routing algorithm fails) and none of the preferred neighbors' (0110 and 1111) safety level is more than the Hamming distance minus one (condition C_2 fails), hence there is no optimal routing. Both spare neighbors' (0101 and 0011) safety levels are 2 which is less than the Hamming distance plus one (condition C_3 also fails), and hence there is no suboptimal routing. Therefore, this routing process is aborted at the source node.

4 Conclusions

We have proposed an optimal and a suboptimal routing algorithms for cube-based multiple-bus systems. These algorithm use limited global information captured by a safety level associated with each node and bus. The safety level can be calculated through a simple $\lfloor \frac{n}{2} \rfloor$-round of information exchanges among adjacent nodes in an n-cube. The source node can easily decide to perform either optimal or suboptimal routing, based on its safety level, its neighbors' (bus's) safety levels, and the Hamming distance between the source and destination nodes, A source node can also identify cases when optimal and suboptimal paths are blocked by faulty nodes and buses or when the corresponding routing tries to forward a message to another part in a disconnected system. The proposed approach is the first attempt to address fault-tolerant routing in multiple-bus systems. Our next step is to generalize this approach to a more general class of multiple-bus systems and to study fault-tolerant collective communication in such systems.

References

[1] Message passing interface forum. *MPI: A Message-Passing Interface Standard.* May 1994.

[2] D. Bulka and J. B. Dugan. Design and analysis of multibus systems using projective geometry. *Proc. of the 22th International Symposium on Fault-Tolerant Computing.* 1992, 122-129.

[3] Y. C. Chang and K. G. Shin. Load sharing in hypercube multicomputers in the presence of node failures. *Proc. of the 21st International Symposium on Fault-Tolerant Computing.* 1991, 188-195.

[4] H. Garcia-Molina. Elections in a distributed computing system. *IEEE Transactions on Computers.* 31, (1), Jan. 1982, 48-59.

[5] H. K. Ku and J. P. Hayes. Connectivity and fault tolerance of multiple-bus systems. *Proc. of 24th International Symposium on Fault-Tolerant Computing.* 1994, 372-381.

[6] T. Lang, M. Valero, and I. Alegre. Bandwidth of crossbar and multiple-bus connections for multiprocessors. *IEEE Transactions on Computers.* 31, 1982, 1227-1234.

[7] L. D. Wittie. Communication structures for large networks of microcomputers computers. *IEEE Transactions on Computers.* 41, 1981, 264-273.

[8] J. Wu. Safety levels – an efficient mechanism for achieving reliable broadcasting in hypercubes. *IEEE Transactions on Computers.* 44, (5), May 1995, 702-705.

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

[10] J. Wu and K. Yao. Fault-tolerant multicasting in hypercubes using limited global information. *IEEE Transactions on Computers.* 44, (9), Sept. 1995, 1162-1166.

[11] S. Q. Zheng. Sparse hypernetworks based on steiner triple systems. *Proc. of the 1995 Internaitonal Conference on Parallel Processing.* 1995, I 92- I 95.