# A Limited-Global Information Model for Fault-Tolerant Routing in Dual-Cubes

Zhen Jiang
Dept. of Computer Science
West Chester University
West Chester, PA 19383
zjiang@wcupa.edu

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

## 1 Introduction

The binary hypercube has been widely used as the interconnection network in a variety of parallel systems such as Intel iPSC, SGI Origin 2000, nCUBE, and Connection Machine CM-2. To extend the hypercube network such that the size of the network can be increased with the limited number of links per node, Li [1] proposed a new interconnection network, called *dual-cube*. An $r$-connected dual-cube consists of $2^{r+1}$ connected r-cubes (also called clusters). The cube links are called cube-edges. All the clusters are connected through the extra links, called cross edges. Each node in a cluster has one and only one cross edge connecting with a node in another cluster. In such a system, efficient communication among the processors is critical to the performance of the system. Hence, the routing of messages is an important issue that needs to be addressed. As the number of nodes in a multicomputer system increases, the chance of failure also increases. The complex nature of networks also makes them vulnerable to disturbances which can be either deliberate or accidental. Therefore, the ability to route messages efficiently in the presence of faulty components is becoming increasingly important.

A central issue in designing a fault-tolerant routing algorithm in networks is the way fault information is collected and used. *Limited-global-information-based* routing is a compromise between local-information-based and global-information-based approaches. In this approach, fault information is collected and packed to achieve a global approximation of the number and distribution of faulty components based on a special coding scheme. In the *safety level* model [3], an integer is associated with each node in an $n$-cube representing the limited global information in the system. In the *safety vector* model [2], each node is associated with a binary vector which is a refinement of the safety level model. It can describe the distribution of faults more precisely. Because the limited global information is easy to update and maintain and the optimality is still preserved, it is more cost effective than the others. In dual-cubes, the number of links per node is limited as the number of nodes increases. And it is less than the distance between a source $s$ and its destination $d$ in most cases. Thus, the safety level and safety vector models cannot be applied directly to dual-cubes.

In this paper, we focus on the minimal path fault-tolerant routing without adding any extra link. First, a depth-first search routing is provided. Unlike the routing in [1], it can handle more faulty components, even when the network is disconnected. And then, it is extended by using our limited global information model. We use limited-safety-level and limited-safety-vector to represent our limited global information in dual-cubes. In a given dual-cube, the limited-safety-level (or the limited-safety-vector) of each node $u$ is its safety level (or safety vector) of the local cluster (an $r$-cube). Each cluster cube maintains its own safety level and safety vector information just like in a regular cube. Adjacent clusters exchange their safety information through the cross-edge to approximate their global safety information for the routing process. Faulty nodes and faulty links are both considered in this paper. In each local cluster cube, the minimal path routing will be guaranteed based on our limited global information. We propose the whole routing process by using segments of minimal routing paths. Compared with the depth-first search routing based on neighbor information, the routing based on limited global information needs fewer extra steps by using several rounds of neighbor information exchanges for each new fault configuration. The simulation results show that the limited global information model can help our routing process to generate a minimal path or a sub-minimal path (a path with only two extra steps).

## 2 Preliminaries

**Dual-cube.** An $r$-connected dual-cube $F_r$ consists of two classes (class 0 and class 1) and each class has $2^r$ clusters. Within a cluster, $2^r$ nodes with $r$ links per node form an $r$-cube. The cube links are called *cube-edges*. All the clusters

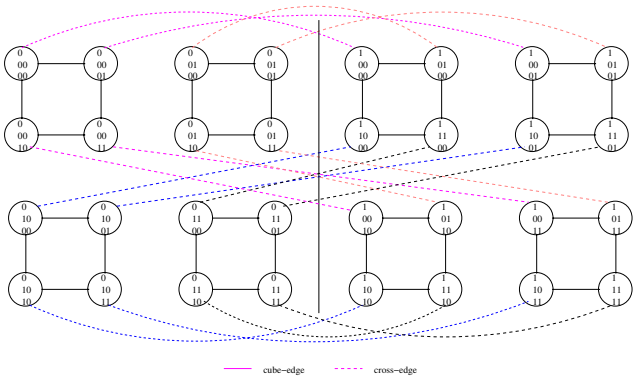


**Figure 1. An $F_2$ sample.**

are connected by the extra links between different classes, called *cross-edges*. Each node in a cluster has one and only one cross-edge. All $2^{2r+1}$ nodes in an $F_r$ are labeled from 0 to $2^{2r+1}$. We use $(c, r[1], r[0])$ to denote a node address. $c\,(\in \{0,1\})$ is one bit $class\_id$, which defines the class of a node. $r[c]$ is $r$-bit $node\_id$ and $r[c \oplus 1]$ is $r$-bit $cluster\_id$. Symbol $\oplus$ denotes the bitwise exclusive OR operation on binary addresses. Two nodes whose addresses differ only in one bit position in $node\_id$ are connected by cube-edge. And two nodes whose addresses differ only in $class\_id$ are connected by cross-edge. Therefore, a node in $F_r$ has $r+1$ links: $r$ links construct a low-level $r$-cube and one link constructs a high-level connection. It is noted that for a node with different value of $c$, its field position ($node\_id$) in the address is different and the dimensions of its cube-edges are different.

An $F_2$ is shown in Figure 1. Nodes 01000, 01001, 01010, and 01011 form a 2-cube 010**. This cube is a cluster in class 0. Each node inside this 2-cube, for example, node 01001, has two cube-edges (solid links in Figure 1 connecting with nodes 01000 and 01011) and one cross-edge (dash link in Figure 1 connecting with node 11001).

**Routing in a fault-free dual-cube.** For each routing message, assume node $u$ is the current node, $s$ is the source node, $d$ is the destination node, and $v$ is a neighbor of node $u$ connected by a link $(u, v)$.

**Theorem 1**[1]: *Assume that nodes $s$ and $d$ in $F_r$ differ in $k$ bit-positions. The distance between $s$ and $d$, $D(s,d)$=$k$+2 if $s$ and $d$ are in the different clusters of the same class; otherwise $D(s,d)$=$k$.*

A path connecting two nodes $s$ and $d$ is called *minimal path* if its length is equal to $D(s,d)$. Based on Theorem 1, it is easy to derive that $D(s,d) \leq 2r+2$ for any $s$ and $d$ in $F_r$. $v$ is called a *preferred neighbor* if $u$ and $v$ are connected by a cube-edge and $D(v,d) < D(u,d)$; otherwise, any neighbor connected with $u$ by a cube-edge is called a *spare neighbor*. Preferred and spare neighbors are called *cube-neighbors* and the neighboring node connected with the cross-edge is called cross-neighbor. It is noted that the definition of preferred and spare neighbors depends on the cube-edges of the current class. If the current node changes to another class, the dimensions of its neighbors change and all its preferred and spare neighbors need re-calculated.

Routing in a fault-free dual-cube from a source $s(c_s, r_s[1], r_s[0])$ to its destination $d(c_d, r_d[1], r_d[0])$ is shown in [1] as follows. If $c_s = c_d$ and $r_s[c_s \oplus 1] = r_d[c_d \oplus 1]$, then, $s$ and $d$ are in the same cluster and it is the routing in this $r$-cube. The routing will select one of the preferred neighbors of the current node to advance to the destination. If $c_s \neq c_d$ ($s$ and $d$ are in different classes), say $c_s = 0$ and $c_d = 1$, we first route $s$ to $(0, r_s[1], r_d[0])$ where the routing exhausts all the preferred dimensions in class 0 and finds $r_u[c_u] \oplus r_d[c_u] = 0$. Such a node is also called *intermediate destination*. Then, route $(0, r_s[1], r_d[0])$ to $(1, r_s[1], r_d[0])$ through the cross-edge (also called a *jump*). After that jump, the routing enters a cluster in another class and the preferred and spared dimensions are all new. Thus, by exhausting all the preferred dimensions in class 1, $(1, r_s[1], r_d[0])$ can be routed to $d(1, r_d[1], r_d[0])$ where the routing will find $r_u[1] \oplus r_d[1] = 0$. Next, assume $c_s = c_d$ and $r_s[c_s \oplus 1] \neq r_d[c_d \oplus 1]$. $s$ and $d$ are in two different clusters of the same class, say class 0. We first route $s$ to its intermediate destination in class 0, $(0, r_s[1], r_d[0])$. Then, we route $(0, r_s[1], r_d[0])$ to $(1, r_s[1], r_d[0])$ through the cross-edge (a jump). Next, we route $(1, r_s[1], r_d[0])$ to its intermediate destination in class 1, $(1, r_d[1], r_d[0])$. Finally, route $(1, r_d[1], r_d[0])$ to $d(0, r_d[1], r_d[0])$ in one step through the cross-edge (another jump). The procedure of routing decision at each intermediate node $u$ is shown in Algorithm 1.

---

**Algorithm 1**: Routing decision at the current node $u(c_u, r_u[1], r_u[0])$ (destination: $d(c_d, r_d[1], r_d[0])$).

1. If $u$=$d$, then stop.
2. If $r_u[c_u] \oplus r_d[c_u] \neq 0$, select one of $u$'s preferred neighbors in the same cluster as the forwarding node.
3. Otherwise, it is at a node so called intermediate destination. Select $u'(c_u \oplus 1, r_u[1], r_u[0])$ as the forwarding node for a cross-edge hop (also called a jump).

---

For example, in an $F_2$ (see in Figure 2), the source $s$(00000) and the destination $d$(00011) have the same $class\_id$ and $cluster\_id$ and they are in the same cluster. The routing inside this cluster is applied: First, source $s$ has two preferred neighbors, nodes 00001 and 00010. One of them, say 00001 along dimension 0, is selected as the forwarding node and the routing message will be sent to it.

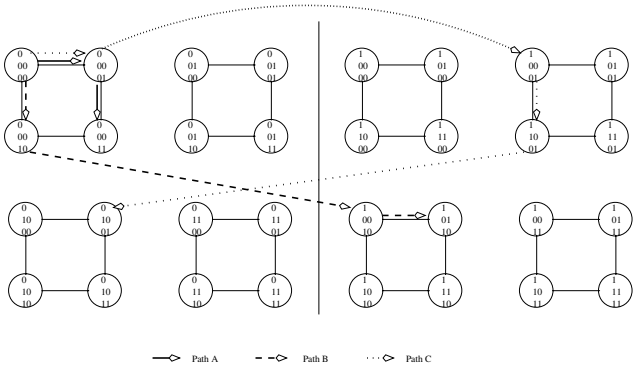


**Figure 2. Routing in a fault-free $F_2$.**

After the routing message arrives, this node (00001) has one preferred neighbor (00011) and one spare neighbor (00000). Node 00011 will be selected next. At last, the routing message will reach the destination $d$(00011). The routing path is shown in Figure 2 as *PATH A*. If the source $s$ and the destination $d$ are in different classes, for example, 00000 and 10110, the routing will first exhaust all the preferred dimensions in the cluster with the source $s$ and arrive at the intermediate destination 00010. And then, a jump from 00010 to 10010 occurs. After the routing message arrives at 10010 in class 1, the routing will exhaust all the preferred dimensions in the new cluster and reach the destination $d$. The routing path is shown in Figure 2 as *PATH B*. *PATH C* in Figure 2 shows the routing path for the case that the source $s$ and the destination $d$ (nodes 00000 and 01001) are in the same class but different clusters. After the routing reaches its intermediate destination in class 0, node 00001, the routing will select a jump to 10001. After that, the routing message will arrive at its intermediate destination in class 1, node 11001. At last, the routing message will reach the destination by one jump from 11001 to 01001.

## 3 Fault-tolerant Routing

**Depth-first search routing based on neighbors' condition.** A fault-tolerant routing scheme using depth-first search is applied to dual-cubes in Algorithm 2. Unlike the fault-tolerant routing in [1] that needs to identify all the faulty nodes, our routing here only requires every node to know the condition of its neighbors. Moreover, the routing in [1] can only handle $r-1$ faults for an $F_r$. Our routing here can handle more faults, even when the network is disconnected.

---

**Algorithm 2**: Depth-first search routing at the current node $u(c_u, r_u[1], r_u[0])$ (destination: $d(c_d, r_d[1], r_d[0])$).

1. If $u$=$d$, then stop.
2. Assume $SP(u)$ is set of $u$'s non-faulty neighbors which have not been tried before. If $SP(u) = \phi$, backtracking along the first incoming link is needed. If the source needs backtracking, the routing interrupts.
3. Select the forwarding node $u'$ from $SP(u)$ in the following priority order and forward the routing message to $u'$.
   (a) Select one of $u$'s preferred neighbor.
   (b) Select the cross-neighbor for a jump if $r_u[c_u] = r_d[c_u]$, the intermediate destination $((0, r_u[1], r_d[0])$ if $c_u = 0$; otherwise, $(1, r_d[1], r_u[0]))$ is a faulty neighbor or has been tried before, or there is no spare neighbor in $SP(u)$.
   (c) Select a spare neighbor.

---

The routing at the current node (including the source) will select a neighbor as the forwarding node and the routing message will be forwarded to it after the routing decision. It is noted that each neighbor tried before cannot be selected again unless a backtracking is needed. Thus, each routing message includes destination address and a list of used nodes along the path. At first, the routing will try any preferred neighbor to achieve the minimal path. If there is no preferred neighbor available (all are faulty neighbor or tried before), the routing will try any spare neighbor to find another way to the intermediate destination in the current cluster. There are several exceptions here to select the cross-neighbor as the forwarding node. First, if the routing is at the intermediate destination in the current cluster, the routing has exhausted all the preferred dimensions and will select the available cross-neighbor. By this jump the routing can go to another class and exhaust the remaining dimensions to reach the destination. Second, if the routing knows the intermediate destination in the current cluster is a faulty neighbor, there is no way to exhaust all the preferred dimensions in this cluster. The routing needs to jump to another class if the cross-neighbor is available. It will jump back to a different cluster in the same class. Since each class has the same dimensions of cube-edges, the routing will exhaust the remained dimension and reach the destination after it jumps back. Third, if the intermediate destination in the current cluster has been tried before, it means that the routing just jumped to this cluster for the second case. Selecting a spare neighbor here will increase the length of routing path. In our routing algorithm, the available cross-neighbor will be selected. Finally, if there is no spare neighbor available, the routing has no way to go and must select the only available

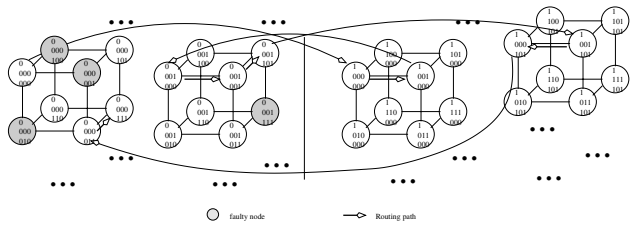


**Figure 3. Depth-first Routing in an $F_3$.**

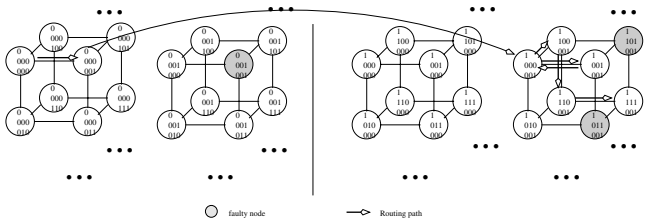


**Figure 4. Routing difficulties in depth-first search routing.**

cross-neighbor. If all neighbors are not available, the routing needs backtracking. And if the source needs backtracking, the dual-cubes may be disconnected and the routing process interrupts.

In Figure 3, an $F_3$ has faulty nodes 0000001, 0000010, 0000100, and 0001111. The source $s$ and the destination $d$ are nodes 0000000 and 0000111. The source $s$ has three faulty neighbors 0000001, 0000010, and 0000100. There is no spare or preferred neighbor in $SP$ set. The cross-neighbor 1000000 is selected as the forwarding node. As a node in class 1, node 1000000 has $r_u[1] = r_d[1]$. But the cross-neighbor 0000000 has just been tried and their connecting dimension is the incoming dimension of node 1000000. So spare neighbor 1001000 is selected next. After that, since 1000000 has been tried, at 1001000, the routing will select its cross-neighbor 0001000. Then, the routing message will reach 0001001 and 0001011 by selecting a preferred neighbor from the $SP$ set. The routing at 0001011 will know its intermediate destination in class 0, node 0001111, is a faulty neighbor. Thus, the cross-neighbor 1001011 from the $SP$ set is selected. At node 1001011, a preferred neighbor 1000011 will be selected. Since $r_u[1] = r_d[1]$ at node 1001011 (the intermediate destination in class 1), the cross-neighbor 0000011 will be selected next. After the routing message arrives at 0000011, it will reach the destination in the next step.

However, without fault information, the routing may enter a region where all the paths to the destination are blocked by faulty nodes. Thus, routing will try all the nodes in this region until it goes back to the entrance of this region. The routing needs detours and backtracking and caused routing difficulties which will increase routing delay and traffic congestion. Figure 4 shows an example of routing difficulty in an $F_3$. As the routing to the destination 1111001 selects 0000001 at the source 0000000, the routing message will go to 1000001 in the next step. After that, the routing will try node 1001001 to reach node 1111001. Since it is blocked by faulty nodes 1011001, 1101001, and 0001001, the routing needs backtracking to its entrance 1000001. After that, the routing can find a path to the destination through 1100001 and 1110001.

Next, we will introduce a routing which uses our limited-safety-level to avoid routing difficulties.

**Limited-safety-level model and limited-safety-level-based routing.** In a given $F_r$, the limited-safety-level of each node $u$, $LS(u) = k$, is its safety level [3] of the local cluster (an $r$-cube). Each cluster ($r$-cube) maintains its own safety level information in $r-1$ rounds of information exchanges among neighbors. Based on such information, a minimal path can be guaranteed for the routing in this local cluster. Adjacent clusters exchange their information to approximate their global fault information for the routing process, so that, a minimal path can be guaranteed before the routing enters a cluster by a jump.

**Definition 1**: *The limited-safety-level of a faulty node is* 0. *For a non-faulty node* $u$, *let* ($LS_0$, $LS_1$, $LS_2$, ..., $LS_{r-1}$) *be the non-descending limited-safety-level sequence of node* $u$*'s* $r$ *cube-neighbors, such that* $LS_i \leq LS_{i+1}$ ($0 \leq i \leq r-1$). *The limited-safety-level of node* $u$ *is defined as: if* ($LS_0$, $LS_1$, $LS_2$, ..., $LS_{r-1}$) $\geq (0, 1, 2, ..., r-1)$ ($seq_1 \geq seq_2$ *if and only if each element in* $seq_1$ *is greater or equal to the corresponding element in* $seq_2$.), *then* $LS(u) = r$ *else if* $(LS_0, LS_1, LS_2, ..., LS_{k-1}) \geq (0, 1, 2, ..., k-1) \wedge (LS_k = k-1)$ *then* $LS(u) = k$.

The limited-safety-levels can be calculated through iterative rounds of information exchanges among neighbors. Initially, all faulty nodes are assigned a limited-safety-level of 0 and all non-faulty nodes are assigned a limited-safety-level of $r$. Update of each limited-safety-level within each round is based on the limited-safety-level definition.

Based on the propositions of safety levels in hypercubes [3], the limited-safety-level of a node $u$, $LS(u)$, has following propositions:

1. The status of limited-safety-level is stabilized exactly in round $(r-1)$.

2. If the limited-safety-level of a node is $k$, then there is at least one minimal path from this node to any node within distance k in the same cluster.

It is clear that $(r-1)$ rounds of information exchanges are needed in the worst case to determine limited-safety-

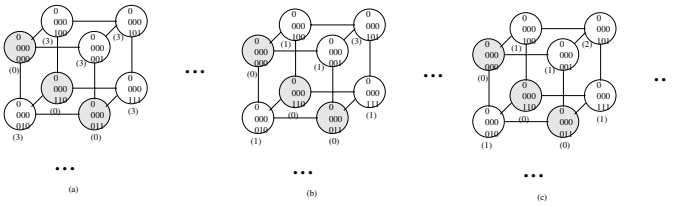

**Figure 5. 2 rounds of information exchanges in deciding limited-safety-level. (a) Initially assignment. (b) After the first round. (c) After the second round.**

levels of all nodes in a dual-cube. Figure 5 shows an example of limited-safety-level calculation in an $F_3$ through 2 rounds of information exchanges in a given faulty cluster (3-cube) with three faulty nodes: 0000000, 0000011, and 0000110. The number in each () represents the limited-safety-level of that node.

It is noted that each node $u$ has collected the limited-safety-level of its cross-neighbor $u'$ ($LS(u')$). $LS(u')$ does not affect the update of $LS(u)$. But it will be used in the routing process to avoid a jump to a cluster whose $r$-cube routing cannot complete due to the block by the faulty nodes. The routing based on the limited-safety-level information at an immediate node $u$ is shown in Algorithm 3.

---

**Algorithm 3**: Limited-safety-level-based routing at the current node $u(c_u, r_u[1], r_u[0])$ (destination: $d(c_d, r_d[1], r_d[0])$).

1. If $u$=$d$, then stop.
2. Forward the routing message to (a) a preferred neighbor $u'$ such that $LS(u') \geq\mid r_{u'}[c_u] \oplus r_d[c_u] \mid$, or (b) a spare neighbor $u'$ such that $LS(u') \geq\mid r_{u'}[c_u] \oplus r_d[c_u] \mid$ **unless** (a) $r_u[c_u] = r_d[c_u]$, or (b) node $u''$ ($(0, r_u[1], r_d[0])$ if $c_u = 0$; otherwise, $(1, r_d[1], r_u[0])$) is a faulty neighbor or has been tried before.
3. Forward the routing message to the cross-neighbor $u'$ such that $LS(u') \geq\mid r_u[c_u \oplus 1] \oplus r_d[c_u \oplus 1] \mid$.
4. Otherwise, interrupted.

---

At the source node $s$, first, the routing will try a preferred neighbor or spare neighbor (which will lead to 2 extra steps) whose limited-safety-level is no less than its distance to the intermediate destination in the same cluster. A minimal routing path or a sub-minimal path is guaranteed here. After that, at each node, the routing will try a preferred neighbor to reach that intermediate destination. However, the limited-safety-level still cannot effectively present

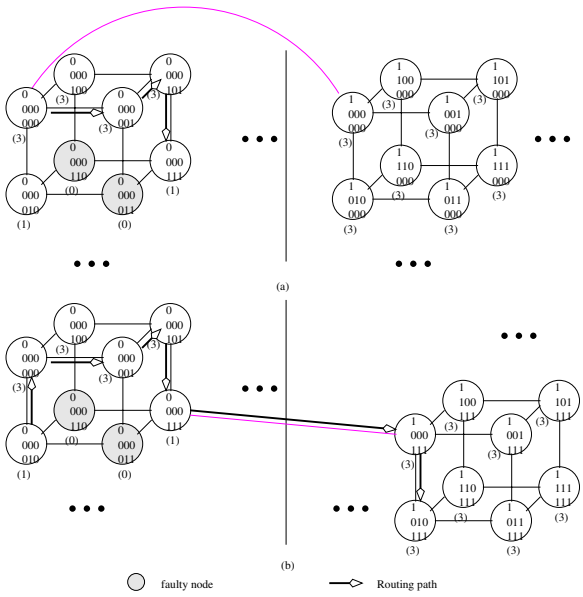


**Figure 6. Cube-routing by using local safety levels. (a) Routing without selecting spare neighbor in the process. (b) Routing with selecting spare neighbor in the process.**

fault information. The routing will not know if the intermediate destination is faulty or healthy until the routing reaches its neighbor. So if that intermediate destination is a faulty node, there is no minimal path and the routing must interrupt. When that healthy intermediate destination is reached, the cross-neighbor whose limited-safety-level is no less than its distance to the intermediate destination in the new cluster will be selected, so that, a minimal path can be guaranteed in the new cluster. If the limited-safety-level of the cross-neighbor is less than that distance, the routing may enter a dangerous area. To avoid routing difficulties, our routing will try only those nodes whose limited-safety-levels can ensure a minimal routing path. If there is no such an available forwarding node existing, the routing will interrupt. It will be tried later after all the faults recovered.

For example, assume the source $s$ and destination $d$ in an $F_3$ are 0000000 and 0000111. They are in the same cluster (see in Figure 6(a)). Among three preferred neighbors of the source $s$: 0000001, 0000010, and 0000100, the routing will select the preferred neighbor 0000001 because $LS(0000001) = 3 >\mid 001 \oplus 111 \mid = 2$. Neighbor 0000010 will be avoided because $LS(0000010) = 1 <\mid 010 \oplus 111 \mid = 2$. By the same reason, the routing will select node 0000101 at node 0000001 and finally reach the destination $d$. In Figure 6(b), the current node in the routing from the source 0000010 to the destination 1010111, as

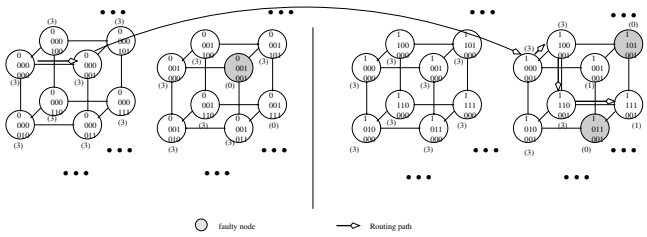


**Figure 7. Limited-safety-level-based routing for the example shown in Figure 4.**

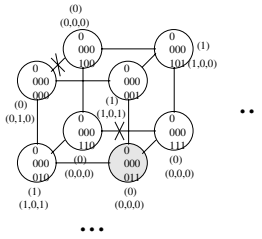

**Figure 8. An example of a faulty dual-cube in $F_3$ with its limited-safety-level and limited-safety-vector assignments.**

the source 0000010, may not have any available preferred neighbor (both preferred neighbors of 0000010: 0000011 and 0000110 are faulty nodes). By selecting a spare neighbor 0000000 ($LS(0000000) = 3 \geq |\ 000 \oplus 111\ |$), the routing will find a path to 0000111. And then, a jump to 1000111 ($LS(1000111) = 3 \geq |\ 000 \oplus 010\ | = 1$). Finally, the routing will reach the destination.

For the example in Figure 4 we saw before, by using the limited-safety-level information (see in Figure 7), after the routing arrives at 1000001, the routing will select a preferred node 1100001 since $LS(1100001) = 3 > |\ 100 \oplus 111\ | = 2$. And then, the routing will select 1110001 in the next step since $LS(1110001) = 3 > |\ 110 \oplus 111\ | = 1$. Compared with the routing path in Figure 4 and that in Figure 7, our routing based on the limited-safety-level information avoided routing difficulties and needs fewer steps.

In the above discussion, it is assumed that all faults are node faults. To extend it to cover link faults, both end nodes of a faulty link must be considered as faulty nodes. Next, we use limited-safety-vector to represent fault information. Each end node of a faulty link treats the other one as faulty, but it does not consider itself faulty.

**Limited-safety-vector model and Limited-safety-vector-based routing.** The limited-safety-vector model is an improvement based on the limited-safety-level model. Basically, each node $u$ in a dual-cube is associated with an $r$-bit vector $VS(u) = (u_1, u_2, ..., u_r)$ called limited-safety-vector, calculated through $r-1$ rounds of information exchanges among cube-neighbors. In a given $F_r$, the limited-safety-vector of each node $u$, $VS(u)$, is its safety vector [2] of the local cluster (an $r$-cube). Based on such information, a minimal path can be guaranteed for the routing in this local cluster. Adjacent clusters exchange their information to approximate their global fault information for the routing process, so that, a minimal path can be guaranteed before the routing enters a cluster by a jump. Compared with limited-safety-levels, Limited-safety-vectors can provide more accurate information about the number and the distribution of faults in the dual-cubes.

**Definition 2**:

1. *The limited-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 $u$.*

2. *Base for the first bit:*

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

3. *Inductive definition for the $k^{th}$ bit:*

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

Based on the propositions of safety vectors in hypercubes [2], the limited-safety-vector of a node $u$, $VS(u)$, has the following proposition: *If $u_k$, the $k^{th}$ bit in $VS(u)$, is 1, there exists at least one minimal path from node $u$ to any node inside the cluster which is exactly distance-k away.*

Figure 8 shows an example of a cluster inside an $F_3$ with one faulty node 0000011 and two faulty links (0000000,0000100) and (0000110,0000111). In this example, the limited-safety-level of each node in this cluster (3-cube) is either 0, 1, i.e., the current node inside this cluster can only ensure a message to its neighbors. Clearly, by inspection, the limited-safety-level information is not accurate. For example, node 0000000, 0000110, and 0000010 can send a message to any nodes inside this cube through a minimal path that are distance-2 or -3 away. This problem is partially resolved in our limited-safety-vector model, where the limited-safety-vector associated with nodes 0000000 and 0000010 are (0,1,0) and (1,0,1), respectively. The reason that node 0000010 has a 1-bit at the $3^{rd}$ bit of its limited-safety-vector is that it has one cube-neighbor 0000000 with 1-bit as the $2^{nd}$ bit of its limited-safety-vector. Therefore, those faulty nodes and faulty links will

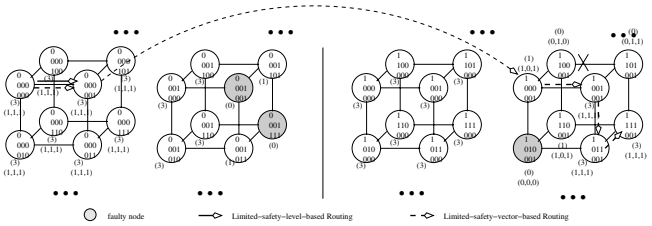


**Figure 9. Examples of limited-safety-level-based routing and limited-safety-vector-based routing in an $F_3$.**

not block all the minimal paths start from node 0000010 to a node inside this cluster that is distance 3 away.

---

**Algorithm 4**: Limited-safety-vector-based routing at the current node $u(c_u, r_u[1], r_u[0])$ (destination: $d(c_d, r_d[1], r_d[0])$).

1. If $u$=$d$, then stop.
2. Forward the routing message to (a) a preferred neighbor $u'$ such that $u'_{|r_{u'}[c_u] \oplus r_d[c_u]|} = 1$, or (b) a spare neighbor $u'$ such that $u'_{|r_{u'}[c_u] \oplus r_d[c_u]|} = 1$ **unless** (a) $r_u[c_u] = r_d[c_u]$, or (b) node $u''$ ($(0, r_u[1], r_d[0])$ if $c_u = 0$; otherwise, $(1, r_d[1], r_u[0])$) is a faulty neighbor or has been tried before.
3. Forward the routing message to the cross-neighbor $u'$ such that $u'_{|r_u[c_u \oplus 1] \oplus r_d[c_u \oplus 1]|} = 1$.
4. Otherwise, interrupted.

---

The procedure of routing decision based on the limited-safety-vector information at an immediate node $u$ is shown in Algorithm 4. Algorithm 4 is similar to Algorithm 3, except for using the vector information. The limited-safety-vector-based routing uses that vector information which is more accurate than the limited-safety-level information. Figure 9 shows the difference between the limited-safety-level-based routing and the limited-safety-vector-based routing in an $F_3$. The source and the destination are 0000000 and 1111001. Based on the limited-safety-level information of node 1000001 at node $u(0000001)$ ($LS(1000001) = 1 <| 000 \oplus 111 |= 3$), the limited-safety-level-based routing at $u$ will interrupt and can not find the minimal routing path. Based on the limited-safety-vector information of node 1000001 at node $u(0000001)$ ($u_{|000 \oplus 111|} = 1$), the limited-safety-vector-based routing will select 1000001 for a jump at node $u(0000001)$. After that, it will reach the destination through a minimal path.

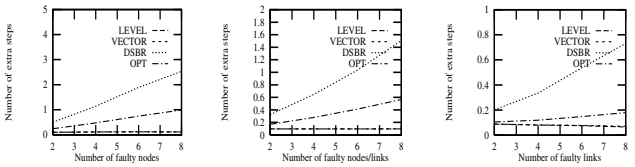


**Figure 10. $F_2$ routings.**

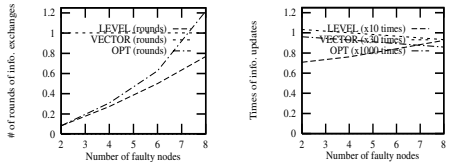


**Figure 11. Comparison of information collection for different routings in an $F_2$.**

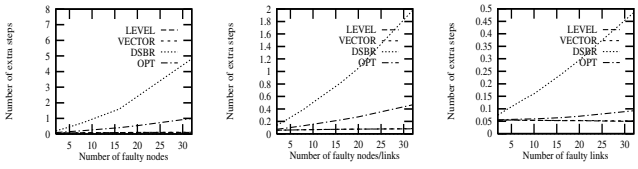


**Figure 12. $F_3$ routings.**

In the following section, we show the performance of our routing algorithms by experimental results.

## 4 Performance

A simulation has been conducted on different size dual-cubes ($F_2$, $F_3$, and $F_4$) to test the average extra steps in different routings: depth-first search routing (DSBR), limited-safety-level-based routing (LEVEL), limited-safety-vector-based routing (VECTOR), and routing using global fault information (OPT).

We randomly generate faulty nodes, faulty links, source and destination. Any $F_r$ ($r$ =2, 3, and 4) has up to $2^{2*r-1}$ faulty components. Figure 10 shows the performance of routing in an $F_2$ for different cases of faulty components: (1) all faulty nodes, (2) half faulty nodes and half faulty links, and (3) all faulty links. Figure 12 and Figure 14 show those in $F_3$ and $F_4$. Figures 11, 13, and 15 show the cost of information collection for different routings (LEVEL, VECTOR, and OPT) in $F_2$, $F_3$, and $F_4$.

We make the following observations from the comparison of these figures.

- The LEVEL routing and VECTOR routing can ensure

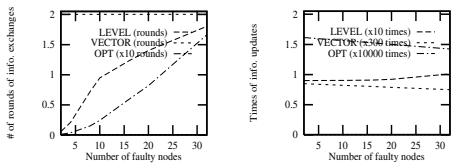


**Figure 13. Comparison of information collection for different routings in an $F_3$.**

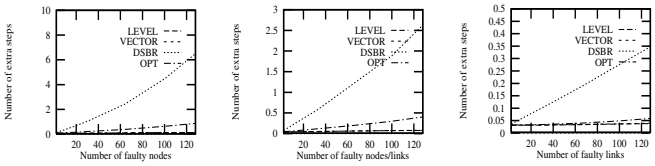


**Figure 14. $F_4$ routings.**

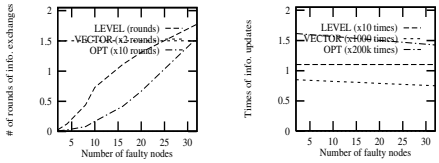


**Figure 15. Comparison of information collection for different routings in an $F_4$.**

a minimal routing path in each cluster when they enter that cluster. Thus, they do not need backtracking.

- If there is no minimal path in one cluster, the OPT routing will try the shortest path which may have a few extra steps. The DSBR routing needs more detours to find a way to the destination and has more extra steps. In such a case, the LEVEL routing and the VECTOR routing which use our limited global fault information will realize that the current network configuration is not good for data transmission in which we emphasize high quality and performance. To get a shorter path, they will retry after the faults are recovered. Therefore, the LEVEL routing and the VECTOR routing will only try those cases with minimal paths in OPT routing and leave a shorter path for data transmission.

- In some cases, the VECTOR routing can find a routing path but the LEVEL routing cannot (see in Figure 9). This is due to the fact that the limited-safety-vector information used in the VECTOR routing is more precise than the limited-safety-level information used in the LEVEL routing. However, the collection of limited-safety-vector information needs more rounds ($\times 2$ in $F_4$) of neighbor information exchanges and more information update times ($\times 100$ in $F_4$) than those in the collection of limited-safety-level information. The experimental results of the average routing path in the VECTOR routing and those in the LEVEL routing are close.

- The OPT routing needs global information broadcasting. Unlike the global information model, our information model is more cost effective. The information update times in our information propagation can be reduced to $\frac{1}{200}$ of that in global information broadcasting in $F_4$. Also, our information model needs only few rounds of neighbor information exchanges ($\frac{1}{5}$ of that in global information broadcasting in $F_4$).

## 5 Conclusions

In summary, we provided a fault-tolerant routing with its extensions based on our limited global information in the dual-cube networks. They can handle more than $r$ faulty nodes (up to $2^{2*r-1}$ in an $F_r$). Also, faulty links are considered here. Here, we use limited-safety-level and limited-safety-vector to represent our limited global information in dual-cubes. The experimental results show the performance of these routing algorithms and the effectiveness of our limited global information. Unlike the routing in [1], we do not need to collect global fault information whose broadcasting propagation will incur traffic congestion and even new component failure in the networks. The information collection for our routings needs few rounds of information exchanges among neighbors. This increases the self-healing ability of such a network. Next, we will extend our results to dynamic dual-cubes in which all the faulty components can occur during the routing process. Also, our results will be extended to other cluster networks.

## References

[1] Y. Li and S. Peng. Fault-tolerant routing and disjoint paths in dual-cube: A new interconnection network. *Proc. of the Eighth International Conf. on Parallel and Distributed Systmes*. 2001, 315-322.

[2] J. Wu. Adaptive fault-tolerant routing in cube-based multicomputers using safety vectors. *IEEE Transactions on Parallel and Distributed Systems*. 9, (4), April, 1998, 321-334.

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