Dynamic Fault-Tolerant Routing in Cube-Based Multicomputers [*]

Jie Wu and Zhen Jiang
Department of Computer Science and Engineering
Florida Atlantic University
Boca Raton, FL 33431
email:{jie, zjaing}@cse.fau.edu

ABSTRACT
The *safety level* model is a special coded fault information model designed to support fault-tolerant routing in hypercubes. In this model, each node is associated with an integer, called safety level, which is an approximated measure of the number and distribution of faulty nodes in the neighborhood. The safety level of each node in an n-dimensional hypercube (n-cube) can be easily calculated through an $(n-1)$-round of information exchanges among neighboring nodes. This paper focuses on an analytical study on routing capability using safety levels in a dynamic system; that is, a system in which new faults might occur during a routing process. Under the assumption that the total number of faults is less than $n-1$, we provide an upper bound on the probability of detours (each detour causes two extra steps) in a routing process. Simulation results are also provided to compare with the proposed upper bound.

KEY WORDS
Dynamic faults, fault tolerance, hypercubes, routing, safety levels

1. Introduction

A central issue in designing a fault-tolerant routing algorithm in hypercubes is the way fault information is collected and used ([1, 2, 3, 4]). *Limited-global-information-based* routing is a compromise between local-information-based and global-information-based approaches. In this approach, global fault information is collected and packed based on a special coding scheme. In the *safety level* model [5], an integer is associated with each node in an n-cube representing limited-global-information in the system. Safety level is an approximated measure of the number and distribution of faulty nodes in the neighborhood based on a special coding method. 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.

Basically, each node in an n-cube is assigned a safety level k, where $k \in \{0, 1, ..., n\}$, and this node is called k-safe. A k-safe node indicates the existence of at least one Hamming distance path (also called an optimal path) from this node to any node within Hamming distance k. 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 hypercube. An optimal unicasting 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 an optimal or suboptimal unicasting can by easily determined at the source node by comparing its safety level, together with its neighbors' safety levels, with the Hamming distance between the source and destination nodes.

However, a static fault model is used in the safety level model [5]: it is assumed that no new faults will occur during a routing process. This paper presents our first attempt to study the effect of dynamic faults (faults that occur during a routing process). When a new fault occurs, safety level information is no longer accurate for certain nodes. Therefore, the updates of safety levels and the routing process may have to proceed hand-in-hand. We focus here the analytical study on the effect of dynamic faults on routing capability. Under the assumption that the total number of faults is less than $n-1$ in an n-cube, an upper bound is given on the probability of k detours (each detour causes two extra steps) in a routing process. Simulation results are also provided to compare with the proposed upper bound.

2. Preliminaries

N-cube. The n-cube Q_n is a graph having 2^n nodes labeled from 0 to 2^n-1. Two nodes are connected by an edge if their addresses, as binary numbers, differ in exactly one bit position. Every node u has an address $u_{n-1}u_{n-2}\cdots u_0$ with $u_i \in \{0,1\}$, where $i \in \{0, 1, ..., n\}$, and u_i is called the ith bit (also called the ith dimension) of the address. We denote node u^i the neighbor of node u along dimension i. Symbol $\oplus$ denotes the bitwise exclusive OR operation on binary addresses of two nodes. 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 called *Hamming distance path* (also *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 *preferred neighbors*. The remaining $n - H(s, d)$ dimensions are called *spare dimensions* and the correspond-

[*]This work was supported in part by NSF grant CCR 9900646.

ing 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. A *detour* corresponds to a selection of a spare neighbor as a forwarding node in a routing process. Based on the hypercube topology, each detour adds two extra steps. For example, a path $0101 \rightarrow 0100 \rightarrow 1100 \rightarrow 1000 \rightarrow 1010 \rightarrow 1011$ has one detour from 0101 to 0100 in dimension 0. A path with one detour is called *suboptimal*. A *backtrack* at node v corresponds to a path segment of form $u \rightarrow v \rightarrow u$, that is, a routing message at node v is sent back to where it comes from (node u). This type of backtrack is also called *direct backtrack*. An *indirect backtrack* corresponds to a path segment of form $u \rightarrow v \rightarrow ... \rightarrow w \rightarrow u$, where $u \neq v \neq w$. A *progressive routing* is a routing without any direct backtrack (it may contain one or more indirect backtracks). A progressive routing will eventually deliver a routing message if the number of detours is limited.

Safety Levels. In a given n-cube, the safety level of each node ranges from 0 to n. Let $S(u) = k$ be the safety status of node u, where k is referred to as the level of safety, and u 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 $k \neq n$.

Definition 1 [5]: *The safety level of a faulty node is* 0. *For a nonfaulty node u, let $(S_0, S_1, S_2, ..., S_{n-1})$, $0 \leq S_i \leq n$, be the non-descending safety level sequence of node u'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 u is defined as: if $(S_0, S_1, S_2, ..., S_{n-1}) \geq (0, 1, 2, ..., n-1)$ ($seq_1 \geq seq_2$ if and only if each element in seq_1 is greater or equal to the corresponding element in seq_2.), then $S(u) = n$ else if $(S_0, S_1, S_2, ..., S_{k-1}) \geq (0, 1, 2, ..., k-1) \wedge (S_k = k-1)$ then $S(u) = k$.*

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

Proposition 1 [5]: *The status of a k-safe ($k \neq n$) node in an n-cube is stabilized exactly in round k.*

From Proposition 1, it is clear that $n-1$ rounds of information exchanges are needed in the worst case to determine safety levels of all nodes in an n-cube. Figure 1 shows an example of safety level calculation through 2 rounds of information exchanges in a given faulty 4-cube with four faulty nodes: 0100, 0011, 0110, and 1001. The number in each node (a circle) represents the safety level of that node.

Proposition 2 [5]: *If the safety level of a node is k, then there is at least one Hamming distance path from this node to any node within Hamming distance k.*

Proposition 2 serves as a base for constructing an optimal path. If a source node's safety level is at least as high as its distance to the destination, a preferred neighbor with the *highest safety level* is selected as the next forward node. Following this procedure at each intermediate node, an optimal path is eventually constructed.

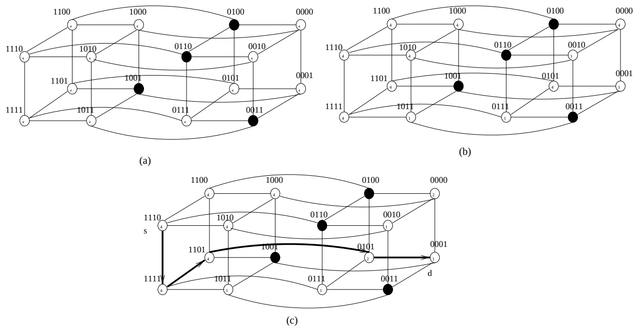

Figure 1. 2 rounds of information exchanges in deciding safety levels. (a) Initially assignment. (b) After the 1st round. (c) After the 2nd round.

Proposition 3 [5]: *In a faulty n-cube with fewer than $n-1$ faulty nodes, each nonfaulty but unsafe node has a safe neighbor.*

Note that a safe neighbor may be a spare node. Therefore, in a static system, suboptimal routing (with at most one detour) is guaranteed based on Proposition 3 as long as the number of faults is no more than $n-1$.

Safety-level-based routing. The safety-level-based routing is an optimal routing in hypercubes, that is, it always tries to find a shortest path if possible. Like a regular optimal routing, at each step it tries to forward the routing message to a preferred neighbor. The difference is that in the safety-level-based routing the selected preferred neighbor ensures the remaining path is still optimal (provided no new fault will occur during the routing process). The selection of a neighbor of the highest safety level is to better handle dynamic faults (to be discussed in the next section).

In a unicasting in Figure 1(c) where $s = 1110$ and $d = 0001$ are the source and destination nodes, respectively, the *navigation vector* is $N = s \oplus d = 1111$ which is used to guide the routing message; hence, $H(s,d) = 4$. Also, the safety level of the source s is 4. Therefore, the optimal algorithm is applied. Among preferred neighbors of the source, 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 0, is selected. Following the above steps, an optimal path from 1110 to 0001 is constructed as shown in Figure 1(c).

3. Dynamic Fault Model

Any faults occurred during a routing process are considered dynamic faults. In other words, during a routing process, safety level information may not be stable (also called inconsistent). In this section, we estimate the maximum number of additional detours for the safety-level-based routing

Safety-level-based routing (in static systems)

1. At the source node, if the safety level of a preferred neighbor is no less than the distance between the neighbor and destination, then perform an optimal routing by forwarding the routing message and destination node address to the preferred neighbor with the highest safety level; if the safety level of the spare neighbor is no less than the distance between the spare neighbor and destination, then perform a suboptimal routing by forwarding the routing message and destination node address to the spare node; otherwise, safety-level-based routing fails.
2. At an intermediate node, when a routing message together with the destination node address is received at an intermediate node, if the intermediate node is the destination, the routing message is saved and the routing process stops; otherwise, the routing message together with the destination node address is forwarded to a preferred neighbor with the highest safety level.

Extended safety-level-based routing (in dynamic systems)

1. Same as step 1 of safety-level-based routing.
2. At an intermediate node, when a routing message together with the destination node address is received, if the intermediate node is the detination, the routing message is saved and routing process stops; otherwise, if the highest safety level preferred neighbor is no less than the Hamming distance between the current and destination nodes, that node is selected as a forwarding node; otherwise, a spare node with the highest safety level is selected.

given the speed of the routing process and the update of safety levels in a dynamic system.

First of all, the safety-level-based routing needs to be extended to a dynamic system. Specifically, at the step 2, the implicit condition that the safety level of the selected neighbor is no less than the distance between the highest safety level neighbor and destination may no longer hold, because the new fault may reduce the safety levels of nodes in the neighborhood; however, Proposition 3 ensures that a safe neighbor (preferred or spare) exists and the selection of such a safe neighbor will be correct if the safety level information is *stable* (also called *consistent*).

Note that the extended safety-level-based routing does not distinguish consistent safety levels from inconsistent ones. When a neighbor with an inconsistent safety level is selected, that node may not satisfy the safety level requirement after it becomes stable. In this case, additional detours may occur.

Assume that there are at most F ($< n-1$) faulty nodes in an n-cube, including dynamically generated faults. Faults f_1, f_2, ..., f_F occur at time $t_1, t_2, ..., t_F$, respectively, where $t_{i+1} - t_i = d_i$ ($1 \leq i < F$). It is assumed that safety levels in Q_n are stabilized before the occurrence of a new fault. Before a routing message is initiated at time t, it is assumed that the first m faults have already occurred in Q_n; that is, $m = \max\{l | t_l \leq t\}$. $D(i)$ represents the distance from the current node (u) to the destination (d) at time t_i when fault f_i occurs ($1 \leq i \leq F$). Before the start time t, the routing message is at its source s and $D(i) = H = H(s,d)$ ($i \leq m$). $F(i)$ represents the distance from the current node (u) to f_i at time t_i when f_i occurs. Table 1 summaries the notation used in this paper.

$S(u)$	safety level of current node u
u^i	the neighbor of current node u along dimension i
$Q_l(u, d)$	spanning l-subcube of current node u and destination d, where $H(u, d) = l$
F	number of faults in a given n-cube and $F < n - 1$
f_i	i^{th} fault
t_i	occurrence time of f_i
d_i	interval between two consecutive faults f_i and f_{i+1}, i.e., $d_i = t_{i+1} - t_i$
t	start time of a routing process
m	number of faults before the routing process starts
$H(u, d)$	Hamming distance between u and d; specially, H=H(s,d)
p	maximum number of new faults that can potentially affect a given routing process
$D(i)$	distance from the current node u to the destination d at time t_i
$F(i)$	distance from the current node u to fault f_i at time t_i
λ	number of rounds of information exchanges and updates at each step

Table 1. List of notation used.

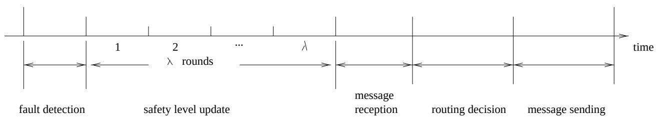

Figure 2. Activities of different phases in a step.

We adopt the following model for safety level update in the routing process. The update consists of a sequence of steps. Each step consists of the following phases. Every node in Q_n starts with fault detection of adjacent nodes, and then collects and distributes fault information through λ rounds of exchanges and updates. Before the end of each step, based on the safety level information, a routing decision is made to select a forwarding node and the routing message is sent to the selected node. Therefore, every routing message advances one hop at each step. The actions within a step are shown in Figure 2. It is assumed that any adjacent faults are detected at "fault detection" phase. Any faults occur after the fault detection phase will be detected at the next step. It is also assumed that the action "message reception" occurs right before the "routing decision" as shown in Figure 2. The model used represents a *reactive approach* where safety level update is done only when there is a change of safety status of at least one neighbor.

Theorem 1: *The safety level of each node is updated at most once for each new fault f_i. Specifically, if node u is Hamming distance k from f_i, its update (if any) occurs at the k^{th} round. In addition, if some neighbors of node u update their levels at the $(k-1)^{th}$ round and trigger the update of $S(u)$ at the k^{th} round, $S(u)$ is higher than all these levels.*

Corollary 1: *Any node in an n-cube has at least two safe neighbors if $F < n - 1$.*

Figure 3 shows two possible routing cases. Node u is

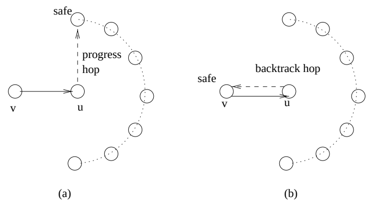

Figure 3. Two routing cases.

the current node and node v is the source of the previous hop. The remaining nodes in the figure are neighbors of u. Corollary 1 ensures that a progressive routing (a routing without backtracking) is always possible if the number of faults is no more than $n-2$ because other than the source of previous hop there is at least one more safe neighbor (preferred or spare). Therefore, backtrack hop never occurs. However, a dynamic fault may cause more than one detour if safety level information is inconsistent by the time a routing decision is made.

Theorem 2: *After the occurrence of f_i, safety levels of all the nodes in Q_n are stabilized in $\lceil \frac{i-1}{\lambda} \rceil$ rounds, provided f_{i+1} does not occur during the period.*

Obviously, the number of rounds for safety level update is bounded by $\lceil \frac{F-1}{\lambda} \rceil$ and $d_i > \lceil \frac{i-1}{\lambda} \rceil$. Our model can be extended for the case where the above condition is violated.

While the safety level information is updated by the occurrence of a new fault f_i, the update signal may not reach the current node u and its neighbors before a routing decision is made at u. If the highest safety level of the preferred nodes of a node is less than the distance to d at the current step, its routing decision will select a spare neighbor and cause one detour. Suppose that the safety levels of spare neighbors of node u are unstable (inconsistent), the routing decision may select a spare neighbor whose safety level is sufficient high at the current step but will be stabilized to a level less than its Hamming distance to at a later step; an extra detour may occur.

Consider the example where a faulty 6-cube has four faults: $f_1(001001)$, $f_2(000101)$, $f_3(000011)$, and $f_4(010001)$ occurred in sequence t_1, t_2, t_3, and t_4, respectively. It is assumed that $d_i \geq 4$ for $i = 1, 2, 3$, and $\lambda = 1$. The number in each circle (node) of Figure 4 represents the safety level of this node. A unicasting where $s = 010100$ and $d = 000001$ starts at $t = t_4 - 1$ (see Figure 4(a) where only the relevant 5-subcube is shown), respectively, the navigation vector is $s \oplus d = 010101$, and hence, $H = 3$. $t = t_4 - 1$ indicates that there are only three faults before the routing message is initiated at time t. Before the appearance of the fourth fault (f_4 at t_4), the neighbors of the source 010101, 010110, 010000, 011100, 000100, and 110100 have safety levels of 6 and their safety levels are no less than H. A neighbor with the highest safety level, say 010101 along dimension 0, is selected. Node 010001

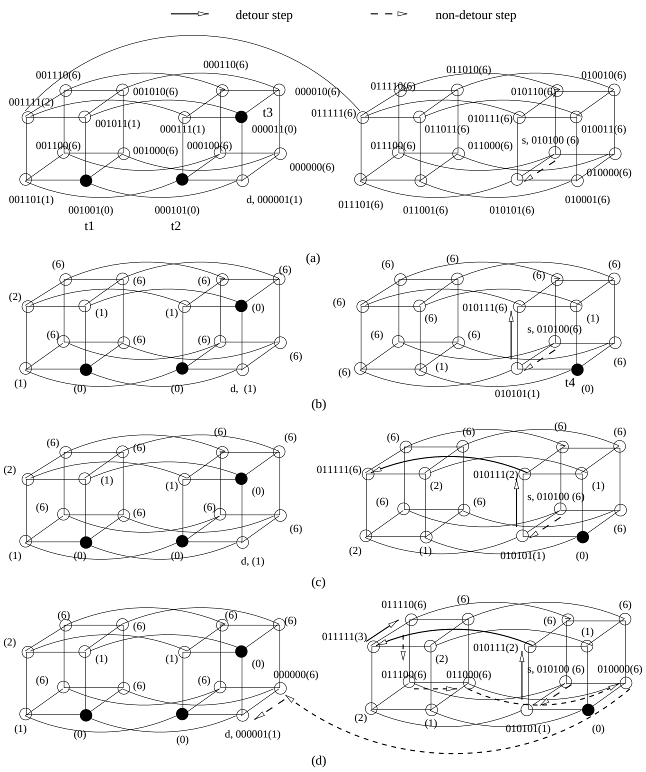

Figure 4. Extra detour caused by a new fault with inconsistent safety level information.

(f_4) is faulty at t_4 and the safety level of 010101 is reduced to 1 before that node makes routing decision (see Figure 4(b)). After collecting its neighbors' status, node 010101 decides to make the routing detour to a spare neighbor; say 010111 along dimension 1. After the routing message arrives at the node 010111 (see Figure 4(c)), the safety levels of all preferred neighbors are all too low. Therefore, a safe spare neighbor; say 011111, is selected. However, when the routing message arrives at 011111, its safety level has been reduced down to 3 (see Figure 4(d)) and the safety levels of all preferred neighbors are all too low. Therefore, a third detour occurs by forwarding the routing message to a safe spare neighbor 011110. The remaining routing path (initiated from node 011110) is shown in Figure 4(d). Note that nodes 010111, and 011111 should not have been selected as forwarding nodes at nodes 010101 and 010111, respectively, to avoid such extra detours.

Extra detours are due to inconsistent safety level information of neighbors which may cause a sequence of detours. The worst case is that when the routing process and the update of safety levels proceed at the same speed ($\lambda = 1$) the routing message may go along with the information propagation caused by f_i. In the example shown in Figure 4(d), the propagation caused by the fourth fault is stabilized in 3 steps and the routing message has 3 detours.

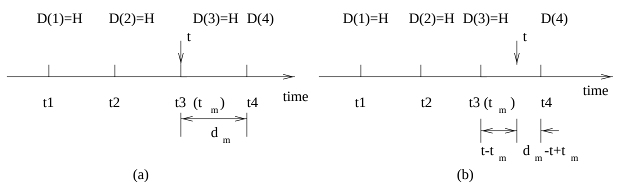

Figure 5. Sequence diagrams of routing time t (a) coincides with t_m ($t = t_m$) and (b) does not coincide with t_m ($t > t_m$) where $m = 3$.

4. Detour Analysis

We perform detour analysis under both consistent and inconsistent safety level information. We assume that $\lambda \geq 2$ in the subsequent discussion. Note that the assumption $\lambda \geq 2$ does not avoid inconsistent safety level information. However, it prevents the sequence of detours caused by a single new fault as shown in Figure 4.

Detour analysis with consistent safety level information. Since there is at most F $(< n-1)$ faults in the n-cube, when any new fault occurs, the safety level information of the entire n-cube is stabilized in one ($\lceil \frac{F-1}{\lambda} \rceil = 1$) step if $\lambda \geq F-1$. Thus, before the routing message selects a forwarding node at current node u, the safety level is stabilized and there is no inconsistent safety level information. The routing process at current node is the same as that at the source in the static model.

Consider a routing starts at time t in interval d_m (see Figure 5), if $t = t_m$, the routing message advances d_m steps from t_m to t_{m+1}. If $t > t_m$, the routing message advances $d_m - (t - t_m)$ steps during interval d_m.

Theorem 3: If $\lambda \geq n-3$ and $D(i+1) > 0$,

$$\begin{cases} D(i) & = & H & i \leq m \\ D(m+1) & = & D(m) - (d_m - t + t_m) & t > t_m \\ D(m+1) & = & D(m) - (d_m - 2) & t = t_m \\ D(i+1) & = & D(i) - (d_i - 2) & i > m \end{cases}$$

Assume $m+p-1$ is the largest index for faults such that $D(m+p-1) > 0$ and f_{m+p-1} is the last fault that could affect the routing process. Actually, p is the maximum number of intervals in which the routing message detours at least once and it is also the maximum number of new faults that could affect the current routing process. Based on Theorem 3, we have $p = \max\{l | H - \sum_{i=m}^{m+l-2}(d_i - 2) > 0\}$. Let us consider several cases:

1. The routing message will get closer to the destination in an interval (period between two of consecutive faults) as long as the interval is longer than 2 time units.
2. When d_i's are uniform, i.e., $d_i = c, p = \max\{l | (l-1)(c-2) < H\}$. Then $p = \lceil \frac{H}{c-2} \rceil$. The maximum number of detours for a message is no more than $\lceil \frac{H}{c-2} \rceil$, the length of the resultant path is no more than $H + 2 \times \lceil \frac{H}{c-2} \rceil$.

When d_i's are uniform and assume that $n = 8$, $c = 6$, and $p = \lceil \frac{H}{4} \rceil$. Since $H \leq 8$ ($H \leq n$), $p = 2$. Thus, the maximum number of detours for any routing message is no more than 2 and the length of resultant path is no more than $H + 4$.

Theorem 4: *After the routing message starts at t ($t \geq t_m$), there are at most p ($p = \lceil \frac{H}{c-2} \rceil$) faults that can affect a routing process in an n-cube, the probability of p detours, i.e, the length of the resultant path is $H + 2p$, is bounded by $\prod_{i=m}^{m+p} 2^{D(i)-n}$.*

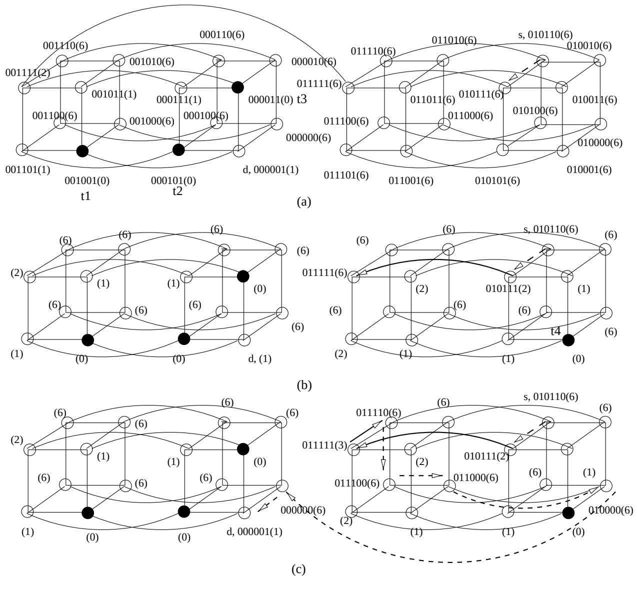

Figure 6. Extra detour caused by incomplete fault information propagation.

Detour analysis with inconsistent fault information. If $2 \leq \lambda < F-1$ is less than $F-1$, safety levels may not be stabilized in one step. When the information propagation of a new fault f_i reduces the safety level of the current node u and forces a routing detour to one of its spare neighbors (v), node v may not receive an update signal, rather it will receive the signal in the next step. The routing decision at the current node u is based on the out dated, $S(v)$. Therefore, after the routing message is received at v, an extra detour is needed. The inconsistent safety level information of v at node u is due to the incomplete information propagation. Since $\lambda \geq 2$, the safety level information of v's neighbors are stabilized at the same step of v's update. Thus, there is no more detour after the extra detour from v.

Consider the example in Figure 6, when $\lambda = 2$, assume that the source $s = 010110$ and $d = 000001$, respectively, the navigation vector is $s \oplus d = 010111$. After the routing process starts from s, the routing message is sent to 010111 which is 2 hops from d. When f_4 appears before the next routing decision, the safety level at node 010111 is changed to 2. Its neighbor node 011111 is still safe and can be selected. But after the routing decision, the safety level propagation information arrives at 011111. The safety level of 011111 is changed to 3. The routing message needs an extra detour to node 011110. Therefore, a new fault causes two detours. The information propagation of f_4 cannot reach node 011111 at the same step as t_4 since the distance from 011111 to 010001 (3) is larger than the rate of propagation ($\lambda = 2$). The incomplete propagation causes the inconsistent safety level information of node 011111 at the "routing decision" of node 010111. This causes an extra detour at node 011111. But after the extra detour, since each neighbor of node 011111 has consistent safety level information, there is no more detour.

Theorem 5: If $2 \leq \lambda < n - 3$ and $D(i+1) > 0$,

$$\begin{cases} D(i) & = & H & i \leq m \\ D(m+1) & \leq & n - (d_m - t + t_m - 4) & \\ D(i+1) & \leq & D(i) - (d_i - 4) & i > m \end{cases}$$

As discussed above, $m + p - 1$ is the largest index for faults such that $D(m + p - 1) > 0$ and f_{m+p-1} is the last fault that could affect the routing process. Based on Theorem 5, we have

Corollary 2: *If a routing message generates detours in at most p intervals (f_{k_1}, f_{k_2}, ..., f_{k_p}), then p is limited as :* $p \leq \max\{l | l - 1 + \frac{\lambda}{\lambda+1} * (\sum_{i=1}^{l-1}(d_{k_i} - 4) + 3) \leq n - 2\}$.
For any λ ($2 \leq \lambda < n - 3$), we have:

1. The routing message will get closer to the destination between two occurrences of consecutive faults as long as these two faults are separated by more than 4 time units.
2. When d_i's are uniform, i.e., $d_i = c$, $p \leq \lfloor \frac{\lambda * n + \lambda * c + n - 8 * \lambda - 1}{c * \lambda - 3 * \lambda + 1} \rfloor \leq \lfloor \frac{3 * n + 2 * c - 17}{2 * c - 5} \rfloor$.

When d_i's are uniform, $n = 8$, $\lambda = 4$ and $c = 6$, p is no more than 2. Thus, the maximum number of detours for any message, $2 * p$, is no more than 4 and the length of resultant path, $8 + 4 * p$, is no more than 16.

5. Simulation

A simulation has been conducted on n-cubes of different size ($4 \leq n \leq 10$) to test the estimated maximum number of detours in a safety-level-based routing. Average number of detours in a safety-level-based routing is also calculated.

We randomly generate faults and destinations. To simplify the simulation, we assume that all the destination nodes are fault free and d_i's are uniform. Table 2 illustrates the analytical, experimental, and average results of maximum number of detours in a safety-level-based routing in n-cubes ($4 \leq n \leq 10$) with different selections of d_i ($d_i = 7$ and 9). Table 2(a) shows the results under consistent safety level information ($\lambda = n - 3$) and Table 2(b) shows the results under inconsistent safety level information ($\lambda = 2$).

In summary, the experimental results of the routing algorithm using safety level information are close to the results from the analytical study. The upper bound is rather accurate even under the inconsistent safety level information. Unlike routing algorithms without using any fault information, the upper bound of the maximum number of detours does not increase as much as the number of faults increases. Therefore, the safety-level-based information is scalable.

n	$d_i = 7, \lambda = n - 3$					$d_i = 9, \lambda = n - 3$				
	safety level			no info.		safety level			no info.	
	S	T	A	S	A	S	T	A	S	A
4	1	1	0.0024	1	0.0048	1	1	0.0024	1	0.0046
5	1	1	0.0014	2	0.0048	1	1	0.0012	2	0.0054
6	2	3	0.0002	4	0.0016	2	3	0.0002	4	0.0014
7	2	3	0.0000	8	0.0012	2	3	0.0000	8	0.0014
8	3	3	0.0000	12	0.0002	2	3	0.0000	12	0.0002
9	4	4	0.0000	20	0.0000	3	3	0.0000	20	0.0000
10	4	4	0.0000	30	0.0000	3	3	0.0000	30	0.0000

(a)

n	$d_i = 7, \lambda = 2$					$d_i = 9, \lambda = 2$				
	safety level			no info.		safety level			no info.	
	S	T	A	S	A	S	T	A	S	A
4	1	1	0.0024	1	0.0048	1	1	0.0024	1	0.0046
5	1	1	0.0014	2	0.0048	1	1	0.0012	2	0.0054
6	2	2	0.0006	4	0.0024	2	2	0.0002	4	0.0014
7	2	4	0.0002	8	0.0012	2	2	0.0001	8	0.0014
8	3	4	0.0002	12	0.0002	2	2	0.0001	12	0.0002
9	4	4	0.0000	20	0.0000	3	4	0.0000	20	0.0000
10	5	6	0.0000	30	0.0000	3	4	0.0000	30	0.0000

(b)

Table 2. Experimental maximum detours S, analytical maximum detours T and average detours A. (a) Routing with consistent information. (b) Routing with inconsistent information.

6. Conclusions

We have provided an upper bound of the maximum number of detours in hypercubes with dynamic faults using a fault-tolerant routing algorithm based on limited global information. The concept of safety level associated with each node has been used to represent limited global information. The safety level update includes fault detection, λ rounds of safety level exchanges and updates, message reception, routing decision, and message sending. An extended safety-level-based routing has been proposed which is applicable to hypercubes with up to $n - 2$ dynamic faulty nodes. Simulation results confirm the accuracy of our analytical study in an upper bound of the maximum number of detours. Applying this approach to other network topologies such as meshes and tori will be our future research.

References

[1] M. S. Chen and K. G. Shin. Processor allocation in an n-cube multiprocessor using Gary codes. *IEEE Transactions on Computers.* 36, (12), Dec. 1987, 1396-1407.

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

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

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

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