To appear in *IEEE Trans. on Computers*, 1989.

ADAPTIVE FAULT-TOLERANT ROUTING IN HYPERCUBE MULTICOMPUTERS

Ming-Syan Chen[‡] and Kang G. Shin[†]

[†] Real-Time Computing Laboratory
Department of Electrical Engineering and Computer Science
The University of Michigan
Ann Arbor, Michigan 48109-2122

[‡] I.B.M. Thomas J. Watson Research Center
P.O. Box 704
Yorktown Heights, New York 10598

ABSTRACT

A connected hypercube with faulty links and/or nodes is called an *injured hypercube*. To enable any non-faulty node to communicate with any other non-faulty node in an injured hypercube, the information on component failures has to be made available to non-faulty nodes so as to route messages around the faulty components.

We propose first a distributed adaptive fault-tolerant routing scheme for an injured hypercube in which each node is required to know only the condition of its own links. Despite its simplicity, this scheme is shown to be capable of routing messages successfully in an injured hypercube as long as the number of faulty components is less than n. Moreover, it is proved that this scheme routes messages via shortest paths with a rather high probability and the expected length of a resulting path is very close to that of a shortest path. Since the assumption that the number of faulty components is less than n in an n-dimensional hypercube might limit the usefulness of the above scheme, we also introduce a routing scheme based on depth-first search which works in the presence of an arbitrary number of faulty components.

Due to the insufficient information on faulty components, however, the paths chosen by the above scheme may not always be the shortest. To guarantee all messages to be routed via shortest paths, we propose to equip every node with more information than that on its own links. The effects of this additional information on routing efficiency are analyzed, and the additional information to be kept at each node for the shortest path routing is determined. Several examples and remarks are also given to illustrate our results.

Index Terms: Injured and regular hypercubes, distributed adaptive fault-tolerant routing, depth-first search, looping effects, network delay tables, failure information.

This work was supported in part by the Office of Naval Research under contracts N00014-85-K-0122 and N00014-85-K-0531, and the NASA under grant NAG-1-296. Any opinions, findings, and conclusions or recommendations expressed in this publication are those of the authors and do not necessarily reflect the view of the funding agencies.

1. INTRODUCTION

In recent years, advances of VLSI and computer networking technologies have made it attractive to build multicomputer systems for numerous applications. Hypercube multicomputers, among others, have been drawing considerable attention due mainly to their structural regularity for easy construction and high potential for the parallel execution of various algorithms [1, 2, 3]. Numerous research projects related to hypercube architectures, operating systems, and programming languages have been undertaken [4, 5, 6, 7, 8, 9], and several research and commercial hypercube multicomputers have been built [10, 11].

Efficient routing of messages is a key to the performance of a multicomputer system. Especially, the increasing use of multicomputer systems for reliability-critical applications has made it essential to design fault-tolerant routing strategies for such systems. By fault-tolerant routing, we mean the successful routing of messages between any pair of non-faulty nodes in the presence of faulty components (links and/or nodes). Several important results on the fault-tolerant routing in various networks have been reported [12, 13, 14]. A few architectures are also proposed and shown to possess fault-tolerant routing capabilities [15, 16, 17, 18]. When hypercube multicomputers are to be used for reliable applications, they must be made to be able to route messages even in the presence of faulty links and nodes.

A connected hypercube with faulty components is called an *injured hypercube*, whereas a hypercube without faulty components is called a *regular hypercube*. It is well-known that routing in a regular hypercube can be handled by a systematic procedure [9]. Some results on improving the routing efficiency in regular hypercubes are reported in [19]. Routing in an incomplete hypercube is also shown to be straightforward [20]. In a regular hypercube, each intermediate node can determine the next hop of a message by examining the message's destination address and choosing, from all its neighboring nodes, the one which is closest to the destination. Clearly, this can be accomplished by aligning the address of the source node with that of the destination node from right to left bit-by-bit. However, this scheme becomes invalid in an injured hypercube, since the message may be routed to a faulty component. In order to enable non-faulty nodes in an injured hypercube to communicate with one another, enough network information has to be incorporated into either the message to be routed or each node in the network so as to route messages around the faulty components. Using additional hardware called a *hyperswitch*, a best-first search algorithm for routing messages in a hypercube is developed in [12]. Several adaptive packet routing algorithms are also presented in [21], which are based on the routing strategies used for wide-area computer networks such as ARPANET. In addition, various algorithms are proposed in [14,22] to broadcast the information about faulty components to all the other nodes in a hypercube so that messages can be routed around the faulty components. Clearly, if each node is equipped with the information on all faulty components, then it can always determine a fault-free path for every message to its destination. However, it is usually too costly (in space and time) to equip every node with the information on all faulty components, especially when the network is large. Hence, it is important to develop routing schemes which require each node to keep only the failure information essential for making correct routing decisions.

For the reasons above, we shall develop first a routing scheme which requires each node to know only the condition of its own links. As will be seen later, this scheme attempts to route every message to its destination via an *optimal path* which is defined as a path of length equal to the Hamming distance[1] between the message's source and destination nodes. When the insufficient knowledge on faulty components causes a message to be sent to an intermediate node from which there is no optimal path to the destination node, an alternative path will be chosen in such a way that the connectivity of a hypercube is fully exploited, and those faulty components encountered before will not be encountered again in future. This scheme is proven to be capable of routing messages between any pair of non-faulty nodes as long as the total number of faulty components is less than n in an n-dimensional hypercube, Q_n. More importantly, this scheme,

[1]To be defined formally in the next section.

despite its simplicity, is shown to be very powerful in that the probability of routing messages via shortest paths is very high and the expected length of a resulting path is very close to the optimal one. To route a message in an injured hypercube, the information on component failures is incorporated into the message as the message travels toward the destination. This fact distinguishes our approach from others, such as those in [12, 21].

The assumption that the number of faulty components is less than n in an injured Q_n could limit the usefulness of the above algorithm. To remove this limitation, we introduce a second routing scheme based on depth-first search which requires more provisions but works in the presence of an *arbitrary* number of faulty components. Since the performance analysis we developed for the first routing scheme can be extended for the routing scheme based on depth-first search, the performance analysis of the latter is not included in this paper.

Due to the absence of information on all faulty components in the network, the paths chosen by the above two schemes may not be shortest. The efficiency of routing (measured in terms of the length of a path chosen) can be improved if every non-faulty node is equipped with more information than that on its own links, since in such a case the faulty components on the way of every message to its destination can be foreseen, and thus, bypassed. It can be observed that the abundant connections in a hypercube usually make routing decisions in a node unaffected by the failure of a component which is located far away from the node. Based on this observation, we shall develop a third routing scheme for which each node is required to keep only the information essential for the shortest path routing of messages. The information in each node required for the shortest path routing will be determined in light of some properties of the hypercube. Since the third method is again based on the assumption that the total number of faulty components is less than n in an injured Q_n, we introduce a fourth routing scheme which uses network delay tables [23] and works in the presence of an arbitrary number of faulty components. However, this scheme requires each node to maintain and update a delay table, which could be costly for large hypercubes.

The paper is organized as follows. Necessary notation and definitions are given in Section 2. We shall present in Section 3 an adaptive fault-tolerant routing scheme which requires each node to know only the condition of its own links. An alternative routing scheme based on depth-first search is also introduced and discussed there. Section 4 presents two fault-tolerant routing schemes which require each node to include more information than that on its own links: one with propagation of failure information, and the other with network delay tables. Illustrative examples and some remarks are also given. The paper concludes with Section 5.

2. PRELIMINARIES

An n-dimensional hypercube (or n-cube or Q_n) is formally defined as follows.

Definition 1: An n-cube, Q_n, is defined recursively as follows.

(i) Q_0 is a trivial graph with one node, and

(ii) $Q_n = K_2 \times Q_{n-1}$,

where K_2 is the complete graph with two nodes, Q_0 is a trivial graph with one node and $\times$ is the product operation of two graphs [24].

It follows from Definition 1 that a Q_n contains 2^n nodes and $n2^{n-1}$ links since the degree of each node in a Q_n is n. Let $\sum$ be the ternary symbol set {0, 1, *}, where * is a *don't care* symbol. Every subcube in a Q_n can then be uniquely represented by a string of symbols in $\sum$. Such a string of ternary symbols is called the *address* of the corresponding subcube. For example, the address of the subcube Q_2 formed by nodes 0010, 0011, 0110 and 0111 in a Q_4 is 0*1*. Fig. 1 shows a Q_2 with address 0*1* in a Q_4. The rightmost coordinate of the address of a subcube will be referred to as *dimension 1*, and the second rightmost coordinate as *dimension* 2, and so on. For each hypercube node, the communication link in dimension i is called the *i-th link* of this node. For notational simplicity, each link is represented by a binary string with a '-' symbol in the corresponding dimension. For example, the link between nodes 0000 and 0010 is represented by

00-0. Let r^+ and r^- denote the two end nodes of a link r, where r^+ (r^-) represents the node whose address is obtained by changing the '-' symbol in the link's address to 0 (1).

Definition 2: The *Hamming distance* between two hypercube nodes with addresses $u = u_n u_{n-1} \cdots u_1$ and $w = w_n w_{n-1} \cdots w_1$ in a Q_n is defined as

$$H(u, w) = \sum_{i=1}^{n} h(u_i, w_i), \text{ where } h(u_i, w_i) = \begin{cases} 1, & \text{if } u_i \neq w_i, \\ 0, & \text{if } u_i = w_i. \end{cases}$$

For the nature of distributed routing strategies to be presented, it is necessary to introduce the *exclusive* operation between two binary strings, and the concept of *relative address* between two hypercube nodes.

Definition 3: The *exclusive* operation of two binary strings $q = q_n q_{n-1} \cdots q_1$ and $m = m_n m_{n-1} \cdots m_1$, denoted by $q \oplus m = r_n r_{n-1} \cdots r_1$, is defined as $r_i = 0$ if $q_i = m_i$ and $r_i = 1$ if $q_i = \overline{m_i}$ for $1 \leq i \leq n$.

Obviously, the exclusive operation is commutative, i.e., $q \oplus m = m \oplus q$. We use $\bigoplus_{i=1}^{k}$ to denote k sequential exclusive operations. The relative address of a node u with respect to another node w, denoted by $u_{/w}$, can then be determined by $u_{/w} = u \oplus w$. The relative address of a subcube with respect to a node u can be determined by the relative addresses of all the nodes it contains. Let $e^k = e_n e_{n-1} \cdots e_1$ where $e_k = 1$ and $e_j = 0 \ \forall \ j \neq k$. For example, $1001 \oplus e^2 = 1011$, $0011_{/1001} = 1010$, and $0{*}1{*}_{/1001} = 1{*}1{*}$.

Definition 4: The *spanning subcube* of two nodes $u = u_n u_{n-1} \cdots u_1$ and $w = w_n w_{n-1} \cdots w_1$ in a Q_n, denoted by $SQ(u,w) = s_n s_{n-1} \cdots s_1$, is defined as $s_i = u_i$ if $u_i = w_i$, and $s_i = *$ if $u_i \neq w_i$ for $1 \leq i \leq n$.

For example, when $u = 0010$ and $w = 0111$, we get $H(u,w) = 2$ and $SQ(u,w) = 0{*}1{*}$. It is easy to see that SQ(u,w) is the smallest subcube that contains both u and w, and H(u,w) is the dimension of SQ(u,w).

A path in a hypercube is represented by a sequence of nodes in which every two consecutive nodes are physically adjacent to each other in the hypercube. The number of links on a path is called the *length* of the path. An *optimal path* is a path whose length is identical to the Hamming distance between the source and destination nodes. A *shortest path* is a path of minimal length among all fault-free paths from the source to the destination. Clearly, an optimal path is a shortest path, but a shortest path is not always an optimal path in an injured hypercube. Also, a link of node u is said to be *toward* any node w if this link is in an optimal path between u and w. Note that due to the special structure of a hypercube, once the source node of a path is given, the path can be described by a *coordinate sequence* that represents the order of the dimensions in which every two consecutive nodes in a path differ [25]. As shown in Fig. 2, [0001, 0011, 0010, 1010] is an optimal path from the source node 0001 to the destination node 1010, and can also be represented by a coordinate sequence [2, 1, 4]. In addition, we shall assume that the source and destination nodes are non-faulty.

3. ROUTING WITH INFORMATION ON LOCAL LINK FAILURES

In this section, we first develop and analyze an adaptive routing algorithm, called Algorithm A_1, which requires every node to know only the condition of its own links. This algorithm will be shown to successfully route messages between any pair of non-faulty nodes as long as the number of faulty components is less than n in a Q_n. As mentioned earlier, the assumption of the total number of faulty components to be less than n in a Q_n may limit the usefulness of A_1. Thus, we shall intorduce a second routing scheme based on depth-first search which works in the presence of an arbitrary number of faulty elements. However, due to the insufficient amount of information on faulty components, the paths chosen by these two algorithm may not always be the shortest.

3.1. Description of Algorithm A_1

Before describing Algorithm A_1, it is necessary to introduce the following lemma which determines relative addresses of those nodes traversed by a given path.

Lemma 1: Let $[c_1, c_2, \cdots, c_k]$ be the coordinate sequence of a given path in a Q_n starting from node u, and $w_{/u} = w_n w_{n-1} \cdots w_1$ denote the relative address of node w with respect to u, where $k = H(u,w)$. Then, the path specified by $[c_1, c_2, \cdots, c_k]$ ends at w if and only if $\bigoplus_{i=1}^{k} e^{c_i} = w_{/u}$.

Proof: Traversal of a message along the i-th dimension is the same as inverting the bit in the i-th coordinate of the relative address of its destination. Therefore, traveling along a certain dimension an even number of times has the same effect as not traveling along that dimension at all, and thus, this lemma follows. **Q.E.D.**

For example, a path with the coordinate sequence [3, 4, 2] from 0110 will traverse nodes 0010, 1010 and then 1000. The following theorem, which was previously introduced in [26], is useful for our discussion that follows.

Theorem 1 [26]: Let u and w be two arbitrary nodes in a Q_n such that $H(u,w) = k$. Then, there are exactly n disjoint paths of length less than or equal to k+2 from u to w. These paths are composed of k disjoint paths of length k, and $(n - k)$ disjoint paths of length k+2.

Fig. 3 gives an example for 4 disjoint paths in a Q_4 when $H(u,w) = H(0000, 0111) = 3$. Theorem 1 leads to the following corollary.

Corollary 1.1: Let f be the number of faulty links and g be the number of faulty nodes in an injured Q_n such that $f + g < n$. Then, there always exists at least one path of length less than or equal to k+2 between any two non-faulty nodes u and w, where $H(u,w) = k$.

We can now describe Algorithm A_1 as follows. To indicate the destination of a message, the coordinate sequence of a path is sent along with the message. Additionally, each message is accompanied with an n-bit vector $tag = d_n d_{n-1} \cdots d_1$ which keeps track of "spare dimensions" that are used to bypass faulty components. All bits in the tag are reset to zero when the source node begins routing of a message. Therefore, such a message can be represented as (k, $[c_1, c_2, \cdots, c_k]$, message, tag), where k is the length of the remaining portion of the path and is updated as the message travels towards the destination. A message reaches its destination when k becomes zero.

When a node receives a message, it will check the value of k to see if the node is the destination of the message. If not, the node will try to send the message along one of those dimensions specified in the remaining coordinate sequence. (Note that the coordinate sequence will also be updated as the message travels through the hypercube.) Each node will attempt to route messages via shortest paths first. However, if all the links in those dimensions leading to shortest paths are faulty, the node will use a spare dimension to route the message via an alternative path. (Recall that the spare dimensions are kept track of by a tag.) More formally, this routing scheme can be described in algorithmic form as follows.

Algorithm A_1: Fault-tolerant routing algorithm to be used by each node only with the information on its own links.

```
/* For each node receiving (k, $[c_1, c_2, \cdots, c_k]$, message, tag) */
if k=0 then {the destination is reached!}
    else begin
        /* Try to send the message along a dimension in the remaining coordinate sequence. */
        for j := 1, k do
            if (the $c_j$-th link is not faulty) then    ----(†)
            /* (†) is a conditional statement which will be modified later in Section 4.1. */
                begin
                    send (k−1, $[c_1, \cdots c_{j-1}, c_{j+1}, \cdots, c_k]$, message, tag) along the $c_j$-th link;
                    stop; /* terminate Algorithm $A_1$ */
                end_begin
    end_do

  /* If the algorithm is not terminated yet, all dimensions in the coordinate sequence are
  blocked because of faulty components and a spare dimension needs to be used. */
    for j := 1, k do /* record all blocked dimensions in tag. */
      $d_{c_j} := 1$
    end_do;

    $h := \min \{i : d_i = 0, 1 \le i \le n\}$; /* choose a spare dimension */
    $d_h := 1$; /* update the tag */
    send (k+1, $[c_1, c_2 \cdots, c_k, h]$, message, tag) along the h-th link;
    stop; /* terminate Algorithm $A_1$ */
  end_begin
```

Consider the Q_4 in Fig. 4, where links 0-01, 1-01 and 100- are faulty. Suppose a message, fm, is routed from u = 0110 to w = 1001. The original message in u = 0110 is (4, [1,2,3,4], fm, 0000). Following the execution of A_1, node 0110 sends (3, [2,3,4], fm, 0000) to node 0111 which then sends (2, [3,4], fm, 0000) to node 0101. Since the 3-*rd* dimensional link of 0101 is faulty, node 0101 will route (1, [3], fm, 0000) to 1101. However, since the 3-*rd* dimensional link of 1101 is faulty, node 1101 will use the 1-*st* dimension (tag = 0100 then), and send (2, [3,1], fm, 0101) to 1100, which will, in turn, send (1, [1], fm, 0101) to 1000. Again, the first link of node 1000 is faulty. The 2-*nd* dimension (tag= 0101 then) will be used and (2, [1,2], fm, 0111) is routed to 1010. After this, the message will reach the destination 1001 via 1011. The length of the resulting path is 8.

3.2. Performance Analysis of Algorithm A_1

The following theorem proves that Algorithm A_1 can route messages between any two non-faulty nodes as long as the number of faulty components is less than n.

Theorem 2: Algorithm A_1 can always route messages between any two non-faulty nodes successfully as long as the number of faulty components is less than n, i.e., $f + g < n$, where f and g are the numbers of faulty links and faulty nodes in a Q_n, respectively.

Proof: Note that each node will try to use a spare dimension only when faulty components are encountered in all the dimensions specified by the coordinate sequence. Those faulty components which block the optimal paths from an intermediate node to the destination node and force the first use of a spare dimension are called *type-A* blocking components. On the other hand, a faulty component is said to be *type-B* if it is encountered first after using a new spare dimension. For the example routing in Fig. 4, 1-01 is a type-A blocking component and 100- is a type-B blocking component, whereas the faulty link 0-01 is neither type-A nor type-B. For the example in Fig. 5 where u = 0000 and w = 1111, 0-11 and -011 are type-A blocking components while 111- is a type-B blocking component. Notice that both the types of blocking components can be either faulty nodes or faulty links. Thus, it is easy to see that the number of both type-A and type-B blocking components in the route determined by A_1 usually increases as the message is routed towards its destination.

Let b_h be a type-B blocking component which is encountered first after using a new spare dimension h. We claim that the blocking component b_h does not belong to the set of those blocking components that had already been encountered before. This claim is proved by considering two possible cases of b_h: (i) b_h is a link of the destination node, and (ii) b_h is not a link of the destination node. In the case of (i), b_h is the h-*th* link of the destination node. Since d_h of tag was 0 before the spare dimension h is used, this faulty link had definitely not been encountered before. In the case of (ii), since b_h is the blocking component encountered first after using the spare dimension h, b_h and the set of previous blocking components must be located in the two different Q_{n-1}'s separated by the dimension h. The claim is thus proved. Since a certain faulty component will not be encountered more than once as long as the number of faulty components is less than n, this theorem thus follows. **Q.E.D.**

The corollary below follows from Theorem 2 and the fact that the number of hops is increased by two whenever a spare dimension is used.

Corollary 2.1: Suppose k spare dimensions are used for routing a message from node u to node w by A_1. Then, the length of the resulting path is $H(u,w) + 2k$.

It can be easily verified that the worst case of Algorithm A_1 needs $H(u,w) + 2(n-1)$ steps to send a message from u to w. To facilitate our presentation, the first node which is forced to use a spare dimension is called an *obstructed node*. For example, the obstructed nodes in the examples of Figs. 4 and 5 are 1101 and 0011, respectively. Then, we have the following lemma.

Lemma 2: Suppose there are f faulty links in a Q_n, and a message is routed by A_1 from node u to node w, where $H(u,w) = k$. Let m_A be the Hamming distance between the obstructed node and the destination node. Then, $P(m_A=j) \leq C_{f-j}^{L-j}/C_f^L$ if $1 \leq j \leq k$, and $P(m_A=j) = 0$ if $j > k$, where C_y^x represents the combinations of choosing y out of x possibilities, and $L = n2^{n-1}$ is the number of links in a Q_n.

Proof: $P(m_A > k) = 0$, since the inequality $m_A > k$ represents an impossible case in which a message is not directed towards its destination before encountering the obstructed node. Consider the case of $1 \leq j \leq k$ and assume there are f faulty links in an injured Q_n. Since these faults may occur at any f links in the Q_n, there are C_f^L different configurations (of faulty links) where $L = n2^{n-1}$. Without loss of generality, we can let $u = 0^n$ and $w = 0^{n-k}1^k$. The problem of obtaining $P(m_A=j)$ is then reduced to that of counting the number of configurations which lead to the case of $m_A = j$. We claim that the number of such configurations is less than or equal to C_{f-j}^{L-j}.

When $m_A = j$, the obstructed node must be within the subcube $0^{n-k}*^k$, and all its j links towards w must be faulty (i.e., j type-A blocking components). Although there are many possible locations of the obstructed node, according to the systematic procedure of A_1, the location of the obstructed node is determined by those non-type-A faulty links which are not within $0^{n-k}*^k$. Suppose $x = 0^{n-k+j}1^{k-j}$ is the obstructed node, then the j links of node x within SQ(x,w) are faulty and there are C_{f-j}^{L-j} different distributions of these non-type-A faulty links. When these non-type-A faulty links cause node y, instead of x, to be the obstructed node, we switch the links (including faulty links) in SQ(y,w) to those in SQ(x,w), and obtain a configuration which leads to the case when the obstructed node is y and $m_A = j$. Notice that some of the C_{f-j}^{L-j} different distributions of non-type-A faulty links may lead to $m_A > j$, meaning that the number of configurations leading to $m_A = j$ is less than or equal to C_{f-j}^{L-j}. This lemma thus follows. **Q.E.D.**

From Lemma 2, we can obtain the following theorem, showing that Algorithm A_1 can route a message to the destination via an optimal path with a very high probability.

Theorem 3: Suppose there are f faulty links in a Q_n. Algorithm A_1 will route a message from a node u to to another node w via an optimal path between u and w with a probability greater than $1 - \sum_{j=1}^{k} \frac{C_{f-j}^{L-j}}{C_f^L}$, where $L = n2^{n-1}$, and H(u,w) = k.

Proof: From Lemma 2, the probability that A_1 has to use spare dimensions is $\sum_{j=1}^{k} P(m_A = j) \leq \sum_{j=1}^{k} \frac{C_{f-j}^{L-j}}{C_f^L}$. Thus, the probability that A_1 will not use any spare dimension at all is $1 - \sum_{j=1}^{k} P(m_A = j) \geq 1 - \sum_{j=1}^{k} \frac{C_{f-j}^{L-j}}{C_f^L}$. **Q.E.D.**

When there are n−1 faulty links in a Q_n, the lower bound of the probability that A_1 will result in the optimal path routing can be derived as follows.

Corollary 3.1: Suppose there are n−1 faulty links in a Q_n. Algorithm A_1 will route a message from a node u to another node w via an optimal path between u and w with a probability greater than $1 - \frac{r_1(1 - r_1^k)}{(1-r_1)}$, where H(u,w) = k and $r_1 = \frac{n-1}{n2^{n-1}}$.

Proof: From Theorem 3, we have

$$\sum_{j=1}^{k} \frac{C_{n-1-j}^{L-j}}{C_{n-1}^{L}} = \frac{n-1}{L} + \frac{(n-1)(n-2)}{L(L-1)} + \cdots + \frac{(n-1)(n-2)\cdots(n-k)}{L(L-1)\cdots(L-k+1)}, \text{ where } L = n2^{n-1}.$$

Since $r_1 = \frac{n-1}{L} > \frac{n-2}{L-1} > \cdots \frac{n-k}{L-k+1}$, we get

$$\sum_{j=1}^{k} \frac{C_{n-1-j}^{L-j}}{C_{n-1}^{L}} < r_1 + r_1^2 + \cdots + r_1^k 1 - \sum_{j=1}^{k} \frac{C_{n-1-j}^{L-j}}{C_{n-1}^{L}} > 1 - \frac{r_1(1 - r_1^k)}{(1-r_1)}.$$

This corollary thus follows. **Q.E.D.**

From Theorem 3, we derive Table 1 which shows the lower bound of the probability of A_1 routing messages via optimal paths (we shall henceforth call this *optimal path routing*) between two nodes of Hamming distance n−1 apart in a Q_n with f faulty links. It can be seen from Theorem 3 that A_1 will route a message to its destination via an optimal path with a rather high probability in the presence of faulty links. Notice that the expression in Theorem 3 can also be applied to the case that the number of faulty links is greater than n in a Q_n. This is the very reason that we included such cases as n=3 and f=5 in Table 1, i.e., f>n. Similarly to the case of faulty links, the performance of A_1 can be analyzed in terms of *node* failures as follows.

Lemma 3: Suppose there are g faulty nodes in a Q_n, and messages are to be routed from an arbitrary node u to another node w, where H(u,w) = k. Let m_B be the Hamming distance between the obstructed node and the destination node w. Then, $P(m_B=j) \leq \frac{C_{g-j}^{N-3-j}}{C_g^{N-2}}$ if $2 \leq j \leq k$, and $P(m_B = j) = 0$ if $j = 1$ or $j > k$, where $N = 2^n$ is the total number of nodes in a Q_n.

Proof: Following the same reasoning as in the proof of Lemma 2, we get $P(m_B>k) = 0$. Besides, $P(m_B=1) = 0$ since the destination node is assumed to be non-faulty. Next, let a configuration and the obstructed node be defined analogously to the case of faulty links. (Link failures are replaced by node failures.) There are C_g^{N-2} different configurations for a Q_n with g faulty nodes since the source and destination nodes are assumed to be non-faulty. In order to determine $P(m_B=j)$, we need to count the number of configurations which lead to the case of $m_B = j$. By the same reasoning as the one used in the proof of Lemma 2, the number of such configurations is less than C_{g-j}^{N-3-j}. Notice that we use C_{g-j}^{N-3-j} instead of C_{g-j}^{N-2-j} because the obstructed node must be non-faulty, and this lemma thus follows. **Q.E.D.**

From Lemma 3 and the reasoning in the proof of Theorem 3, we can obtain Theorem 4 and its corollary, showing that A_1 can also route a message between any pair of non-faulty nodes in an injured hypercube via an optimal path with a high probability.

Theorem 4: Suppose there are g faulty nodes in a Q_n. Algorithm A_1 will route a message from a node u to another node w via an optimal path between u and w with a probability greater than $1 - \sum_{j=2}^{k} \frac{C_{g-j}^{N-3-j}}{C_g^{N-2}}$, where $H(u,w) = k$.

Corollary 4.1: Suppose there are n–1 faulty nodes in a Q_n. Algorithm A_1 will route a message from a node u to another node w via an optimal path between u and w with a probability greater than $1 - \frac{(n-1)r_2(1-r_2^{k-1})}{(2^n-2)(1-r_2)}$, where $H(u,w) = k$, and $r_2 = \frac{n-2}{2^n-3}$.

Proof: Notice that

$$\frac{C_{n-1-j}^{N-3-j}}{C_{n-1}^{N-2}} = \frac{(n-1)(n-2)\cdots(n-j+1)(n-j)(N-n-1)}{(N-2)(N-3)\cdots(N-j-2)}$$

$$\leq \frac{(n-1)(n-2)\cdots(n-j+1)(n-j)}{(N-2)(N-3)\cdots(N-j-1)} \quad \text{(Since } N-n-1 \leq N-j-2.\text{)}$$

$$= \frac{n-1}{N-2} r_2^{j-1}.$$

By letting $g = n-1$ in Theorem 4 we get $1 - \sum_{j=2}^{k} \frac{C_{n-1-j}^{N-3-j}}{C_{n-1}^{N-2}} > 1 - \sum_{j=2}^{k} \frac{n-1}{N-2} r_2^{j-1} =$ $1 - \frac{n-1}{N-2} \sum_{j=2}^{k} r_2^{j-1} = 1 - \frac{(n-1)r_2(1-r_2^{k-1})}{(N-2)(1-r_2)}$, where $N = 2^n$. **Q.E.D.**

Again, from Theorem 4, we derive Table 2, showing the lower bound of the probability for A_1 to route messages via optimal paths between two nodes of Hamming distance n–1 apart in a

Q_n with g faulty nodes. Furthermore, as it will be shown below, the expected length of a path resulting from the use of A_1 is very close to that of an optimal path, i.e., the Hamming distance between the source and destination nodes. Before analyzing the quality of the paths selected by A_1, it is necessary to introduce the following proposition.

Proposition 1: Let $\{p_i\}_{i=1}^{n}$ and $\{q_i\}_{i=1}^{n}$ be, respectively, two decreasing sequences with $p_n = q_n = 0$. Suppose $p_i \le q_i$ for $1 \le i \le n-1$, then $\sum_{i=1}^{n-1} i(p_i - p_{i+1}) \le \sum_{i=1}^{n-1} i(q_i - q_{i+1})$.

Proof: Let $d_i = q_i - p_i$ for $1 \le i \le n$. Then, we get

$$\sum_{i=1}^{n-1} i(q_i - q_{i+1}) - \sum_{i=1}^{n-1} i(p_i - p_{i+1})$$

(Since $d_i \ge 0$ for $1 \le i \le n-1$.) **Q.E.D.**

We can now derive the following important theorem.

Theorem 5: Let u and v be a pair of nodes with $H(u,w) = n$ in an injured Q_n which contains n–1 faulty links. Let H_1 be the length of a path between u and w that is chosen by A_1. Then, $E(\Delta H_1) \le \frac{n-1}{2^{n-2}}$, where E(x) denotes the expected value of a random variable x, and $\Delta H_1 = H_1 - n$.

Proof: Notice that $P(\Delta H_1 \ge 2i) \le \sum_{j=1}^{n-i} P(m_A = j)$. Then,

$$\begin{aligned} E(\Delta H_1) &= \sum_{i=1}^{n-1} 2i\, P(\Delta H_1 = 2i) \\ &= \sum_{i=1}^{n-1} 2i \left[P(\Delta H_1 \ge 2i) - P(\Delta H_1 \ge 2(i+1)) \right] \\ &\le \sum_{i=1}^{n-1} 2i \left[\sum_{j=1}^{n-i} P(m_A = j) - \sum_{j=1}^{n-i-1} P(m_A = j) \right] \quad \text{(By Proposition 1.)} \\ &= \sum_{i=1}^{n-1} 2i\, P(m_A = n-i) \end{aligned}$$

$$\le \sum_{i=1}^{n-1} 2(n-i)\frac{C_{n-1-i}^{L-i}}{C_{n-1}^{L}} \quad \text{(By Lemma 2 and } L = n2^{n-1}.)$$

$$< \sum_{i=1}^{n-1} 2(n-i)r_1^i \quad (r_1 = \frac{n-1}{L})$$

$$= 2n\sum_{i=1}^{n-1} r_1^i -$$

$$= 2nr_1 + 2nr_1(r_1 + r_1^2 + \cdots + r_1^{n-2})$$

$$< 2nr_1 = \frac{n-1}{2^{n-2}}. \quad \text{(Since } nr_1 < 1.) \quad \textbf{Q.E.D.}$$

Using the same reasoning in the proof of Theorem 5, we have the following corollary for an injured Q_n with n−1 faulty components.

Corollary 5.1: Suppose messages are to be routed from an arbitrary node u to another node w in an injured Q_n which contains n−1 faulty *nodes*. Let H_2 be the length of a path between u and w that is chosen by A_1, and let $\Delta H_2 = H_2 - n$, where $H(u,w) = n$. Then, $E(\Delta H_2) \le \frac{2n(n-1)(n-2)}{(2^n-2)(2^n-3)}$.

3.3. Depth-First Search Routing and Other Remarks

Note that in the presence of more than n−1 faulty components in a Q_n, Algorithm A_1, due to its simplicity, cannot ensure a faulty component not to be encountered more than once, and thus, cannot always lead to a successful message routing. Depth-first search can be applied to deal with the problem of routing messages between connected pairs of non-faulty nodes in a hypercube with an arbitrary number of faulty components [27]. In such a case, a set of faulty components encountered before has to be added to the message, and a more complicated procedure is required to guide the backtracking whenever it is forced to backtrack from a deadend. Instead of keeping track of the entire path traveled, the depth-first search routing can also be implemented by using a stack, in which case the operations required for backtracking are simplified, but additional provisions are needed to ensure that a node will not be visited more than once. Note that our results on the performance analysis of A_1 can be extended and applied to the analysis of the depth-first search routing. In order not to distract the readers from the main theme of this paper, we have not included here such an extended analysis. Interested readers are referred to [28].

Despite its limiting assumption on the number of faults, using the concept of spare dimensions, Algorithm A_1 only needs to keep an n-bit vector (tag), and is shown to be very simple and powerful. One cannot overemphasize two important aspects of this algorithm: the ability to route messages via optimal paths with a high probability, and the fact that the expected length of a resulting path is very close to the Hamming distance between the source and destination nodes.

However, due to the absence of information at each node on components other than its own links, the presence of a certain faulty component is not known until a message gets to the faulty component. This may force an intermediate node to use a spare dimension for routing messages around the faulty component, thus increasing the length of the actual path taken. Consequently, in order to route messages more efficiently, each node needs to be equipped with more information than that on its own communication links such that the faulty components on a path to the destination can be bypassed.

4. ROUTING WITH LIMITED GLOBAL INFORMATION

As mentioned before, routing efficiency can be improved by increasing each node's information on component failures. We shall examine the effects of failure information at each node on the efficiency of routing. First, we shall propose and analyze in Section 4.1 a routing scheme in which the condition of each component is not only known to its adjacent nodes but also available to those nodes one hop away from that component. Although this scheme improves routing efficiency, the paths chosen are not guaranteed to be the shortest. Thus, we shall in Section 4.2 investigate the information essential for the shortest path routing. The approach of using network delay tables [23] to the fault-tolerant routing in hypercubes with an arbitrary number of faults will be discussed in Section 4.3.

4.1. Propagation of Failure Information to Neighboring Nodes

Consider a simple modified scheme of Algorithm A_1. In addition to keeping track of the condition of its own links, every node also makes this information available to all its neighboring nodes. Thus, every node will know not only the condition of its own links but also that of those links which are one hop away from it. To use this information, the conditional statement in A_1 marked with (†) is modified in such a way that every intermediate node checks one more hop in the coordinate sequence of each message, and uses a spare dimension to bypass faulty components if necessary. From a reasoning similar to the one in the proof of Theorem 2, it can be verified that this modified routing scheme can also successfully route a message to any other node if the number of faulty components is less than n. More specifically, the performance of this modified routing scheme can be described by the following theorem.

Theorem 6: Suppose a message is to be routed in an injured Q_n with f faulty links from node u to node w where H(u,w) = k. Let m_C be the Hamming distance between the obstructed node and the destination when each node is informed about the conditions of its own links as well as those links one hop away from the node. Then, $P(m_C=j) \leq \sum_{i=1}^{j} C_i^j C_{f-ij-j+2i}^{L-ij-j+2i} / C_f^L$ if $2 \leq j \leq \min\{1+\lceil \frac{f}{2} \rceil, k-1\}$, $P(m_C=k) \leq \sum_{i=0}^{k} C_i^k C_{f-ik-k+2i}^{L-ik-k+2i} / C_f^L$, and $P(m_C=j) = 0$ otherwise.

Proof: Let x denote the obstructed node. First, consider the case when the obstructed node is the source node, i.e., $m_C = k$. Clearly, x has k links within SQ(x,w). There will be no optimal path from x to w if and only if each of these k links is either faulty or connected to a neighboring node with k−1 faulty links toward w. Suppose x has exactly i non-faulty links toward w. Then, there are C_i^k ways to determine which of x's links are non-faulty. For each case, there are (k−i) + i(k−1) specific faulty links, thus resulting in $C_{f-ik-k+2i}^{L-ik-k+2i}$ different configurations of faulty links. The expression for the case of $P(m_C = k)$ thus follows.

Since every node is informed about the condition of the links one hop away, a message will not be routed to any neighboring node whose every link toward w is faulty. Thus, every link of the obstructed node toward w can be faulty only when the obstructed node is the source node. This is the very reason that different expressions are needed for the cases of $2 \leq m_C \leq k-1$ and $m_C = k$. Note, however, that when $m_C = j \neq k$ and the obstructed node x has i non-faulty links toward w, these i links are connected to those nodes with j−1 faulty links toward w. Therefore, from the fact that the total number of faulty links $i(j-1) + j - 1$ is less than f, we get $j \leq 1 + \lceil \frac{f}{2} \rceil$, and thus, this theorem follows. **Q.E.D.**

To illustrate the improvement of routing efficiency with the above additional information at each node, let $h_A(j) = C^{L-j}_{f-j}/C^L_f$ and $h_C(j) = \sum_{i=1}^{j} C^j_i C^{L-ij-j+2i}_{f-ij-j+2i}/C^L_f$. As shown in Lemma 2 and Theorem 6, $P(m_A=j) \leq h_A(j)$ and $P(m_C=j) \leq h_C(j)$. It can be verified that $h_C(j) < h_A(j)$ for $j > 2$, meaning that $P(m_C=j)$ has a smaller upper bound than $P(m_A=j)$. Note, however, that $h_C(2) > h_A(2)$. This is based on the fact that, under our modified routing scheme, an intermediate node located two hops away from the destination may foresee the unreachability from itself to the destination and thus use spare dimensions to bypass those faulty components which could not be seen by the same intermediate node under A_1. The upper bound of $P(m_C = 2)$ is thus greater than that of $P(m_A = 2)$. (Note, however, that $h_C(2) < h_A(2) + h_A(1)$.) Therefore, in light of the reasoning in Theorem 5, routing efficiency is improved with the additional information at each node.

Note, however, that even the above routing scheme cannot guarantee the shortest path routing. When a pair of nodes communicate with each other frequently, it is desirable to have the shortest path routing, since extra hops will otherwise be traveled during each transmission. Although each node can always find a shortest fault-free path from itself to any other node if it contains the information on every faulty component, it is impractical to maintain and update such information, especially when the size of the network is very large. Therefore, it is important to determine the information required for the shortest path routing.

4.2. Routing via Shortest Paths

To reduce the amount of information at each node required for the shortest path routing, the unnecessary propagation of information on faulty components should be avoided. Notice that a faulty node can be viewed as a node with its all links faulty. Therefore, the network information kept at each node can be represented by a set of addresses of faulty links. As will be proved later, when the number of faulty components is less than n in a Q_n, each node does not have to propagate the information on faulty links to its neighboring nodes unless these faulty links block all the optimal paths from itself to another node. In other words, only when node u finds that all its optimal paths to another node, say x, have been blocked by a set of faulty links F, will node u propagate the information on F to its neighboring nodes so as to prevent them from choosing node u as a next hop toward node x.

Note that the coordinate sequence of an optimal path from u to w consists of H(u,w) different numbers representing those dimensions in which u and w differ, meaning that there are H(u,w)! different optimal paths from u to w. Then, we have the following proposition, describing the effect of a link failure on the optimal paths between the two nodes.

Proposition 2: Let $N(u\backslash w, r)$ be the number of optimal paths from u to w which traverse a link r. Then,

(i). For any link $r \in SQ(u,w)$, $N(u\backslash w, r) = [H(u,w)-H(u,x)-1]!H(u,x)!$, where x is the one of the link r's two end nodes that is closer to u.

(ii). For any link $r \notin SQ(u,w)$, $N(u\backslash w, r) = 0$.

For example, if u = 0100, w = 1001 and r = 0-01, then x = 0101 and $N(u\backslash w, r) = 2!1! = 2$. On the other hand, if r = 0-11, then $N(u\backslash w, r) = 0$, since $0\text{-}11 \notin SQ(u,w) = {*}{*}0{*}$. In light of Proposition 2, we can derive the condition for a set of faulty links to block all the optimal paths between any given pair of nodes as follows. Let r_x and r_y be the relative addresses of any two links with respect to node u. Then, the link r_y is called a *downstream* link of r_x, written as $r_x < r_y$, if r_x appears before r_y in an optimal path from u to $\bar{r_y}$. Note that $r_x = x_n x_{n-1} \cdots x_1 < r_y = y_n y_{n-1} \cdots y_1$ iff (1) $x_i \leq y_i$ for $1 \leq i \leq n$, where the symbol '-' is ordered such that $0 \leq - \leq 1$, and (2) r_x and r_y are the links placed in different dimensions. This fact results in a straightforward procedure for determining if a link is a downstream link of another. Each node can thus store the relative addresses of faulty links as a partially ordered set [29]. A set of relative addresses of faulty links is called a *linear set* if for any two links r_x and r_y in the set, either $r_x < r_y$ or $r_y < r_x$. It can be verified that an optimal path from u to w will traverse all links in a set of links B only if B is a linear set. Let $M(u\backslash w, F)$ be the number of optimal paths from u to w which traverse *every* link in F. Then, following the concept of exclusion and inclusion [30], we obtain the following theorem which can determine the number of optimal paths blocked by a set of faulty links.

Theorem 7: Given a set of faulty links F, the number of the optimal paths from u to w blocked by the links in F is $N(u\backslash w, F) = \sum_{i=1}^{r}(-1)^{i+1}m_i$, where $m_i = \sum_{B_i \in F \cap SQ(u,w)} M(u\backslash w, B_i)$ and B_i denotes a linear set of i links.

Clearly, the condition for a set of faulty links to block all the optimal paths between two nodes u and w is $N(u\backslash w, F) = H(u,w)!$. The operations of Algorithm A_2 can be outlined as follows. Each node keeps the information about two types of faulty links in the form of relative addresses. The first type, denoted by F_0, is the set of those faulty links whose status has not yet been propagated to neighboring nodes, whereas the second type, denoted by F_1, is the set of those faulty links whose status has already been propagated to neighboring nodes.

Since relative addresses of faulty links of a node are kept in that node, the information on F_0 must be modified in accordance with the addresses of receiving nodes when it is propagated to neighboring nodes. A formal description of the algorithm for the determination and modification of link failure information is given below.

Algorithm A_2: Collection of failure information for the shortest path rou

Testing

```
/* Each node tests all its communication links.*/
        if (the k-th link of the node is faulty) then
        begin
          $F_1 := F_1 \cup \{e^k\}$;
          for i := 1 to n do
            if $i \neq k$ then send $e^k \oplus e^i$ along the i-th dimension;
          Propagation;
        end
```

Receiving

```
/* For each node receiving the information on the failure of the link r.*/
        if $r \notin F$ then        /*$F = F_0 \cup F_1$ */
          begin
            $F_0 := F_0 \cup \{r\}$;
            Propagation;
          end
```

Propagation

```
    if $N(0^n \setminus r^-, F) = H(0^n, r^-)!$ then
        begin
          $F_0 := F_0 - \{r\}$;
          $F_1 := F_1 \cup \{r\}$;
          /* Propagate the information on the failure of r to neighboring nodes.*/
          for i := 1 to n do send $r \oplus e^i$ along the i-th dimension;
          /* Check if propagation of information on other faulty links is necessary.*/
          for all $r_x \in F_0$ and $r \in SQ(0^n, r_x^-)$ do
            if $N(0^n \setminus r_x^-, F) = H(0^n, r_x^-)!$ then
              begin
                /* Propagate the information on the failure of r to neighboring nodes.*/
                for i := 1 to n do send $r_x \oplus e^i$ along the i-th dimension;
                  $F_0 := F_0 - \{r_x\}$;
                  $F_1 := F_1 \cup \{r_x\}$;
              end
        end
```

When a node receives the information on the failure of link r, it will update its F_0 and F_1 accordingly, and check if any further propagation of information on other link failures in F_0 is required. For example, consider the Q_4 in Fig. 4. By executing A_2, each node can determine $F = F_0 \cup F_1$ as well as the information to be propagated to neighboring nodes, F_1. Table 3 shows the information to be kept in each node. Notice that the faulty links are represented in each node by their relative addresses with respect to the node. Then, we have the theorem below.

Theorem 8: Under Algorithm A_2, every node can obtain the failure information essential for the shortest path routing as long as the number of faulty components is less than n.

Proof: Notice that the necessary and sufficient condition for all the optimal paths from node u to node w to be blocked is that "for all $z \in SQ(u,w)$ reachable from u via an optimal path, then there is no optimal path from z to w." Since every node propagates the corresponding failure information to its neighboring nodes if all its optimal paths to a certain node are blocked, the fact that every node will know if all its optimal paths to a certain node are blocked can be proved by induction.

When node u finds all its optimal paths to w are blocked, there are at least $H(u,w) = k$ faulty components in SQ(u,w). Note that there are still $n - k$ disjoint paths of length k+2 via the neighboring nodes of u which are not within SQ(u,w), and at least one of them is fault-free because there are at most $n - 1$ faulty components in the Q_n. Since those neighboring nodes not having any optimal path to w will propagate the corresponding failure information to u to prevent u from choosing one of them as a next hop, this theorem follows. **Q.E.D.**

Theorems 1 and 8 lead to the following corollary.

Corollary 8.1: Algorithm A_2 can route a message from node u to node w in H(u,w)+2 hops as long as the number of faulty components is less than n.

When a node needs to send a message to another node, it will use its information on faulty components to determine the coordinate sequence of a shortest fault-free path to the destination node as if it had the information on every faulty component in the entire network. Then, according to the first entry of the coordinate sequence, the source node will determine the next hop of the message and its associated coordinate sequence. On the other hand, when a node receives a message and the coordinate sequence of the remaining part of the path, it will check whether the remaining path contains faulty links and permute the order of entries in the coordinate sequence to bypass the faulty components, if necessary.

For example, consider the injured Q_4 in Fig. 4. The source node is not aware of any faulty link, and thus, routes a message (3, [2,3,4], fm) to 0111. However, the node 0111 will find the path [2,3,4] is faulty, since the path will encounter the faulty link 0-01 whose relative address is $0\text{-}01_{/0111}$ =0-10. Thus, a new non-faulty path [3,2,4] is determined by 0111. The message will be routed to 0011, and then to the destination 1001 via 1011. The length of the resulting path is 4. This is far less than the length of the path determined by A_1, 8.

It is interesting to see that the information about an isolated faulty link needs to be propagated only to its neighboring nodes, whereas the information about clustered faulty links has to be propagated a little farther to ensure each message to be routed via a shortest path. For example, node 1111 has to be informed by node 1101 about the failure 0-01 (two hops away) and 1-01 (one hop away), since all the optimal paths from 1101 to 0001 are blocked by the failure of 0-01 and 1-01. This agrees well with our intuition, since clustered faulty components are likely to block more optimal paths between a pair of nodes, and thus, have to be kept by those nodes far away from them to achieve the shortest path routing. Clearly, when the size of the hypercube increases, faulty components will tend to spread out and the size of the zone influenced by a faulty component will become rather small relative to the size of the entire network.

Notice, however, when the number of faulty components is more than n−1 in a Q_n, each node may not be able to gather enough information required for the shortest path routing, since too many faulty components may block the propagation of failure information.

4.3. Routing with Delay Tables

In the presence of more than n−1 faulty components in a Q_n, the concept of using network delay tables, which was previously used in the ARPANET [23], can be applied to accomplish the shortest path routing. Under the algorithms in [23], every node maintains a network delay table to record the shortest delay via each link of the node to every other node. When a node is to send a message to another node, it will check its network delay table and determine the next hop of the message for the shortest path routing. A *minimal delay vector* in a node, which contains the delays of the shortest paths from that node to all the other nodes, is periodically passed to all of its adjacent nodes as routing information. After receiving minimal delay vectors from its adjacent nodes, every node will update its network delay table accordingly. For example, the network delay tables for nodes 000, 100 and 101 in Fig. 6 are given in Table 4a, 4b and 4c, respectively. The routing information generated by node 100 is also shown in Table 4d. As we pointed out in [31], looping may occur in the presence of component failures when this routing scheme is used. The approach of using high-order routing strategies [32] can be applied to eliminate the looping.

It is worth mentioning that when the number of faulty components is less than n in a Q_n, Algorithm A_2 is shown to be capable of routing messages via shortest paths without using network delay tables, and is thus preferred over the one based on network delay tables. Note that it becomes very costly to maintain and update network delay tables as the size of a hypercube gets large. It is therefore advantageous to use A_2, whenever possible.

5. CONCLUSION

In this paper, we have proposed and analyzed two distributed routing schemes (A_1 and A_2), and introduced two more schemes (using depth-first search and network delay tables), to route messages in injured hypercube multicomputers. A_1 is very simple and powerful. It requires each node to know only the failure of its own links and uses the abundant connections in hypercubes. Performance of this scheme has been rigorously analyzed; we showed that this scheme is not only capable of routing messages successfully in an injured Q_n when the number of component failures is less than n, but also able to choose a shortest path with a very high probability. To handle the case when the total number of faults is greater than n−1 in a Q_n, we introduced a routing scheme based on depth-first search. However, due to the insufficient amount of information on faulty components, these two schemes do not always guarantee the shortest path routing.

To ensure the shortest path routing, we proposed A_2 which requires each node to be equipped with more information than that on its own links. We developed a method which determines the failure information essential for each node to guarantee the shortest path routing. It turns out that each node is required to know only the condition of a relatively small number of components in its vicinity. In case there are more than n−1 faults in a Q_n, we can use a more expensive routing scheme based on network delay tables.

Due to their simplicity and/or power, the fault-tolerant routing algorithms derived in this paper have high potential use for the growing number of fault-tolerant applications on large hypercube multicomputers.

REFERENCES

[1] P. J. Denning, ''Parallel Computing and Its Evolution,'' *Commun. of the Assoc. Comp. Mach.*, vol. 29, no. 12, pp. 1163-1167, Dec. 1986.

[2] P. Wiley, ''A Parallel Architecture Comes of Age at Last,'' *IEEE Spectrum*, vol. 24, no. 6, pp. 46-50, Jun. 1987.

[3] L. G. Valiant, ''A Scheme for Fast Parallel Communication,'' *SIAM J. on Computing*, vol. 11, no. 2, pp. 350-361, May, 1982.

[4] M. S. Chen and K. G. Shin, ''Processor Allocation in an N-Cube Multiprocessor Using Gray Codes,'' *IEEE Trans. on Comput.*, vol. C-36, no. 12, pp. 1396-1407, Dec. 1987.

[5] L. N. Bhuyan and D. P. Agrawal, ''Generalized Hypercube and Hyperbus Structures for a Computer Network,'' *IEEE Trans. on Comput.*, vol. C-33, no. 4, pp. 323-333, Apr. 1984.

[6] T. F. Chan and Y. Saad, ''Multigrid Algorithms on the Hypercube Multiprocessor,'' *IEEE Trans. on Comput.*, vol. C-35, no. 11, pp. 969-977, Nov. 1986.

[7] M. - S. Chen and K. G. Shin, ''On the Relaxed Squashed Embedding of Graphs into a Hypercube,'' *SIAM J. on Computing*, 1989 (in press).

[8] B. Becker and H. U. Simon, ''How Robust is the n-Cube?,'' *Proc. 27-th Annual Symposium on Foundations of Computer Science*, pp. 283-291, Oct. 1986.

[9] Y. Saad and M. H. Schultz, *Data Communication in Hypercubes*. Dep. Comput. Sci., Yale Univ. Res. Rep. 428/85., 1985.

[10] C. L. Seitz, ''The Cosmic Cube,'' *Commun. of the Assoc. Comp. Mach.*, vol. 28, no. 1, pp. 22-33, Jan. 1985.

[11] NCUBE Corp., ''NCUBE/ten: an overview'', Beaverton, OR., Nov. 1985.

[12] E. Chow, H. S. Madan, and J. C. Peterson, ''An Adaptive Message-Routing Network for the Hypercube Computer,'' *Proc. of the Third Conf. on Hypercube Concurrent Computers and Applications*, Jan. 19-20, 1988.

[13] D. K. Pradhan, ''Fault-Tolerant Multiprocessor Link and Bus Network Architectures,'' *IEEE Trans. on Comput.*, vol. C-34, no. 1, pp. 33-45, Jan. 1985.

[14] J. G. Kuhl and S. M. Reddy, ''Distributed Fault Tolerance for Large Multiprocessor Systems,'' *Proc. 7-th Annual Int'l Symposium on Computer Architecture*, pp. 23-30, May 1980.

[15] A. H. Esfahanian and S. L. Hakimi, ''Fault-Tolerant Routing in DeBruijn Communication Networks,'' *IEEE Trans. on Comput.*, vol. C-34, no. 9, pp. 777-788, Sep. 1985.

[16] A. Ghafoor, T. R. Bashkow, and I. Ghafoor, ''Fault-Tolerance and Diagnosability of Bisectional Interconnection Networks,'' *Proc. 6-th Int'l Conf. on Distributed Computing Systems*, pp. 62-69, 1986.

[17] D. K. Pradhan and S. M. Reddy, ''A Fault-Tolerant Communication Architecture for Distributed Systems,'' *IEEE Trans. on Comput.*, vol. C-31, no. 9, pp. 863-870, Sep. 1982.

[18] A. Varma and C. S. Raghavendra, ''Fault-Tolerant Routing of Permutations in Extra-Stage Networks,'' *Proc. 6-th Int'l Conf. on Distributed Computing Systems*, pp. 54-61, 1986.

[19] C. T. Ho and S. L. Johnsson, ''Distributed Routing Algorithms for Broadcasting and Personalized Communication in Hypercubes,'' *Proc. Int'l Conf. on Parallel Processing*, pp. 640-648, Aug. 1986.

[20] H. Katseff, ''Incomplete Hypercube,'' *Hypercube Multiprocessors, edited by M. T. Heath*, pp. 258-264, 1987.

[21] C. K. Kim and D. A. Reed, ''Adaptive Packet Routing in a Hypercube,'' *Proc. of the Third Conf. on Hypercube Concurrent Computers and Applications*, Jan. 1988 .

[22] J. R. Armstrong and F. G. Gray, ''Fault Diagnosis in a Boolean n Cube Array of Microprocessors,'' *IEEE Trans. on Comp.*, vol. C-30, no. 8, pp. 587-590, Aug. 1981.

[23] J. M. McQuillan and D. C. Walden, ''The ARPA Network Design Decisions,'' *Computer Networks*, vol. 1, no. 5, pp. 243-289, Aug. 1977.

[24] F. Harary, *Graph Theory*. Mass.: Addison-Wesley, 1969.

[25] E. N. Gilbert, ''Gray Codes and Paths on The N-cube,'' *Bell System Tech. J.*, vol. 37, pp. 263-267, 1973.

[26] Y. Saad and M. H. Schultz, ''Topological Properties of Hypercubes,'' Res. Rep. No. 389/85., Dep. Comput. Sci., Yale University , 1985.

[27] A. V. Aho, J. E. Hopcroft, and J. D. Ullman, *The Design and Analysis of Computer Algorithms*. Reading, Mass: Addison-Wesley Publishing Co., 1974.

[28] M. - S. Chen and K. G. Shin, ''Fault-Tolerant Routing in Hypercube Multicomputers Using Depth-First Search,'' Technical Report No. TR-89-006, International Computer Science Institute, 1947 Center Street, Berkeley, CA 94704, 1989.

[29] K. A. Ross and R. B. Wright, *Discrete Mathematics*. Englewood Cliffs, NJ.: Prentice Hall, 1985.

[30] C. L. Liu, *Introduction to Combinatorial Mathematics*. New York: McGraw Hill, 1968.

[31] K. G. Shin and M. S. Chen, ''Performance Analysis of Distributed Routing Strategies Free of Ping-Pong-Type Looping,'' *IEEE Trans. on Comput.*, vol. C-36, no. 2, pp. 129-137, Feb. 1987.

[32] K. G. Shin and M. S. Chen, ''Minimal Order Loop-Free Routing Strategy,'' *IEEE Trans. on Comput.*, 1989 (in press).

n \ f	1	2	3	4	5	6
3	0.916	0.818	0.700	0.557	0.386	0.182
4	0.968	0.935	0.900	0.862	0.821	0.777
5	0.987	0.974	0.961	0.948	0.934	0.920

Table 1. Lower bounds of the probabilities obtained from Theorem 3 for the optimal path routing between two nodes of distance n apart in the presence of link failures.

n \ g	2	3	4	5	6
4	0.989	0.967	0.934	0.891	0.838
5	0.998	0.993	0.986	0.977	0.965
5	0.999	0.998	0.996	0.994	0.992

Table 2. Lower bounds of the probabilities obtained from Theorem 4 for the optimal path routing between two nodes of distance n apart in the presence of node failures.

Nodes	F_0	F_1
0000	{0-01,1-01,100-}	∅
0001	{100-}	{0-00,1-00}
0010	∅	∅
0011	{0-10,1-10}	∅
0100	{0-01,1-01,110-}	∅
0101	∅	{0-00,1-00}
0110	∅	∅
0111	{0-10,1-10}	∅
1000	{0-01}	{000-}
1001	∅	{1-00,0-00,000-}
1010	{001-}	∅
1011	{1-10,0-10,001-}	∅
1100	{1-01}	{0-01,010-}
1101	{010-}	{1-00,0-00}
1110	{0-11,011-}	∅
1111	{1-10,0-10}	∅

Table 3. Information in each node generated by Algorithm A_2 for the injured hypercube in Fig. 4.

destination / dimension	001 *	010	011	100	101	110 *	111
1 (001)	∞	∞	∞	∞	∞	∞	∞
2 (010)	3	1	2	3	4	2	3
3 (100)	5	3	4	1	2	4	3

(a). Network delay table of node 000.

destination / dimension	000	001 *	010	011	101	110 *	111
1 (101)	3	4	4	3	1	3	2
2 (110)	∞	∞	∞	∞	∞	∞	∞
3 (000)	1	4	2	3	3	3	4

(b). Network delay table of node 100.

destination / dimension	000	001 *	010	011	100	110 *	111
1 (100)	2	5	3	4	1	4	3
2 (111)	4	3	3	2	3	2	1
3 (110)	∞	∞	∞	∞	∞	∞	∞

(c). Network delay table of node 101.

000	001 *	010	011	101	110	111
1	4	2	3	1	3	2

(d). Routing information generated by node 100.

Table 4. Network delay tables for hypercube nodes in Fig. 6.

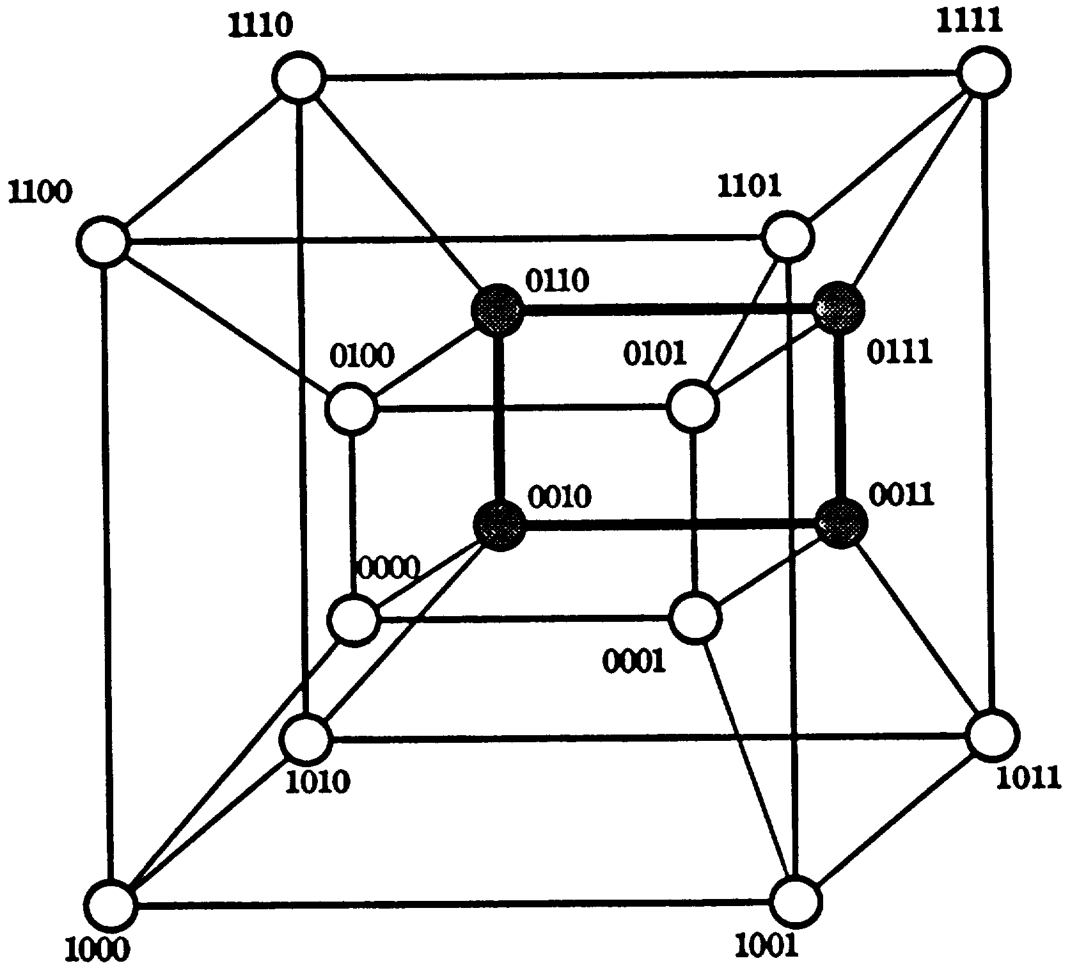

Figure 1. A Q_2 with address 0*1* in a Q_4.

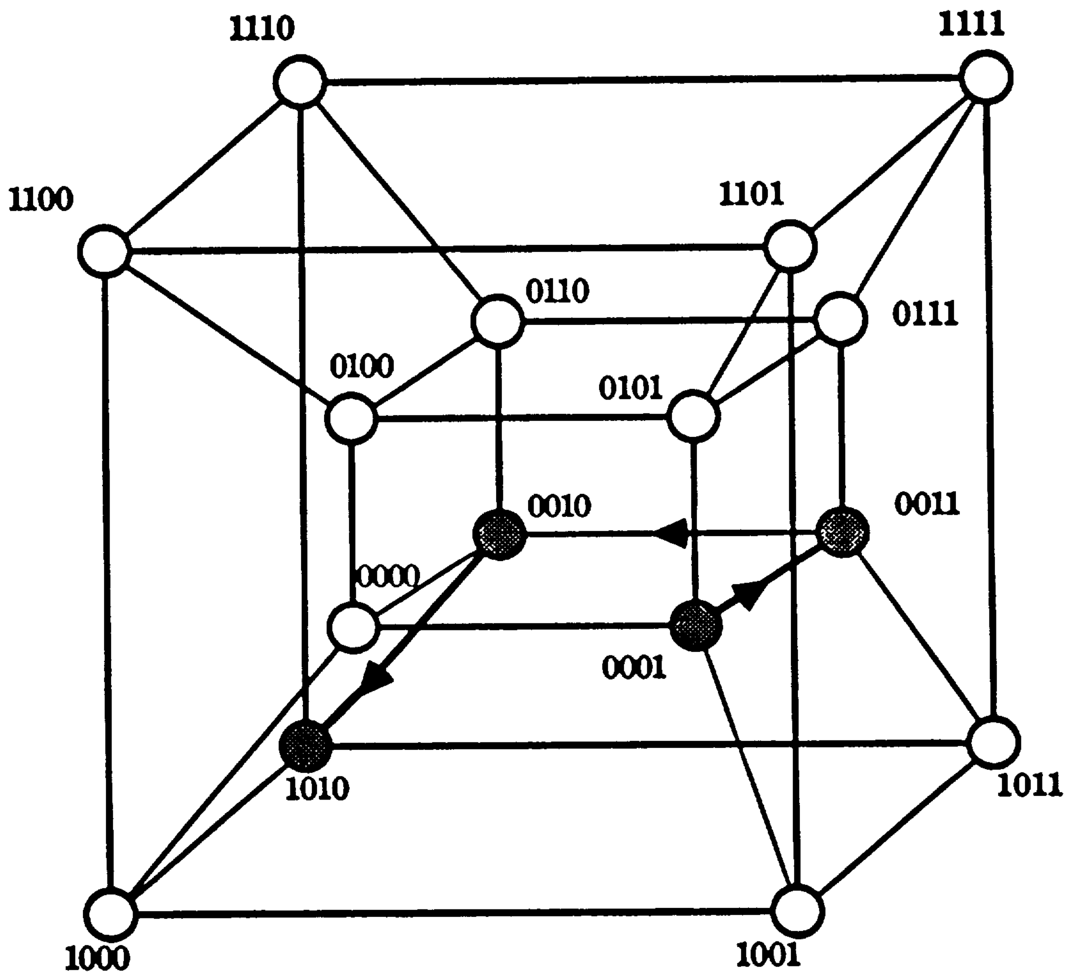

Figure 2. An optimal path from 0001 to 1010 .

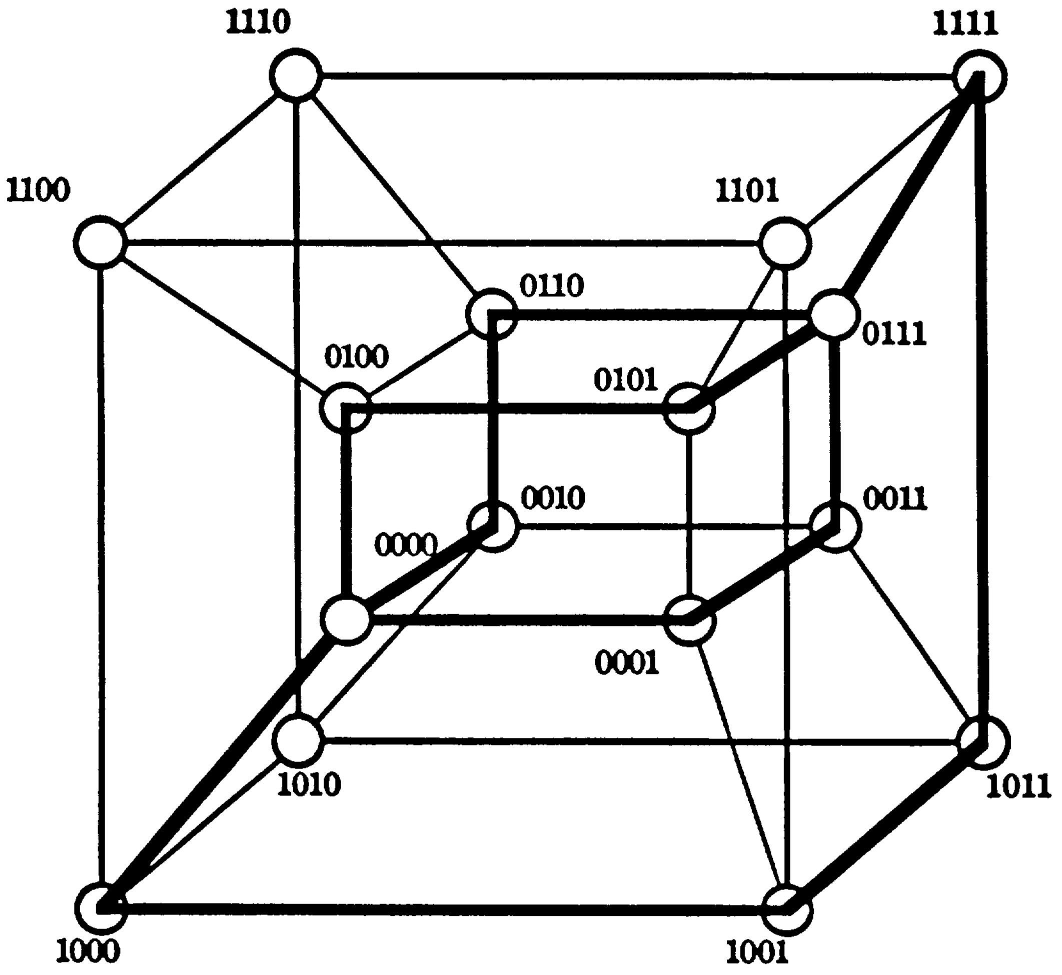

Figure 3. Four disjoint paths from 0000 to 0111.

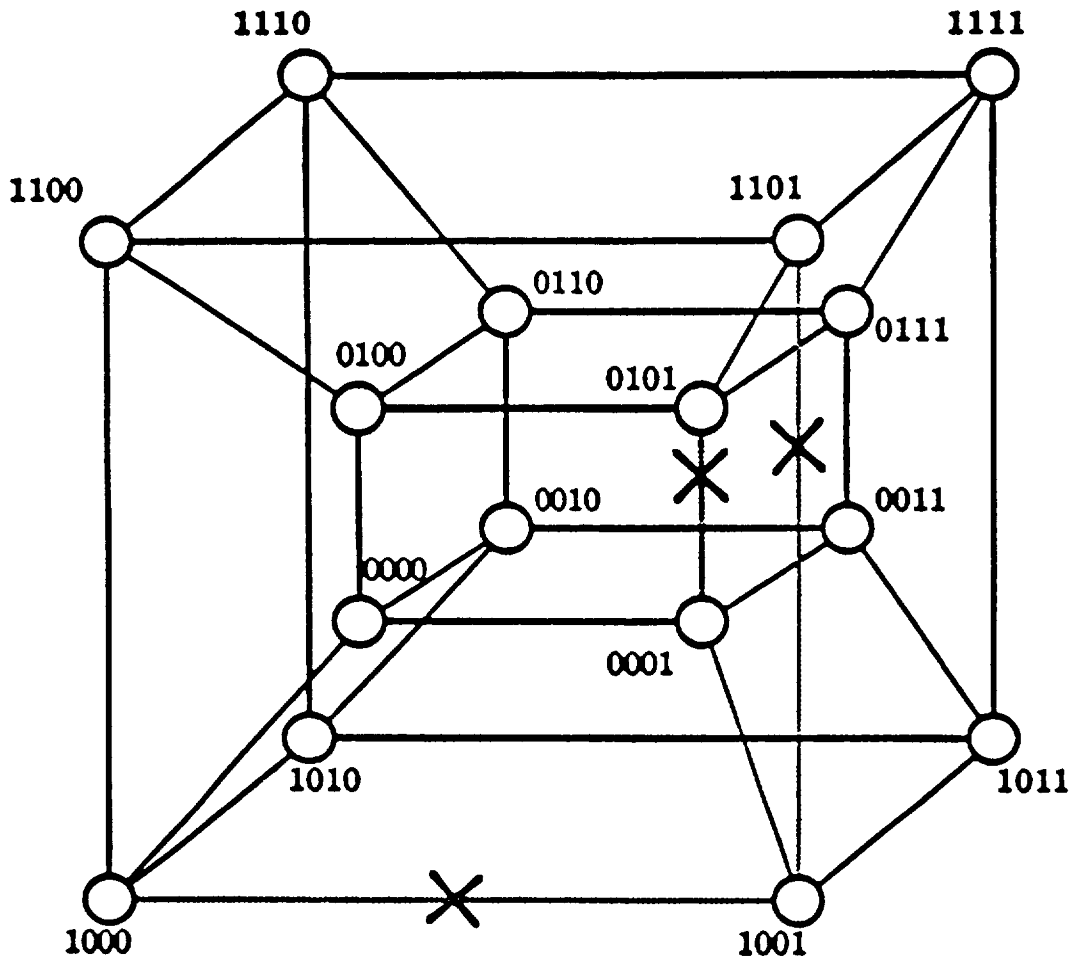

Figure 4 . An injured Q_4 where links 0-01, 1-01 and 100- are faulty.

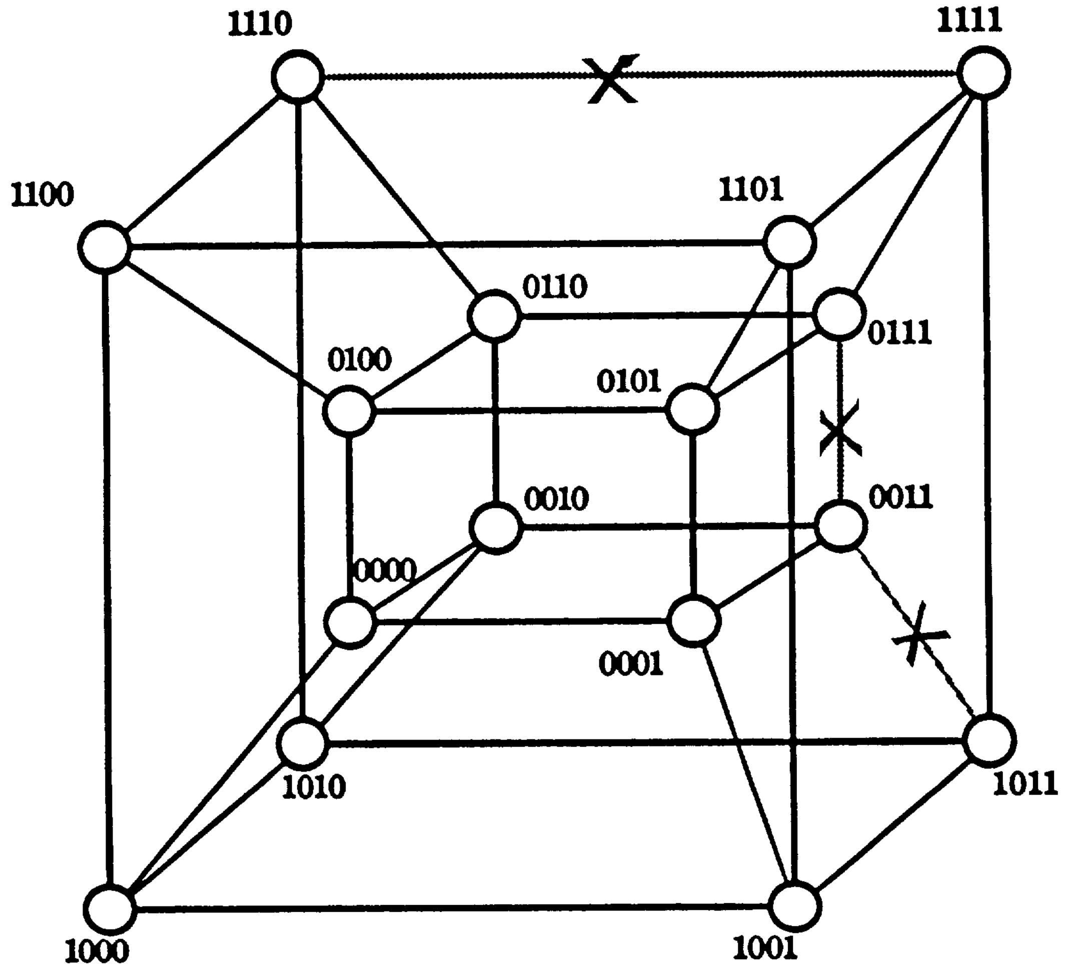

Figure 5 . An injured Q_4 where links 0-11, -011 and 111- are faulty.

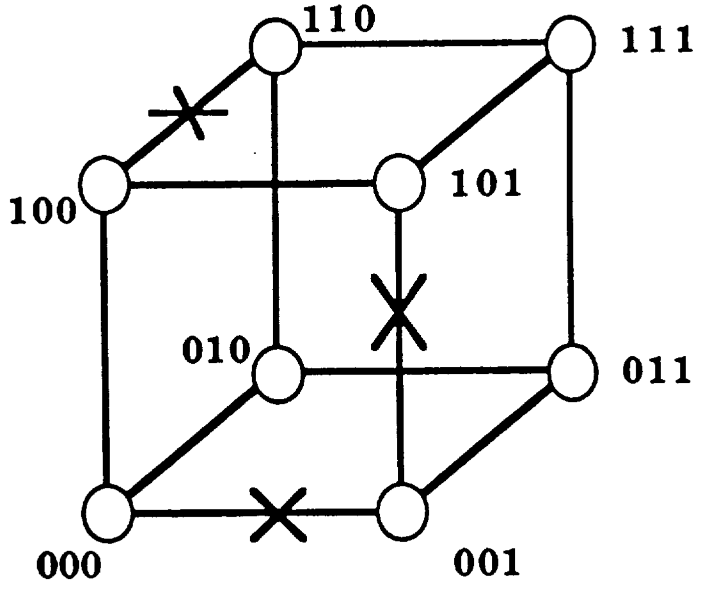

Figure 6 . An example Q_3 for the routing scheme based on the minimal delay tables.