Depth-First Search Approach for Fault-Tolerant Routing in Hypercube Multicomputers

MING-SYAN CHEN, MEMBER, IEEE, AND KANG G. SHIN, SENIOR MEMBER, IEEE

Abstract—**Using depth-first search, we develop and analyze the performance of a routing scheme for hypercube multicomputers in the presence of an arbitrary number of faulty components. We derive an exact expression for the probability of routing messages via optimal paths (of length equal to the Hamming distance between the corresponding pair of nodes) from the source node to an *obstructed node*. The obstructed node is defined as the first node encountered by the message that finds no optimal path to the destination node. Also, bounds for this probability are derived in closed form. Note that the probability of routing messages via an optimal path between any two nodes is a special case of our results, and can be obtained by replacing the obstructed node with the destination node. Numerical examples are also given to illustrate our results, and show that in the presence of component failures the depth-first search routing can route a message via an optimal path to its destination with a very high probability.**

Index Terms—**Coordinate sequences, depth-first search, fault-tolerant routing, hypercube multicomputers, inversions.**

I. Introduction

OWING to their structural regularity and high potential for the parallel execution of various algorithms, hypercube multicomputers have drawn considerable attention in recent years from both academic and industrial communities [1]–[10]. Topological properties of hypercubes were investigated in [1]. Task and subcube allocation schemes were proposed to exploit the parallelism within a hypercube multicomputer [2]–[4]. Numerous applications on hypercube multicomputers were explored [5]–[7], and several research and commercial hypercube machines were built [8]–[10]. Efficient message routing is a key to the performance of distributed memory multicomputers, such as hypercubes [11]. To make hypercube multicomputers useful for reliability-critical applications, significant research efforts have been made on the design of fault-tolerant routing schemes for them [12]–[20].

A connected hypercube with faulty components is called an *injured hypercube*. In order to enable nonfaulty nodes in an injured hypercube to communicate with one another, enough network information must be either kept at each node or added to the message to be routed. For the first approach, several algorithms have been proposed in [12]–[15], which generally require each hypercube node to keep a certain amount of global information for its routing decisions. This, however, necessitates the propagation/broadcasting of updated network information when the network condition changes. Some broadcasting schemes in injured hypercubes can also be found in [15]–[17]. For the second approach, in which each node is required to know only the condition (faulty or not) of its adjacent components (links or nodes), several routing schemes based on the concept of depth-first search and its variation have been proposed [18]–[20]. Under a depth-first search routing scheme, each message is accompanied with a stack which keeps track of the *history* of the path traveled, and tries to avoid visiting a node more than once unless a backtracking is enforced. Although the concept of depth-first search is well understood and appears to be able to exploit the abundant connections in the hypercube and thus achieve a high degree of fault-tolerance, neither formal presentation nor rigorous performance analysis of this scheme has been reported in the literature. Development and analysis of a fault-tolerant routing scheme using depth-first search for hypercube multicomputers is, therefore, the subject of this paper.

We shall first develop a routing scheme using depth-first search, in which every node is only required to know the condition of its adjacent components. The path that a message has traversed is kept track of by the message as it is routed toward its destination. Performance of this routing algorithm will be rigorously analyzed. An *optimal* path between a pair of nodes is a path of length equal to the Hamming distance between the two nodes. Under this routing scheme, the first node in the message's route that is aware of the nonexistence of an optimal path from itself to the destination is called the *obstructed node*. At the obstructed node, the message has to take a detour. In this paper, we derive exact expressions for the probabilities of optimal path routing from the source node to a given obstructed node in the presence of both link and node failures. Bounds for these probabilities are also derived in closed form, which will be shown to closely approximate the exact expressions. Note that determination of the probability for optimal path routing between any two nodes is a special case of our results since the destination node can be viewed as an obstructed node that is 0 hop away from the destination node. Numerical examples are also given to illustrate our results. It can be seen that in the presence of component failures, the depth-first search routing can route a message via

Manuscript received May 15, 1989; revised December 11, 1989. This work was supported in part by the Office of Naval Research under Contract N00014-85-K-0122 and NASA under Grant NAG-1-296.

M.-S. Chen is with I.B.M. Thomas J. Watson Research Center, Yorktown Heights, NY 10598.

K. G. Shin is with Real-Time Computing Laboratory, Department of Electrical Engineering and Computer Science, The University of Michigan, Ann Arbor, MI 48109.

IEEE Log Number 8934117.

an optimal path to its destination with a very high probability.

It is worth mentioning that the concept of depth-first search is not only useful for the fault-tolerant routing in hypercubes, but also applicable to fault-tolerant and multipath routing in other multicomputer systems such as square meshes and hexagonal meshes [21], [22]. Note, however, that the depth-first search routing must be guided to fully exploit the connections in each multicomputer system, and thus, the method of guiding depth-first search closely depends on system topology.

This paper is organized as follows. Necessary notation and definitions are introduced in Section II. A fault-tolerant routing scheme using depth-first search for hypercube multicomputers is presented in Section III. The performance of this routing scheme is analyzed rigorously in Section IV where illustrative examples are also given. The paper concludes with Section V.

II. Preliminaries

An n-dimensional hypercube is defined recursively as $Q_n = K_2 \times Q_{n-1}$ 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 on two graphs [23]. 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 Σ be the ternary symbol set $\{0, 1, *\}$ where $*$ is a *don't care* symbol. Every subcube in a Q_n can then be uniquely addressed by a string of symbols in Σ. The rightmost coordinate of the address of a subcube will be referred to as *dimension 1*, and the second to the rightmost coordinate as *dimension 2*, and so on. For each hypercube node, the communication link in dimension i is called the *ith link* of the node. For notational simplicity, each link is represented by a binary string with a "-" symbol in its corresponding dimension. For example, the link between nodes 0000 and 0010 is represented by 00-0.

Definition 1: 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

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

Also, it is necessary to introduce the *exclusive* operation between two binary strings, and the concept of *relative address* between two hypercube nodes.

Definition 2: 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 $1 \leq i \leq n$.

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 $u_{/w} = u \oplus w$. 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 $0^*1^*_{/1001} = 1^*1^*$.

A path in a hypercube is represented by a sequence of nodes in which every two consecutive nodes are physically adjacent to each other. Thus, a path in a Q_n can be viewed as a sequence of binary numbers of n bits such that every two consecutive binary numbers differ by exactly one bit. The number of links on a path is called the *length* of the path. An optimal path is a path whose length is equal to the Hamming distance between the source and destination nodes. We shall call the routing via an optimal path the *optimal path routing*. Also, a link of node u is said to be *toward* another node w if the link belongs to one of the optimal paths from u to w.

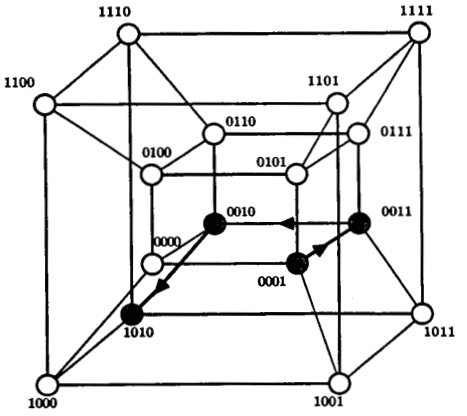

Fig. 1. An optimal path from 0001 to 1010.

Note that due to the special structure of a hypercube, once the source node of a path of length k in a Q_n is given, the path can be described by a *coordinate sequence* $C = [c_1, c_2, \cdots, c_k]$ where $1 \leq c_i \leq n$ for $1 \leq i \leq k$. A coordinate sequence is a sequence of ordered dimensions in each of which the corresponding two adjacent nodes in the path differ [24]. A coordinate sequence is said to be *simple* if any dimension does not occur more than once in that sequence. It is easy to see that a path is optimal if and only if its coordinate sequence is simple. As shown in Fig. 1, [0001, 0011, 0010, 1010] is an optimal path from the source 0001 to the destination 1010, and can also be represented by a coordinate sequence [2, 1, 4].

Definition 3 [25]: The number of *inversions* of a simple coordinate sequence $C = [c_1, c_2, \cdots, c_k]$, denoted by $V(C)$, is the number of pairs (c_i, c_j) such that $1 \leq i < j \leq k$ and $c_i > c_j$.

For example, $V([3,4,1,2]) = 4$ and $V([2,4,1]) = 2$. Let the superscript R denote the *reversing* of a coordinate sequence, and the notation $\odot$ denote an *append* operation. Also, the ith element in a coordinate sequence C is denoted by $C[i]$. For example, if $C = [2,4,1]$ is the coordinate sequence of a path, then $C[2] = 4$, $C^R = [1,4,2]$ and $C \odot 3 = [2,4,1,3]$.

III. Depth-First Search Routing

In this section, we present an adaptive routing algorithm based on depth-first search, which requires every node to know only the condition (faulty or not) of its own links. The case that a node is faulty is treated as that all links of the node are faulty. This algorithm can successfully route messages between any pair of connected, nonfaulty nodes. When

the insufficient knowledge of faulty components causes a message to be sent to an intermediate node from which no optimal path to the destination node exists, an alternative path will be chosen in such a way that the connectivity of a hypercube is fully exploited. Before presenting the routing algorithm, it is necessary to introduce the following proposition which determines, from the coordinate sequence of a path, the relative addresses of those nodes traversed.

Proposition 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: A message's traversal along the ith dimension is the same as inverting the bit in the ith coordinate of the relative address of its destination. Therefore, traversing along a certain dimension an even number of times has the same effect as not traversing along that dimension at all. This proposition thus follows. Q.E.D.

Let $S(C)$ denote the set of the relative addresses of those nodes reachable by the coordinate sequence $C = [c_1, c_2, \cdots, c_k]$ from a given node. By Proposition 1, $S(C) = \{\bigoplus_{i=1}^{r} e^{c_i} : 1 \leq r \leq k\}$. For example, a path with the coordinate sequence [2,1,4] from 0000 will traverse the set of nodes $S(C) = \{0010, 0011, 1011\}$.

To indicate the destination of a message, the relative address of the destination node is sent along with the message. The depth-first search routing algorithm will attempt to avoid visiting the same node more than once except when backtracking is forced. Thus, those dimensions that a message traversed so far are recorded in a set TD in the same order if visited, and will be delivered together with the message to the next node. Note that using TD^R each intermediate node can determine the addresses of those nodes visited before where TD^R is the reversing of TD as defined in Section II. Clearly, sending TD along with the message is much cheaper than sending the addresses of all the nodes visited. When the source node begins routing a message, TD is set to the empty set ϕ. Therefore, the information to be phased on to the next node can be represented as (d, message, TD) where d is the relative address of the destination node with respect to an intermediate node, and is updated as the message is routed toward the destination. A message reaches its destination when d becomes 0^n.

When a node receives a message, it will check the value of d to see if the destination is reached. If not, the intermediate node will try to send the message along an optimal path to the destination. However, if all the optimal paths are blocked by faulty components and those nodes visited before, the node will route the message via an alternative path using the concept of depth-first search. When there is no alternative path available, backtracking is enforced. More formally, this routing scheme can be described in an algorithmic form as follows.

Algorithm G: Depth-first search routing algorithm.

```
/* For each node receiving (d, message, TD). */
if d = 0^n then stop /* The destination is reached. */
else
begin
    for j := 1, n do
    begin
        if (d_j = 1) and (the j-th link is not faulty) and (e^j ∉ S(TD^R)) then
          begin
              send (d ⊕ e^j, message, TD ⊙ j) along the j-th link;
              stop; /*terminate Algorithm G*/
          end
    end

        /* If the algorithm is not terminated yet, all optimal paths to the destination node are blocked
        by faulty components and nodes traversed before.*/
        if {i : e^i ∉ S(TD^R) and the i-th link is not faulty, 1 ≤ i ≤ n} ≠ φ
            then h := min_{1≤i≤n} {i : e^i ∉ S(TD^R) and the i-th link is not faulty} /* A detour is taken. */
            else
            begin /* Backtracking */
                  g := max{m : ⊕_{i=1}^{m} e^{TD^R[i]} = 0^n};
                  if g = |S(TD)| then stop /* The source and destination nodes are not connected. */
                  h := TD^R[g + 1];
            end
    send (d ⊕ e^h, message, TD ⊙ h) along the h-th link;
    stop; /*terminate Algorithm G*/
end
```

Note that an intermediate node can determine whether or not its ith link is connected to a node that was visited before by checking if e^i belongs to $S(TD^R)$. When backtracking is forced, the message must be returned to the node from which this message was originally received. This explains the way we determine the value g in the segment commented by /* Backtracking */ in the above algorithm. When $g = |S(TD)|$,

we know that backtracking is enforced at the source node, and can thus conclude that the source and destination nodes are disconnected.

Note that there is an unavoidable overhead required to achieve fault-tolerant routing. To route a message to its destination in an injured hypercube, those nodes traversed before by the message must be made known to the intermediate nodes so as to avoid message looping. This is the very reason that under G every intermediate node has to append to the message a tag TD, from which the addresses of nodes traversed before can be recovered in light of the topology of a hypercube. Depth-first search is thus ensured at the cost of carrying TD with each message. The values of g and h in G can be easily determined from TD. It can be verified that the complexity of determining g and h is $O(|\mathrm{TD}|)$. Note that when the probability of the occurrence of a faulty component is low, the above scheme can be modified to reduce the operational overhead in such a way that the tag TD is appended to the message only after faulty components are encountered by the message. However, when the probability of the occurrence of a faulty component is high, it is worth using the tag TD from the beginning so as to fully exploit different paths and avoid sending the message back to a node which received the message before. There are some other techniques conceivable to implement the depth-first search routing for hypercubes. Instead of keeping the entire path traveled in TD, the search can also be implemented by using a stack. In such a case, the operation required for backtracking is simplified, but additional provisions are needed to ensure that a node will not be visited more than once.

Consider an injured Q_4 in Fig. 2. Suppose a message fm is routed from $u = 0110$ to $w = 1001$. The original message at $u = 0110$ is (1111, fm, ϕ). Following the execution of G, node 0110 sends (1110, fm, [1]) to node 0111 which then sends (1100, fm, [1,2]) to node 0101. Since the third dimensional link of 0101 is faulty, node 0101 will send (0100, fm, [1,2,4]) to 1101. However, since the third dimensional link of 1101 is faulty, node 1101 will choose an alternative path, and send (0101, fm, [1,2,4,1]) to 1100, which will, in turn, send (0001, fm, [1,2,4,1,3]) to 1000. Then, since links of 1000 in dimensions 1, 2, and 4 are faulty, backtracking is enforced and the message is sent back to 1100. It can be verified that the message is thereafter sent to its destination via 1110, 1111, 1011, and then 1001. The following proposition follows directly from the fact that depth-first search is a general graph search algorithm and can start from any node in a graph [26].

Proposition 2: Algorithm G will route a message to its destination successfully as long as the destination is reachable.

Algorithm G is a generalized version of the algorithm presented in [18] where the number of faults is assumed to be less than n. Also, from the fact that those links traversed will not form a cycle, we have the following proposition.

Proposition 3: The nodes and links traversed under G form a tree.

Proposition 3 in turn leads to another proposition.

Proposition 4: The worst case of G uses $H(u, w)+2(2^n - H(u, w) - 1)$ hops to send a message from node u to node w for a pair of connected nodes u and w.

Proof: Let $H(u, w) = k$, then there are at most $2^n - k$ nodes where backtracking may occur under G. These nodes will form a search tree with $2^n - k$ nodes and $2^n - k - 1$ links. Since every link in the search tree is traversed twice, the message will traverse $2(2^n - k - 1)$ hops in the search tree and k hops on its optimal path to the destination. Q.E.D.

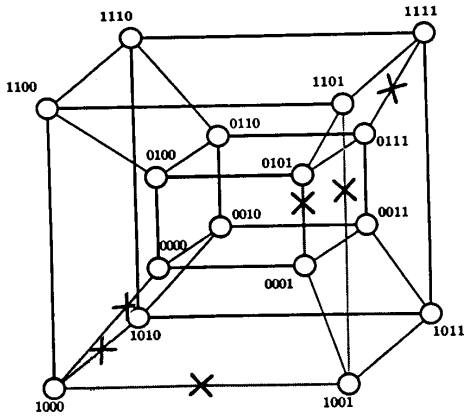

Fig. 2. An injured Q_4.

IV. Performance Analysis of Depth-First Search Routing

To illustrate the performance of depth-first search routing, let us consider an injured hypercube with a given number of faulty components. All possible distributions of faulty components are assumed to be equally likely. The number of faulty nodes required for a simple coordinate sequence C to be the path found by depth-first search is equal to the number of inversions in C [19]. This fact can be formally stated as follows.

Proposition 5: Suppose $H(u, w) = n$ in a Q_n. Then, under G the minimum number of faulty nodes required for the simple coordinate sequence $C = [c_1, c_2, \cdots, c_n]$ to be the path determined by G from u to w is $V(C)$.

For example, let 0000 and 1111 be the source and destination nodes, respectively. Then, the set of nodes whose failures will make G choose the coordinate sequence [3,4,1,2] is {0001, 0010, 0101, 0110}. In light of the fact that the addresses of hypercube nodes form a partial order set, the effects of a faulty node, say x, are the same as those of a faulty link via which an intermediate node is trying to send the message toward x. For the above example, the set of links whose failures will force G to choose the coordinate sequence [3,4,1,2] is {000-, 00-0, 010-, 01-0}. Proposition 5 can thus be extended and generalized as follows.

Lemma 1: Let u and w be two nodes in an injured Q_n with f faulty links and g faulty nodes where $H(u, w) = n$. Suppose $C = [c_1, c_2, \cdots, c_n]$ is the path chosen by G from u to w. Then, $f + g \geq V(C)$.

Recall that the obstructed node is the first node on a path that is aware that there is no optimal path to the destination node and a detour (i.e., nonoptimal path) has to be taken. To facilitate our presentation, define the *weight* of a set of dimensions to be their summation, i.e., $W(c_1, c_2, \cdots, c_m) = \sum_{i=1}^{m} c_i$. Then, we can derive the following theorem which characterizes the performance of G.

Theorem 1: Suppose u and w are respectively the source and destination nodes in a Q_n where $H(u, w) = n$. Then, the

number of faulty components (nodes and/or links) required for the simple coordinate sequence $C = [c_1, c_2, \cdots, c_m]$ to be the path chosen by G to an obstructed node located j hops away from w is $V(C) + W(c_1, \cdots, c_m) - \sum_{i=1}^{m} i + j$ where $m = n - j$.

To prove Theorem 1, we need the following lemma.

Lemma 2: Let $C = [c_1, c_2, \cdots, c_m]$ be a simple coordinate sequence. Then, $V(C) + V(C^R) = m(m-1)/2$.

Proof: There are $C_2^m = m(m-1)/2$ different ways to choose pairs (c_i, c_j), $i < j$, from $(c_1, c_2, \cdots, c_m)$. Since C is a simple coordinate sequence, either $c_i > c_j$ or $c_i < c_j$, meaning that each selection (c_i, c_j) will be counted for either $V(C)$ or $V(C^R)$, thus proving the lemma. Q.E.D.

Proof of Theorem 1: According to the operations of depth-first search routing, the lowest dimension among those dimensions not traversed before will be chosen first. Since $H(u, w) = n$, the selection of dimension c_1 to the first hop implies that $c_1 - 1$ faulty components have been encountered. Also, the selection of dimension c_2 to the second hop means a) there are another $c_2 - 1$ faulty components encountered if $c_2 < c_1$, or b) there are another $c_2 - 2$ faulty components encountered if $c_2 > c_1$. Following the same reasoning, we know that up to the obstructed node (i.e., the first m hops), the message must have encountered $\sum_{i=1}^{m}(c_i - 1) - V(C^R) = \sum_{i=1}^{m} c_i - m - m(m-1)/2 + V(C) = V(C) + \sum_{i=1}^{m} c_i - m(m+1)/2$ faulty components since $V(C^R) + V(C) = m(m-1)/2$ by Lemma 2. Also, additional j faulty components are required to block all the optimal paths from the obstructed node to w, thus proving the theorem. Q.E.D.

For example, suppose $C = [3,2]$ is the coordinate sequence of the path chosen by G from 0000 to the obstructed node 0110 when node 1111 is the destination. (Without loss of generality, one can consider node failures only.) Then, the required faulty nodes are 0001, 0010, 0101, 0111, and 1110. This agrees with that $V(C) + W(3,2) - (1+2) + 2 = 1 + 5 - 3 + 2 = 5$. It can be seen that Lemma 1 is a special case of Theorem 1, where the obstructed node is the destination node, i.e., $m = n$, $j = 0$ and $W(c_1, c_2, \cdots, c_n) = \sum_{i=1}^{n}$ whereas Proposition 5 is a special case of Lemma 1 in which both node and link failures are considered.

Let $S(n, m)$ be the set of combinations of m different numbers out of $\{1, 2, \cdots, n\}$ [27]. For example, $S(3,2) = \{(1,2), (1,3), (2,3)\}$. Clearly, $|S(n, m)| = C_m^n$ is the number of combinations of m objects out of n different objects. Let $I_n(k)$ denote the number of permutations of n numbers with exactly k inversions. Note that the value of $I_n(k)$ can be obtained from its generation function given in (1) later [25]. Then, from Theorem 1 we have the following important theorem.

Theorem 2: Suppose there are f faulty links in a Q_n, and a message is routed by G from node u to node w where $H(u, w) = n$. Let h_L be the Hamming distance between the obstructed node and the destination node. Then

$$P(h_L = j) = \frac{1}{C_f^L} \sum_{\sigma \in S(n,m)} \sum_{k=0}^{\min\left\{\frac{n(n-1)}{2}, f-j\right\}} \cdot I_m\left(k - W(\sigma) + \frac{m(m+1)}{2}\right) C_{f-j-k}^{L-n-k}$$

where $P(E)$ is the probability of event E, $L = n2^{n-1}$ and $m + j = n$.

Proof: Since faults may occur at any f links in the Q_n, there are C_f^L different configurations of faulty links. The problem of obtaining $P(h_L = j)$ is then reduced to that of counting the number of configurations which lead to the case of $h_L = j$. Note that the message traverses $m = n - j$ hops before it reaches the obstructed node, meaning that there are $|S(n, m)|$ possible locations of the obstructed node. Without loss of generality, one can assume that 0^n is the source node and 1^n is the destination node. Then, there is a one-to-one correspondence between each element in $S(n, m)$ and each possible location of the obstructed node.

Consider an obstructed node location x which is determined by an element $\sigma \in S(n, m)$. Let C be the coordinate sequence from node u to x. From Theorem 1, we know that the message has encountered $V(C) + W(\sigma) - m(m+1)/2$ faulty links before reaching x. Thus, the number of different paths from u to x while traversing the dimensions in σ and encountering k faulty links can be expressed as $I_m(k - W(\sigma) + m(m+1)/2)$ where $I_m(q)$ is the number of permutations of m numbers with q exactly inversions. For each given coordinate sequence to x, the locations of these k faulty links encountered before reaching x are determined. Moreover, there are additional j faulty links adjacent to x. Also, note that m links in the path from u to x are nonfaulty. Therefore, the number of different configurations for a given coordinate sequence or path to a certain obstructed node location x is $C_{f-j-k}^{L-j-k-m} = C_{f-j-k}^{L-k-n}$. Thus, this theorem follows. Q.E.D.

It is worth mentioning that determination of the probability of an optimal path routing can be viewed as a special case of Theorem 2 by setting the obstructed node to the destination node, i.e., the determination of $P(h_L = 0)$. More formally, we have the following corollary.

Corollary 2.1: Under Algorithm G, the probability for a message to be routed in an injured Q_n with f faulty links via an optimal path to a destination node which is n hops away can be expressed as

$$P(h_L = 0) = \frac{1}{C_f^{n2^{n-1}}} \sum_{k=0}^{\min\left\{\frac{n(n-1)}{2}, f\right\}} I_n(k) C_{f-k}^{n2^{n-1}-n-k}.$$

Proof: Since the destination node is viewed as the obstructed node which is 0 hop away from the destination node, one can substitute $j = 0$ and $m = n$ into the expression in Theorem 2. Note that the only element in $S(n, n)$ is $(1, 2, \cdots, n)$ and $W(1, 2, \cdots, n) = n(n+1)/2$, leading to this corollary. Q.E.D.

As mentioned earlier, the value of $I_n(k)$ can be obtained from its generation function [25],

$$G_n(z) = \sum_{j=0}^{n(n-1)/2} I_n(j) z^j = (1+z)(1+z+z^2)\cdots(1+z+z^2+\cdots+z^{n-1}), \quad (1)$$

because $I_n(k) = 1/k!\, d^k G_n(0)/dz^k$. It can be seen that

TABLE I
PERMUTATIONS WITH k INVERSIONS

n	$I_n(0)$	$I_n(1)$	$I_n(2)$	$I_n(3)$	$I_n(4)$	$I_n(5)$	$I_n(6)$	$I_n(7)$	$I_n(8)$	$I_n(9)$	$I_n(10)$	$I_n(11)$	$I_n(12)$	$I_n(13)$	$I_n(14)$	$I_n(15)$
3	1	2	2	1	0	0	0	0	0	0	0	0	0	0	0	0
4	1	3	5	6	5	3	1	1	0	0	0	0	0	0	0	0
5	1	4	9	15	20	22	20	15	9	4	1	0	0	0	0	0
6	1	5	14	29	49	71	90	101	101	90	71	49	29	14	5	1

TABLE II-A
$P(h_L = 0)$, PROBABILITIES OF OPTIMAL PATH ROUTING BETWEEN TWO NODES WITH HAMMING DISTANCE n IN A Q_n WITH f FAULTY LINKS

n \ f	1	2	3	4	5	6
3	.916667	.818182	.704545	.577778	.443182	.309524
4	.968750	.935484	.900202	.862903	.823599	.782318
5	.987500	.974684	.961551	.948101	.934335	.920253
6	.994792	.989529	.984211	.978839	.973413	.967932

$I_n(k) = 0$ if $k < 0$ or $k > n(n-1)/2$. The values of $I_n(k)$, for $3 \le n \le 6$ and $0 \le k \le n(n-1)/2$, are computed in Table I. The exact values for $P(h_L = j)$ can thus be determined by Theorem 2, and the probability of an optimal path routing between two nodes can be obtained by Corollary 2.1. From Corollary 2.1 and the data in Table I, we obtain Table II-A which shows the probabilities of optimal path routing between two nodes n hops away from each other in a Q_n with f faulty links. It can be seen that in the presence of link failures, Algorithm G can route a message to the destination via an optimal path with a very high probability.

Notice, however, that when n is large, the determination of $I_n(k)$, although it is feasible, may require excessive computation. To obtain more insights into the performance of G as well as reduce the computation required to obtain $P(h_L = j)$, an upper bound of $P(h_L = j)$ can be derived in closed form as follows.

Lemma 3: Suppose there are f faulty links in an injured Q_n, and a message is routed by G from node u to w, where $H(u, w) = n$. Then, $P(h_L = j) \le C_{f-j}^{L-j}/C_f^L$ if $0 \le j \le \min\{n, f\}$ where $L = n2^{n-1}$ is the number of links in a Q_n.

Proof: As mentioned before, the problem of obtaining $P(h_L = j)$ is to count the number of configurations which lead to the case of $h_L = j$. When $h_L = j$, all the j links toward w of the obstructed node, say x, must be faulty. According to depth-first search, the location of the obstructed node is determined by those faulty links which are not within $SQ(x, w)$ where $SQ(x, w)$ is the smallest subcube that contains both x and w Since the j links of node x within $SQ(x, w)$ are faulty, there are C_{f-j}^{L-j} different distributions of the other $f - j$ faulty links. When these $f - j$ faulty links cause node y, instead of x, to be the obstructed node, we exchange the links (including faulty links) in $SQ(y, w)$ with those in $SQ(x, w)$, and obtain a configuration which leads to the case when the obstructed node is y and $h_L = j$. Notice that some of the C_{f-j}^{L-j} different distributions of faulty links may lead to the case $h_L > j$, meaning that the number of configurations leading to the case $h_L = j$ is less than or equal to C_{f-j}^{L-j}. Q.E.D.

In light of Lemma 3 and the fact that $\sum_{j=0}^{n} P(h_L = j) = 1$, a lower bound for the probability $P(h_L = 0)$ can be derived in closed form as below.

TABLE II-B
APPROXIMATION FOR $P(h_L = 0)$ BY (2)

n \ f	1	2	3	4	5	6
3	.916667	.818182	.700000	.557576	.386364	.181818
4	.968750	.935484	.900000	.862069	.821434	.777809
5	.987500	.974684	.961538	.948052	.934211	.920000
6	.994792	.989529	.984211	.978836	.973404	.967914

$$P(h_L = 0) \ge 1 - \sum_{j=1}^{\min\{n, f\}} \frac{C_{f-j}^{n2^{n-1}-j}}{C_f^{n2^{n-1}}}. \tag{2}$$

Numerical examples for (2) are given in Table II-B. From Table II-A and B, it can be observed that (2) very closely approximates the exact expression given in Corollary 2.1. Similarly to the case of faulty links, the performance of G can be analyzed with respect to faulty nodes as stated in the following theorem.

Theorem 3: Suppose there are g faulty nodes in an injured Q_n, and a message is routed by G from node u to node w where $H(u, w) = n$. Let h_N be the Hamming distance between the obstructed node and the destination node. Then, for $2 \le j \le \min\{g, n\}$, we have,

$$P(h_N = j) = \frac{1}{C_g^{2^n-2}} \sum_{\sigma \in S(n,m)} \sum_{k=0}^{\min\left\{\frac{n(n-1)}{2}, g-j\right\}} \cdot I_m\left(k - W(\sigma) + \frac{m(m+1)}{2}\right) C_{g-j-k}^{2^n-2-n-k}$$

where $m + j = n$.

Proof: Recall that the source and destination nodes are assumed to be nonfaulty. We, therefore, consider only $C_g^{2^n-2}$ different distributions of the g faulty nodes. Also, note that when a message reaches an obstructed node which is j hops away from the destination, the message must have traversed $n - j = m$ nonfaulty nodes (not including the source node), and is blocked by at least j faulty nodes at the obstructed node. The conditions (faulty or not) of these $m + j + 2 = n + 2$ (including the source and destination nodes) are then determined from a given coordinate sequence to the obstructed node. This theorem thus follows from the same reasoning as the one in the proof of Theorem 2. Q.E.D.

Also, the probability of an optimal path routing in the presence of g faulty nodes, i.e., $P(h_N = 0)$, can be obtained by Corollary 3.1 below.

Corollary 3.1: Under Algorithm G, the probability for a message to be routed in a Q_n with g faulty nodes via an optimal path to a destination node located n hops away is

$$P(h_N = 0) = \frac{1}{C_g^{2^n-2}} \sum_{k=0}^{\min\left\{\frac{n(n-1)}{2}, g\right\}} I_n(k) C_{g-k}^{2^n-1-n-k}.$$

Proof: Note that in this case the destination node is viewed as an obstructed node that is 0 hop away from the destination node. From a given coordinate sequence of an optimal path, the locations of $n + 1$ nonfaulty nodes (including

TABLE III-A
$P(h_N = 0)$, PROBABILITIES OF OPTIMAL PATH ROUTING BETWEEN TWO NODES WITH HAMMING DISTANCE n IN A Q_n WITH g FAULTY NODES

n \ g	1	2	3	4	5	6
3	1.000000	.933333	.750000	.400000	.000000	.000000
4	1.000000	.989011	.964286	.922078	.858142	.768565
5	1.000000	.997701	.992857	.985185	.974366	.960045
6	1.000000	.999471	.998387	.996720	.994438	.991512

the source and destination nodes) in the path are known. This corollary thus follows from Theorem 3. Q.E.D.

Using Table I and Corollary 3.1, we can obtain Table III-A which gives the probabilities of optimal path routing between two nodes n hops away from each other in an Q_n with g faulty nodes. It can be seen that $P(h_N = 0)$ is greater than $P(h_L = 0)$ when the number of faulty components $g = f$ is relatively small as compared with the number of nodes in a hypercube. This is due to the fact that a faulty node is viewed as all the links attached to the node are faulty, and thus, the message can find a faulty node sooner than a faulty link. This in turn means that a detour will be taken sooner in the presence of faulty nodes than in the presence of faulty links. This is the reason that $P(h_N = 0)$ is greater than $P(h_L = 0)$ when $g = f$ is small. However, when the number of faulty components $g = f$ is large, $P(h_L = 0)$ is greater than $P(h_N = 0)$. This also agrees with our intuition since there are much more links than nodes in a hypercube. Similarly to the case of faulty links, the associated bounds in the case of faulty nodes can be obtained from the following lemma.

Lemma 4: Suppose there are g faulty nodes in an injured Q_n and messages are to be routed from an arbitrary node u to another node w where $H(u, w) = n$. Then, $P(h_N = j) \le C_{g-j}^{N-2-j}/C_g^{N-2}$ if $2 \le j \le \min\{n, g\}$, and $P(h_N = j) = 0$ if $j = 1$ where $N = 2^n$ is the total number of nodes in a Q_n.

From Lemma 4 and the reasoning used in determining the lower bound of $P(h_L = 0)$, we can obtain a lower bound for the probability $P(h_N = 0)$ in closed form as:

$$P(h_N = 0) \ge 1 - \sum_{j=2}^{\min\{n,g\}} \frac{C_{g-j}^{2^n-2-j}}{C_g^{2^n-2}}. \tag{3}$$

Numerical examples for (3) are given in Table III-B. It can be seen from Tables III-A and B that (3) closely approximates the expression given in Corollary 3.1, and also that Algorithm *G* routes a message to the destination via an optimal path with a rather high probability in the presence of faulty nodes.

V. CONCLUSION

In this paper, we presented a routing scheme based on depth-first search. The knowledge on the number of inversions of a given permutation is used to analyze the performance of this routing scheme. The number of faulty links/nodes required for a coordinate sequence to become the coordinate sequence of a path toward a given obstructed node is determined. Probabilities for routing messages via optimal path to given obstructed node locations are determined, and their associated bounds are derived in closed form. By setting the obstructed node to the destination node, the probability of the optimal path routing between any two nodes can be obtained as a special case of our results. It is also shown that this routing scheme can route a message to its destination via an optimal path with a very high probability.

TABLE III-B
APPROXIMATION FOR $P(h_N = 0)$ BY (3)

n \ g	1	2	3	4	5	6
3	1.000000	.933333	.750000	.400000	.000000	.000000
4	1.000000	.989011	.964286	.922078	.857642	.765235
5	1.000000	.997701	.992857	.985185	.974359	.960002
6	1.000000	.999471	.998387	.996720	.994438	.991511

ACKNOWLEDGMENT

The authors would like to thank J.-C. J. Chen at IBM for her assistance in obtaining the data for Tables II and III.

REFERENCES

[1] Y. Saad and M. H. Schultz, "Topological properties of hypercubes," *IEEE Trans. Comput.*, vol. C-37, pp. 867–872, July 1988.
[2] F. Ercal, J. Ramanujam, and P. Sadayappan, "Task allocation onto a hypercube by recursive mincut bipartitioning," in *Proc. 3rd Conf. Hypercube Concurrent Comput. Appl.*, Jan. 19–20, 1988, pp. 210–221.
[3] M.-S. Chen and K. G. Shin, "Processor allocation in an n-cube multiprocessor using gray codes," *IEEE Trans. Comput.*, vol. C-36, pp. 1396–1407, Dec. 1987.
[4] ——, "On relaxed squashed embedding of graphs into a hypercube," *SIAM J. Comput.*, vol. 18, pp. 1226–1244, Dec. 1989.
[5] T. N. Mudge and T. S. Abdel-Rahman, "Vision algorithms for hypercube machines," *J. Parallel Distributed Comput.*, vol. 4, pp. 79–94, 1987.
[6] T. F. Chan and Y. Saad, "Multigrid algorithms on the hypercube multiprocessors," *IEEE Trans. Comput.*, vol. C-35, pp. 969–977, Nov. 1986.
[7] C. Aykanat, F. Ozguner, F. Ercal, and P. Sadayappan, "Iterative algorithms for solution of large sparse systems of linear equations on hypercubes," *IEEE Trans. Comput.*, vol. C-37, pp. 1554–1568, Dec. 1988.
[8] C. L. Seitz, "The cosmic cube," *Commun. Assoc. Comp. Mach.*, vol. 28, no. 1, pp. 22–33, Jan. 1985.
[9] NCUBE Corp., "NCUBE/ten: An overview," Beaverton, OR, Nov. 1985.
[10] Intel Corp., *Intel iPSC System Overview,* Jan. 1986.
[11] S. L. Johnson and C. T. Ho, "Optimum broadcasting and personalized communication in hypercubes," *IEEE Trans. Comput.*, vol. C-38, pp. 1249–1268, Sept. 1989.
[12] M.-S. Chen and K. G. Shin, "On hypercube fault-tolerant routing using global information," in *Proc. Fourth Conf. Hypercube Concurrent Comput. Appl.*, Mar. 6–8, 1989.
[13] C. K. Kim and D. A. Reed, "Adaptive packet routing in a hypercube," in *Proc. 3rd Conf. Hypercube Concurrent Comput. Appl.*, Jan. 19–20, 1988, pp. 625–630.
[14] J. G. Kuhl and S. M. Reddy, "Distributed fault tolerance for large multiprocessor systems," in *Proc. 7th Annu. Int. Symp. Comput. Architecture,* May 1980, pp. 23–30.
[15] T. C. Lee and J. P. Hayes, "Routing and broadcasting in faulty hypercube computers," in *Proc. 3rd Conf. Hypercube Concurrent Appl.*, Jan. 19–20, 1988, pp. 346–354.
[16] P. Ramanathan and K. G. Shin, "Reliable broadcast in hypercube multicomputers," *IEEE Trans. Comput.*, vol. C-37, pp. 1654–1656, Dec. 1988.
[17] J. R. Armstrong and F. G. Gray, "Fault diagnosis in a Boolean n-cube array of multiprocessors," *IEEE Trans. Comput.*, vol. C-30, pp. 587–590, Aug. 1981.
[18] M.-S. Chen and K. G. Shin, "Message routing in an injured hypercube," in *Proc. 3rd Conf. Hypercube Concurrent Comput. Appl.*, Jan. 19–20, 1988, pp. 312–317.
[19] J. M. Gordon and Q. F. Stout, "Hypercube message routing in the presence of faults," in *Proc. 3rd Conf. Hypercube Comput. Appl.*, Jan. 19–20, 1988, pp. 318–327.

[20] E. Chow, H. S. Madan, J. C. Peterson, D. Grunwald, and D. Reed, "Hyperswitch network for the hypercube computer," in *Proc. 15th Annu. Int. Symp. Comput. Architecture,* May 30–June 2, 1988, pp. 90–99.
[21] T. Y. Feng, "A survey of interconnection networks," *IEEE Comput.*, pp. 12–27, Dec. 1981.
[22] M.-S. Chen, K. G. Shin, and D. D. Kandlur, "Addressing, routing, and broadcasting in hexagonal mesh multiprocessors," *IEEE Trans. Comput.*, vol. C-39, pp. 10–18, Jan. 1990.
[23] F. Harary, *Graph Theory*. Reading, MA: Addison-Wesley, 1969.
[24] E. N. Gilbert, "Gray codes and paths on the n-cube," *Bell Syst. Tech. J.*, vol. 37, pp. 263–267, 1973.
[25] D. E. Knuth, *The Art of Computer Programming,* Vol. 3. Reading, MA: Addison-Wesley, 1973.
[26] A. V. Aho, J. E. Hopcroft, and J. D. Ullman, *The Design and Analysis of Computer Algorithms*. Reading, MA: Addison-Wesley, 1974.
[27] C. L. Liu, *Introduction of Combinatorial Mathematics*. New York: McGraw-Hill, 1968.

Ming-Syan Chen (S'87–M'88) was born in Taipei, Taiwan, Republic of China. He received the B.S. degree in electrical engineering from National Taiwan University in 1982, and the M.S. and Ph.D. degrees in computer, information, and control engineering from The University of Michigan, Ann Arbor, MI, in 1985 and 1988, respectively.

Since Fall 1988, he has been with I.B.M. Thomas J. Watson Research Center, Yorktown Heights, NY, where he is currently a research staff member in the Architecture Analysis and Design group. From 1982 to 1984, he served in the ROC military service as an electronics instructor. In Fall 1984, he went to The University of Michigan, Ann Arbor, MI, where he was a research assistant in Real Time Computing Laboratory of EECS department, and completed graduate studies in the area of routing and task allocation in multiprocessor interconnection networks. His research interests include parallel computer architectures, especially hypercubes, distributed database systems, multiprocessor interconnection networks, algorithms, graph and combinatorial theory, especially graph embedding, and enumerative combinatorics.

Dr. Chen is a member of the Association for Computing Machinery.

Kang G. Shin (S'75–M'78–SM'83) received the B.S. degree in electronics engineering from Seoul National University, Seoul, Korea, in 1970, and both the M.S. and Ph.D. degrees in electrical engineering from Cornell University, Ithaca, NY, in 1976 and 1978, respectively.

He is currently a Professor of Electrical Engineering and Computer Science, The University of Michigan, Ann Arbor, which he joined in 1982. From 1970 to 1972 he served in the Korean Army as an ROTC officer and from 1972 to 1974 he was on the research staff of the Korea Institute of Science and Technology, Seoul, Korea, working on the design of VHF/UHF communication systems. From 1978 to 1982 he was an Assistant Professor at Rensselaer Polytechnic Institute, Troy, NY. He was also a visitor at the U.S. Airforce Flight Dynamics Laboratory in Summer 1979 and at Bell Laboratories, Holmdel, NJ in Summer 1980. During the 1988–1989 academic year, he was a Visiting Professor in the CS Division, Electrical Engineering and Computer Science, UC Berkeley and at International Computer Science Institute, Berkeley, CA. He has authored/coauthored over 160 technical papers in the areas of fault-tolerant computing, distributed real-time computing, computer architecture, and robotics and automation. In 1985, he founded the Real-Time Computing Laboratory, where he and his colleagues are currently building a 19-node hexagonal mesh multicomputer, called HARTS, to validate various architectures and analytic results in the area of distributed real-time computing.

Dr. Shin received the Outstanding Paper Award in 1987 from the IEEE TRANSACTIONS ON AUTOMATIC CONTROL for a paper on robot trajectory planning. In 1989, he also received the Research Excellence Award from The University of Michigan. He was the Program Chairman of the 1986 IEEE Real-Time Systems Symposium (RTSS), the General Chairman of the 1987 RTSS and the Guest Editor of the 1987 August special issue of IEEE TRANSACTIONS ON COMPUTERS on real-time systems. He is currently a Distinguished Visitor of the Computer Society of the IEEE and an Area Editor of *International Journal of Time-Critical Computing Systems*.