Multicasting in Injured Hypercubes Using Limited Global Information

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

Abstract

We study the multicast problem in injured hypercubes with faulty nodes by assuming that each node in a cube has limited global information, which is captured by the safety level associated with the node. Basically, the safety level is an approximate measure of the distribution of faulty nodes in the neighborhood. We first propose a safety-level based multicasting scheme which guarantees time optimality. Then an address-sum based multicasting scheme is studied which utilizes both the distribution of destination nodes and safety levels of neighboring nodes. Simulation results show that the traffic generated in both schemes is very close to the optimal solution. Time optimality is guaranteed when the source is safe and an additional time step is required when the source is unsafe in an n-cube with up to $n - 1$ node faults.

1 Introduction

Message passing techniques are commonly used in hypercubes [7] for interprocessor communications. Usually, a message must go through several intermediate nodes to reach its destination. Thus, the routing scheme used [6] becomes very important for interprocessor communication. In general, there are three types of communications: *one-to-one*, *one-to-many*, and *one-to-all* [3]. *Time steps* and *traffic steps* [3] are the main criteria used to measure the performance of communication at the system level. The maximum number of links the message traverses to reach one of the destinations is defined as time steps, and the total number of links the message traverses to reach all destinations is measured in traffic steps.

It has been shown [5] that the problem of finding a time and traffic optimal solution for multicasting in hypercubes is NP-hard. A heuristic multicast algorithm proposed by Lan, Esfahanian, and Ni [3] achieves time optimality and traffic sub-optimality in a nonfaulty hypercube. This algorithm starts at the source node and is executed at a set of intermediate nodes until all the destination nodes are reached. Each intermediate node receives a copy of the message and a set of destination node addresses. This set is partitioned at the intermediate node and each subset in the partition is sent to an appropriate neighbor. The partition is performed in such a way that the set of destination nodes is sent to a few neighbors (to achieve maximum sharing of common paths) along shortest paths from the source node to each destination.

Several fault-tolerant multicasting schemes have been proposed which can be classified by the amount of network information used at each node. In local-information-based multicasting as proposed by Lan [1], each node knows only the status of its adjacent links and nodes. Simplicity is the main advantage of this scheme, although an uncontrollable number of additional time steps may occur in the worst case. A time-optimal multicasting has recently been proposed [2] which can be categorized as local-information-based multicasting. However, it is based on a restricted fault model - each node in the cube has at most one faulty neighbor. Global-information-based multicasting [8] assumes that each node knows distribution of faults in the network. This scheme guarantees time optimality. However, it requires a complex process that collects global information. Limited-global-information-based multicasting is a compromise of the above two schemes. It can obtain a time optimal or suboptimal solution while requiring a relatively cheap process that collects and maintains limited global information. Liang, Bhattacharya, and Tsai [4] recently proposed a multicasting scheme where each node knows the status of all the links within two hops away (a particular form of

1063-6374/93 $03.00 © 1993 IEEE

limited global information). This scheme can tolerate up to $n-1$ faulty links and the number of additional time steps required in the worst case is $2n$. Although this scheme can be easily extended to tolerate node failures, it has two flaws: first, it cannot guarantee time optimality; second, it has to maintain at each node a relatively large table which contains the status of components within two hops from the node.

In this paper, we propose a multicasting method based on limited global information. The limited global information is captured by the *safety level* [9] which is an integer associated with each node. Basically, the safety level associated with a node is an approximate measure of the number of faulty nodes in the neighborhood. An efficient algorithm for calculating and updating the safety level of each node in a cube was proposed in [9]. The proposed multicasting method uses the safety level information and the distribution of destination nodes. It can tolerate $n-1$ faulty nodes in an n-cube while still guaranteeing time optimality and traffic suboptimality. Several variations are studied and presented in order of complexity of the approach.

This paper is organized as follows: Section 2 defines some notation and preliminaries. The basic strategy used in the proposed scheme is also discussed. Section 3 provides a safety-level based multicasting scheme, and one variation of this scheme is studied. Section 4 provides an address-sum based multicasting scheme. Section 5 studies the performance of the proposed schemes and compares the results with ideal cases, i.e. cases where both traffic and time optimality are achieved. Section 6 presents some discussions and conclusions.

2 Preliminaries

The n-dimensional hypercube (or n-cube), Q_n, is a graph having 2^n nodes labeled from 0 to 2^n-1. Two nodes are joined by an edge if their addresses, as binary integers, differ in exactly one bit. More specifically, every node u has address $u(n)u(n-1)\cdots u(1)$ with $u(i) \in \{0,1\}$, $1 \leq i \leq n$, and $u(i)$ is called the i^{th} bit (also called the i^{th} dimension) of the address. For convenience, we use u to represent a node as well as its associated address. Node u^i is the neighbor of u along dimension i. An *injured hypercube* is a connected faulty hypercube.

Let $G(V,E)$ be a graph that corresponds to a hypercube structure. $D = \{u_1, u_2, ..., u_m\} \subseteq V(G)$ is a *multicasting set*. The multicasting problem can be modeled as the problem of finding a subtree $T(V,E)$ of $G(V,E)$, such that (a) $D \subseteq V(T)$, (b) $d_T(s,u_i) = d_G(s,u_i)$, where s is the source node and d_T and d_G are distances in graphs T and G, respectively, and (c) $|E(T)|$ is as small as possible. Our objective here is to find a multicasting scheme that guarantees time optimality and traffic suboptimality.

In our approach, the concept of limited global information is captured by assigning a *safety level* [9] to each node in an injured hypercube. The safety level associated with a node is an approximated measure of the number of faulty nodes in the neighborhood. Let $st(u) = k$ be the safety status of node u, where k is referred as the level of safety, and u is called *k-safe*, $1 \leq k \leq n$. A faulty node is 1-safe which corresponds to the lowest level of safety, while an n-safe node (also referred as a *safe node*) corresponds to the highest level of safety. A node with k-safe status is called *unsafe* if $k \neq n$.

Definition 1 [9]: *Let $(st_0, st_1, st_2, ..., st_{n-1})$, $1 \leq st_i \leq n$, be the non-ascending safety status sequence of node u's n neighboring nodes in an n-cube, such that $st_i \geq st_{i+1}$, $0 \leq i \leq n-1$. The safety status of node a is defined as: if $(st_0, st_1, ..., st_{n-2}, st_{n-1}) \geq (n-1, n-2, ..., 1, 0)$[1] then $st(u) = n$ else if $(st_{n-k}, st_{n-(k-1)}, ..., st_{n-2}, st_{n-1}) \geq (k-1, k-2, ..., 1, 0) \wedge (st_{n-(k+1)} = k-1)$ then $st(u) = k$.*

The following GLOBAL_STATUS (GS) algorithm determines each node's status in Δ iterative rounds through message exchange among neighboring nodes. The algorithm can be used when (1) the system initializes, all nonfaulty nodes initially are assigned n-safe status, (2) a node detects a new faulty node, or (3) a faulty node is recovered. In cases (2) and (3) each node uses its old status as its initial status before applying the algorithm.

```
Algorithm GLOBAL_STATUS (GS)
/* Each node $u_i$ keeps a non-ascending status sequence $st$
= $(st_0, st_1, st_2, ..., st_{n-1})$ of neighboring nodes. Initially all
nonfaulty nodes are $n$-safe and $round = 0$. */
begin
 while round < $\Delta$
  ( parbegin
     NODE_STATUS($u_i$), $0 \leq i \leq 2^n - 1$
    parend;
    /* Each node determines its status. One calculation of
    the above statement is considered to be one round. */
    round := round +1 )
end.
Procedure NODE_STATUS($u_i$) (NS)
begin
```

[1] $seq_1 \geq seq_2$ if and only if each element in seq_1 is greater or equal to the corresponding element in seq_2.

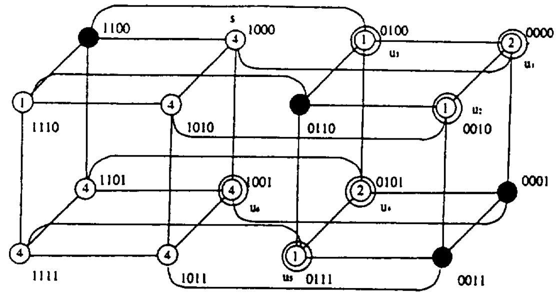

Figure 1: An injured Q_4 with four faulty nodes

```
determine the non-ascending status sequence
if $(st_0, st_1, ..., st_{n-2}, st_{n-1}) \geq (n-1, n-2, ..., 1, 0)$
    then mark $u_i$ as $n$-safe (or safe);
if $(st_{n-k}, st_{n-(k-1)}, ..., st_{n-2}, st_{n-1})$
    $\geq (k-1, k-2, ..., 1, 0) \wedge (st_{n-(k+1)} = k-1)$
    then mark $u_i$ as $k$-safe;
end.
```

In the GS algorithm, Δ represents the lower bound on the number of rounds required to ensure that all the nodes in the cube reach a *stable state*[2]. It has been shown in [9] that $\Delta = n-2$ in any faulty n-cube (including a *disconnected cube*) when $n > 2$ and $\Delta = 1$ in any faulty 2-cube(including a *disconnected cube*).

The GS algorithm uses *barrier synchronization* (the **while** statement together with the **parbegin** and **parend** statement) to ensure that each node cannot start round $i+1$ before all the other nodes finish round i, although it is acceptable for the status function to be a values resulting from any prior or future iteration of neighbor processes. That is, each NS in the GS algorithm can truly execute in parallel without synchronization. We assume that each faulty node is fail-stop and it will not participate in NODE_STATUS.

Figure 1 shows an injured Q_4 with four faulty nodes (represented as black nodes): 1100, 0110, 0011, and 0001. The number in each node indicates the safety level of that node. Notice that whenever there is more than one faulty neighbor of a node, then this node is 1-safe. Each nonfaulty node is either a safe node (4-safe) or an unsafe node that has a safe neighbor. Actually, it has been shown in [9] that each unsafe node has a safe neighbor in any injured hypercube with up to $n-1$ faulty nodes. The key issue in a multicast is how each intermediate node u (including the source node s) should forward a set of destination nodes $\{u_1, u_2, ..., u_m\}$ to its appropriate neighboring nodes. We represent this destination set in terms of their relative addresses with respect to node u, $R = \{r_i\}$, where $r_i = u \oplus u_i, 1 \leq i \leq m$. The ***address summation*** $as = \sum_{r_i \in R} r_i$, represents the distribution of destination nodes along different dimensions. $|r_i| = \sum_{1 \leq j \leq n} r_i(j)$ represents the distance between nodes u and u_i. For example, if a set of destination nodes $\{u_1, u_2, u_3\} = \{0101, 1001, 0000\}$ and node $u = 1010$, then $R = \{r_1, r_2, r_3\} = \{1111, 0111, 1010\}$ with $|r_1| = 4$, $|r_2| = 3$, and $|r_3| = 2$ and $as = 2232$.

[2]A node status is stable at the k^{th} round if it remains unchanged after round k.

To ensure time optimality we use the following simple strategy [3]:

Strategy: *When bit j of the relative address of destination node u_i equals one, u_i should be sent to u's neighbor, u^j, along dimension j.*

When r_i of destination node u_i has more than one 1 value at different bits (dimensions), then u_i could be forwarded along any of these dimensions. In this case a *conflict* arises. To resolve the conflict, a priority order is determined among n dimensions. The formation of this priority order determines the result of a multicasting. Three formations of priority order are studied in this paper. In the safety-level-based multicasting (SLBM), the priority of a dimension is determined *a priori* based on the safety level of the neighbor along that dimension. Two approaches can be used when two or more than two neighbors have the same highest safety level. In the SLBM, a priority order among those dimensions along which neighbors have the same priority order is randomly selected. In the modified SLBM (MSLBM), the priority is based on the locations of nodes in the destination set. More specifically, the priority of a dimension is based on the corresponding bit value in the address summation of the destinations. The next highest priority dimension is determined using the same approach but is based on the updated destination set, i.e. the one after removing those nodes to be forwarded to dimensions in higher priorities. In the address-sum based multicasting (ASBM) approach, the dimension priority depends primarily on bit values in the address summation. A dimension has the highest priority if the corresponding neighbor can carry the maximum possible nodes. To ensure time optimality, as will be discussed later, only those nodes that are no more than k steps away from the current node will be selected, where k is the safety level of the neighbor.

The proposed schemes can also be applied to any injured n-cubes as long as the source node is safe. In case the source node is unsafe, but there exists a safe neighbor, the proposed scheme is still applicable. The feasibility of the scheme can be easily checked at the source node by examining its safety status together with the safety status of its neighbors (if necessary). Actually, this is another major advantage of this approach over existing solutions where restricted fault models are used.

3 Safety-Level Based Multicasting

The SLBM algorithm works as follows: assume that u is an intermediate node in an injured Q_n. $st = (st_0, st_1, ..., st_{n-1})$ is u's status sequence of neighboring nodes in descending order and $ds = (d_0, d_1, ..., d_{n-1})$ is the corresponding neighbor dimension sequence. Node u forwards destination nodes to its appropriate neighboring nodes. Conflicts are resolved by defining a priority order on the dimension set. In SLBM, the priority order is specified by ds, i.e. the order is based on the safety level of neighbors. We use u to represent the current node address, R for destination set, M for message, and R_i for a set of destination node addresses to be sent to one of the u's neighbors.

```
Algorithm SLBM(u, R, M)
( $R_i \leftarrow \phi$, $0 \le i < n$;
  for $c = 0$ to $n-1$ do
   ( for each $r_i \in R$ do
      if $r_i(d_c) = 1$ then
       ( $R_c \leftarrow R_c \cup r_i^{d_c}$;
         $R \leftarrow R - \{r_i\}$
       );
     if $R_c \ne \phi$ then send $(R_c, M)$ to node $u^{d_c}$
   );
   /* send $R_c$ together with $M$ to the neighbor
   along dimension $d_c$ */
   if $R \ne \phi$ then node $u$ keeps a copy of $M$
   /* when the current node is a destination node,
   a copy of $M$ is kept */
)
```

The MSLBM is a refinement of SLBM. When there are two or more neighbors that have the same safety level, instead of randomly ordering these neighbors as is done in SLBM, MSLBM uses address summation to determine the order. When two or more neighbors have the same highest safety level and the same number of destination nodes that they can carry, the priority among these neighbors is randomly defined.

```
Algorithm MSLBM(u, R, D)
( $R_i \leftarrow \phi$, $0 \le i < n$;
  for $c = 0$ to $n-1$ do
   ( if there exists $l \ne c$ such that $st_c = st_{c+1} = ...$
      $= st_{l-1} = st_l > st_{l+1}$ then
      ( calculate $as = \sum_{r_i \in R} r_i$; /* address summation */
        select $st_j \in \{st_c, st_{c+1}, ..., st_{l-1}, st_l\}$ such that
        $as(d_j) = max\{as(d_i) | c \le i \le l\}$;
        /* select dimension $d_j$ such that the maximum
        possible destination nodes can be passed to $u^{d_j}$ */
        exchange $d_c$ and $d_j$ in $ds$
        /* rearrange $st$ and $ds$ sequences /*
      );
  for each $r_i \in R$ do
   if $r_i(d_c) = 1$ then
   ( $R_c \leftarrow R_c \cup r_i^{d_c}$;
     $R \leftarrow R - \{r_i\}$
   );
  if $R_c \ne \phi$ then send $(R_c, M)$ to node $u^{d_c}$
 );
 if $R \ne \phi$ then node $u$ keeps a copy of $M$
)
```

The following two lemmas are used to prove that both SLBM and MSLBM achieve time optimality when the source node s is safe.

Lemma 1 *If the source node is safe and node u is k-safe, then any destination node received at node u is no more than k Hamming distance away from u.*

Proof. Let us classify all the nodes in the cube based on their distances from the source node and we prove this lemma based on induction on the distance between the source s and node u. Clearly there are n groups of nodes in an n-cube. The source node s is a node that is zero distance form the source node. Since all the destination nodes in the n-cube are no more than n Hamming distance away from s, this lemma automatically holds when u is s and is n-safe. Assume this lemma holds for all the nodes which are distances l away from the source node and we arbitrarily select a node u which is k-safe from these nodes. That is, all the destination nodes received at u have at most k 1-bits (bits that take the value 1) in their addresses. Suppose $(st_0, st_1, ..., st_{n-1})$ is u's status sequence of neighboring nodes in a non-ascending order and $(d_0, d_1, ..., d_{n-1})$ is the corresponding neighbor dimension. Based on the safety definition in Section 2, $(st_0, st_1, ..., st_{n-k}, st_{n-(k-1)}, st_{n-(k-2)}, ..., st_{n-2}, st_{n-1}) \ge (k-1, k-1, ..., k-1, k-2, k-3, ..., 1, 0)$. Clearly, the theorem holds for all the neighbors along dimensions d_0, d_1, ..., d_{n-k}, since their safety level is at least $k-1$. Note that the number of 1-bits in destination addresses will be reduced by one when these destination nodes are forwarded to appropriate neighbors. The safety level of the neighbors along dimension $d_{n-(k-i)}$, where $1 \le i \le k-1$, is at least $st_{n-(k-i)} = k-i-1$. Based on SLBM and MSLBM, a destination node r is selected to be forwarded to the neighbor along $d_{n-(k-i)}$ only when all the bit values of r at dimensions $d_0, d_1, ..., d_{n-k}, d_{n-(k-1)}, ..., d_{n-(k-(i-1))}$ are zero. That is, there are at least $n-(k-(i-1))+1 = n-(k-i)$ 0-bits in r, i.e. at most $n-(n-(k-i)) = k-i$ 1-bits in r. The 1-bit at $d_{n-(k-i)}$ will be set to zero when r is forwarded to the neighbor along that dimension. Therefore, there are at most $k-i-1$ 1-bits in r which are less than or equal to the safety level of the neighbor along that dimension. This lemma holds for all the neighbors of u, including those which are distances $l+1$ away from the source. Since the selection of u is random, this result apply to all the nodes which are distances $l+1$ from the source. □

Lemma 2: *Every destination node address (except node u) received at node u is sent together with the multicast message to one of the neighbors along one of the Hamming distance paths between u and that destination node.*

Proof: A node r in the destination set received at node u will be sent to node u^d only when $r(d) = 1$. Node u^d must be a node in one of the Hamming distance paths from node u and d. By induction, we can prove that message will reach each destination from the source node through one of the Hamming distance paths between the source and the destination. Next we prove that all destination addresses (except the one with zero address which represents the current node) will be passed to one of the nonfaulty neighbors along Hamming distance paths from the source to each destination or to a faulty neighbor only if the destination is the faulty neighbor itself. Suppose node u is k-safe, $k > 1$. Based on the safety definition, there is at most one faulty neighbor (along dimension d). For each destination r (excluding u and faulty neighbors) there is at least one bit other than d that has value 1. Since dimension d has the lowest priority, r will not be forwarded along this dimension. If the d^{th} dimension is the only one bit that has value one in r, then the destination node is a faulty neighbor. In this case, r is either sent along dimension d or it is discarded. The only destination node (if one exists) that is not selected is the node u itself. In this case, a copy of the multicast message is kept at node u. Suppose node u is 1-safe, based on the definition of the safety concept, there are 2 to $n-1$ faulty neighbors. Based on Lemma 1, only those destinations which are no more than 1 Hamming distance away from u are forwarded to u, i.e. all the destinations are either neighbors of u or u itself. This lemma automatically holds. □

Theorem 1: *A multicast generated by SBLM or MSBLM is guaranteed to be time optimal if the source node is safe in any injured n-cube. When the source node is unsafe, one extra step is required in a n-cube with up to $n-1$ faulty nodes.*

Table 1: The *st* and *ds* for each node of Figure 2

node	*st*	*ds*	*node*	*st*	*ds*
0000	(4,2,2,1)	(4,3,2,1)	0001	-	-
0010	(4,3,1,1)	(4,2,3,1)	0011	-	-
0100	(3,3,1,1)	(3,1,4,2)	0101	(4,2,2,1)	(4,2,1,3)
0110	-	-	0111	(4,3,1,1)	(4,2,3,1)
1000	(4,4,3,1)	(2,1,4,3)	1001	(4,4,4,1)	(3,2,1,4)
1010	(4,4,2,2)	(2,1,4,3)	1011	(4,4,4,1)	(3,2,1,4)
1100	-	-	1101	(4,4,3,1)	(3,2,4,1)
1110	(4,4,1,1)	(3,2,4,1)	1111	(4,4,2,2)	(3,2,4,1)

This theorem can be directly derived from Lemmas 1 and 2 and the discussion in Section 2.

Consider the injured Q_4 shown in Figure 1 with four faulty nodes: 1100, 0110, 0011, and 0001. The safety level associated with each node in this Q_4 is shown in Figure 1. Table 1 shows the descending status sequence (st) and the corresponding neighbor dimension sequence (ds) associated with each node in the injured Q_4 of Figure 1. We use "–" to present a don't-care symbol for st and ds associated with faulty nodes. In Figure 1, each destination is represented by a double-circled node, and each black node is a faulty node.

Suppose the source node is the safe node 1000, and the multicasting set is $u = \{u_1, u_2, u_3, u_4, u_5, u_6\} = \{0000, 0010, 0100, 0101, 0111, 1001\}$. The set of relative addresses between the source and destinations nodes is $R = \{r_1, r_2, r_3, r_4, r_5, r_6\} = \{1000, 1010, 1100, 1101, 1111, 0001\}$. Therefore, the address summation is $as = 5323$. In the SLBM algorithm, which is a safe-level based multicasting, we assume that the algorithm follows the non-ascending status sequence shown in Table 1. At the source node 1000 the non-ascending status sequence and the corresponding neighbor dimension sequence are $(4,4,3,1)$ and $(2,1,4,3)$, respectively.

The SLBM algorithm uses only the neighbor dimension sequence (ds) in determining the priority among the neighboring nodes. In this case, dimension 2 has the highest priority, followed by dimension 1 and then dimension 4. Dimension 3 has the lowest priority. Since the 2^{nd} bit value is one in r_2 and r_5, u_2 and u_5 are passed to node 1010, the neighbor of 1000 along dimension 2. Among the remaining nodes in R, r_4 and r_6 have value one in the 1^{st} bit. Both u_4 and u_6 are passed to node 1001. Since the 4^{th} bit value is one in the remaining r_1 and r_3, nodes u_1 and u_3 are passed to 1000's neighbor along dimension 4. No destination node are passed to the neighbor along dimension 3. The procedure is applied recursively to 1000's neigh-

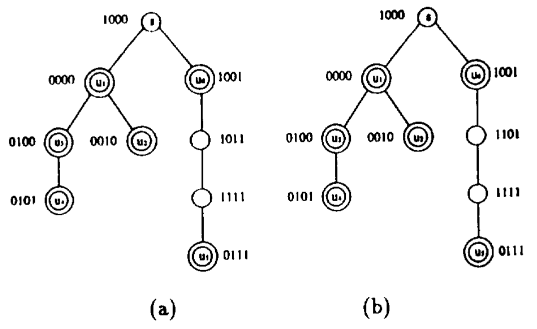

Figure 2: The multicasting tree generated using (a) the SLBM algorithm and (b)the MSLBM algorithm

bors that receive destination nodes, and a multicasting tree is formed as is shown in Figure 2 (a). The depth of this tree is the number of time steps used and the number of edges in the tree is the number of traffic steps used. In this case, the time steps are 4 and the traffic steps are 10.

The MSLBM algorithm also uses the neighbor dimension sequence (ds) in determining the priority. However, when two or more two neighbors have the same highest safety level, the address summation (as) on the remaining destination nodes is used to break the tie. In the example shown in Figure 1, two neighbors of the source node 1000 along dimensions 1 and 2 have the same safety level. Based on $as = 5323$, the neighbor along dimension 2 (node 1010) can carry 2 (the 2^{nd} bit value of as) destination nodes, and the neighbor along dimension 1 (node 1001) can carry 3 (the 1^{st} value of as) destination nodes. Therefore, node 1001 has a higher priority over node 1010. As a result, u_4, u_5, and u_6 are passed to 1001, u_2 is passed to node 1010, the neighbor of 1000 along dimension 2. Nodes u_1 and u_3 are passed to 1000's neighbor along dimension 4. Node 1001 in Figure 1 receives three destination nodes u_4, u_5, and u_6 (node 1001 itself) from node 1000. Two of its neighbors, 1101 and 1011, tie in terms of their safety levels. However, node 1101 can carry all the remaining destination nodes while node 1011 can carry only 0111. Therefore, node 1101 has a higher priority than 1011. At node 1101, u_4 is directly sent to 0101, and u_5 is sent to 0111 via 1111 (node 1111 has a higher safety level than 0101). Clearly, u_5 can be forwarded to 0111 via $0101 = u_4$ to save one traffic step. This deficiency is overcome in the ASLM approach discussed in the next section. Figure 2 (b) shows the resultant multicasting tree. This tree uses 4 time steps and 9 traffic steps.

4 Address-Sum Based Multicasting

In the ASBM algorithm, the dimension priority is only primarily dependent on the address summation of destination nodes. That is, the dimension with the maximum bit value in the address summation has the highest priority. All the destination addresses with value one at that dimension will be sent to the neighbor along that dimension. However, in order to avoid forwarding too many destinations to an unsafe or faulty neighbor, we forward destination addresses to a k-safe neighbor only when the corresponding destination nodes are less or equal to k Hamming distance away from this k-safe neighbor. Again, we use u to represent an intermediate node (when $u = s$, we assume that u is a safe node). Let st_i, $0 \leq i \leq n-1$, the safety status of the neighbor of u along dimension d_i and dd be the dimension set in an n-cube. Each node has a copy of dd which is initialized $\{1, 2, ..., n\}$ at each multicast. When there are two or more neighbors that can carry the same maximum number of destination nodes, the selection is random, although we can easily extend the ASBM algorithm in such a way that the selection among these neighbors is based on their safety levels.

```
Algorithm ASBM($u, R, M$)
(   $R_i \leftarrow \phi$, $0 \leq i \leq n-1$;
    while $R \neq \phi \wedge R \neq \{0\}$ do
    /* continue when $R$ contains at least one
    destination other than the current node */
    ( calculate $as = \sum_{r_i \in R} r_i$;
      select $d_c \in ds$ such that $as(d_c) = max\{as(d_i) | d_i \in dd\}$;
      for each $r_i \in R$ do
          if $r_i(d_c) = 1$ and $|r_i| \leq st_c$ then
          ( $R_c \leftarrow R_c \cup r_i^{d_c}$;
            $R \leftarrow R - \{r_i\}$;
            $dd \leftarrow dd - \{d_c\}$
          );
      if $R_c \neq \phi$ then send $(R_c, M)$ to $u^{d_c}$
    );
    if $R \neq \phi$ then node $u$ keeps a copy of $M$
)
```

For the example shown in Figure 1, $cs = 5323$, therefore dimension 4 has the highest priority. Potentially, node 0000, 1000's neighbor along dimension 4, can carry five destination nodes. However, only those nodes which are no more than two steps (two is the safety level of node 0000) away from 0000 will be passed to it. In this case, only nodes u_1, u_2, u_3, and u_4 are passed to node 0000. The next highest dimension is determined based on the updated address summation, $as = -112$, on the remaining destination nodes u_5 and u_6. (Since dimension 4 has been selected, we

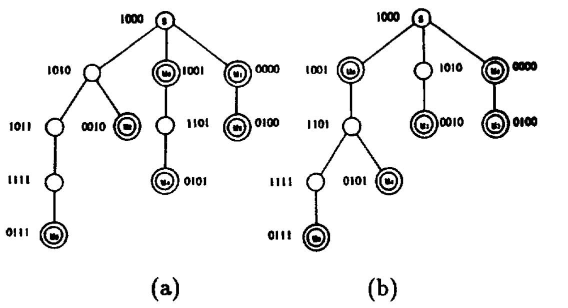

Figure 3: Two multicasting trees generated using the ASBM algorithm

use the don't-care symbol "$-$" at the 4^{th} bit.) Therefore, dimension 1 is selected. Since the safety level of node 1001, the neighbor along dimension 1, is four, the remaining destination nodes are passed to it. At node 1001 the $as = -11-$ and neighbors 1011 and 1101 of 1001 have the same safety level and the same bit value in as. Therefore, u_5 can be forwarded either through node 1011 or node 1101 to the destination. The two resulting multicasting trees are shown in Figure 3 (a) and (b).

Lemma 1 is straightforward based on the definition of algorithm ASBM. Lemma 2 can be proved based on the following fact that can be derived following the proof of Lemma 1: Suppose an intermediate node u is k-safe, then all the destinations it receives are no more that k distance away. Assume an arbitrarily selected destination is $l(< k)$ distance away from u, i.e. there are l 1-bits in its relative address. Based on the safety definition (Definition 1), there are no more than $l-1$ neighbors of u that have safety levels lower than l. Therefore, each destination associated with node u will be selected to be sent to one of u's neighbors along a Hamming distance path from u to that destination node. Based on the above two lemmas, we can easily show that Theorem 1 is still valid.

5 Performance

As has been shown in the previous sections, all three multicasting schemes, SLBM, MSLBN, and ASBM, guarantee time optimality when the source node is safe. One extra time step is required if the source node is nonsafe.

The multicasting simulation is conducted on 4-cubes and 5-cubes with various numbers of faulty nodes and destination nodes. To simplify the simulation, we assume that all the destination nodes are fault free. Both faulty nodes and destination nodes are randomly selected from the 2^k nodes of the k-cube. We use d to represent the number of destination nodes and f to represent the number of faulty nodes. Figure 4 shows the average traffic steps generated from these three algorithms in 5-cubes with 4 faulty nodes. The optimal result and low bound are also shown in this figure. The lower bound of traffic steps is defined as the number of destination nodes. That is, there are no wasted steps in a multicasting. The lower bound is unachievable for most cases. Figure 5 shows the percentage of saved traffic using MSLBN and ASBM compared with the one obtained by using SLBM in 5-cubes. Figure 6 shows the relationship between the traffic steps versus the number of faulty nodes with various numbers of destination nodes in 4-cubes.

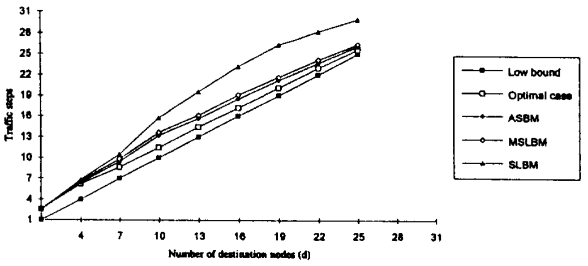

Figure 4: Traffic comparison of three algorithms

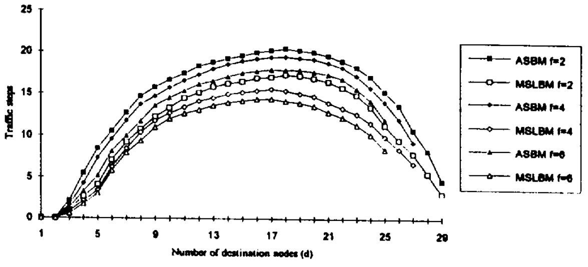

Figure 5: Percentage of saved traffic

We make the following observations from the simulation results shown in Figures 6, 7, and 8.

(1) The average traffic steps achieved by using MSLBN and ASBM are close to the optimal case. In general, ASBM outperforms MSLBN. The SLBM algorithm does not produce results which are close to the optimal case as do the other two algorithms.

(2) The traffic steps generated by using ASBM and MSLBM increase when the number of faulty nodes increases until $f = n$. When $f > n$, the number of

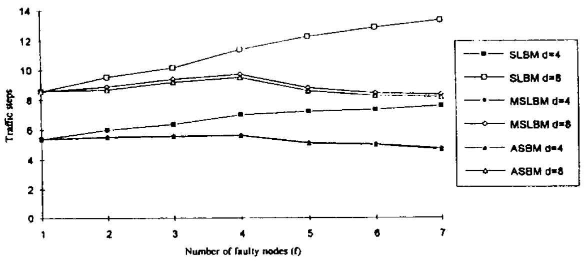

Figure 6: Traffic steps versus number of faulty nodes in a 5-cube

traffic steps decreases as f increases.

When the number of faulty nodes increases, the possibility of sharing common paths becomes smaller. However, as the number of faulty nodes increases, the number of (nonoptimal) paths to each destination is also reduced. Initially, the former factor outweights the later one. When the number of faulty nodes (f) reaches a certain number, which is close to n, the latter factor outweights the former. In the extreme case, when all the paths except the optimal one are blocked, no traffic overhead is generated.

(3) Compared to the SLBM approach, the percentages of saved traffic generated by using the MSLBM and ASBM approaches reaches the maximum when the number of destination nodes is about half of the total number of nodes (i.e., $d = n/2$). When the number of destination nodes is more than half of the total number of nodes, the percentage of saved traffic decreases.

This occurs because there are not many nonoptimal paths to each destination when there are few destinations or when nearly all the nodes are destinations. Naturally, results generated by the SLBM are close to be optimal, and there is not much room left for improvement. The number of nonoptimal paths to each destination reaches its maximum when the number of destinations reaches half of the total number of nodes. This is when the effect of incorporating more information in path selection is manifested.

In summary, the performance of ASBM and MSLBM is close to the optimal result. On the average ASBM outperforms MSLBM, which in turn outperforms SLBM. MSLBM seems to be the most cost-effective approach.

6 Conclusion

We have studied two types of fault-tolerant multicasting schemes based on limited global information. The concept of safety level associated with each node has been used to represent limited global information. Three algorithms have been proposed which are applicable to injured hypercubes with up to $n-1$ faulty nodes. Results show that when the source node is safe, all three algorithms find the shortest path for each destination node. When the source node is nonsafe, the multicasting can be initiated from one of its safe neighbors. Then in the worst case, one extra time step is required for each destination node. Simulation results show that the performance of both the modified safety-level-based multicasting and the address-sum-based multicasting algorithms are very close to the traffic optimal solution with the former being a more cost-effective approach.

References

[1] Y. Lan. Adaptive fault-tolerant multicasting in faulty hypercube computers. Submitted to Journal of Parallel and Distributed Computing.

[2] Y. Lan. Fault-tolerant multi-destination routing in hypercube multicomputers. *Proc. of 12th International Conference on Distributed Computing Systems.* June 1992, 632-639.

[3] Y. Lan, A. H. Esfahanian, and L. M. Ni. Multicast in hypercube mutiprocessors. *Journal of Parallel and Distributed Computing.* 8, 1990, 30-41.

[4] A. C. Liang, S. Bhattacharya, and W. T. Tsai. Fault-tolerant multicasting on hypercubes. To appear in Journal of Parallel and Distributed Computing.

[5] X Lin and L. M. Ni. Multicast communication in multicomputer networks. *Proc. of 1990 International Conference on Parallel Processing.* III, 1990, 114-118.

[6] G. D. Pifarre, S. A. Felperin, L. Gravano, and J.L.C. Sanz. Routing techniques for massively parallel systems. *Proceedings of the IEEE.* 74, (4), April 1991, 488-503.

[7] Y. Saad and M. H. Schultz. Topological properties of hypercubes. *IEEE Transactions on Computers.* 37, (7), July 1988, 867-872.

[8] J. P. Sheu and M. Y. Su. A multicast algorithm for hypercube multiprocessors. *Proc. of 1992 International Conference on Parallel Processing.* III, 1992, 18-22.

[9] J. Wu. Broadcasting in injured hypercubes using incomplete spanning binomial trees. TR-CSE-92-29, Dept. of Computer Science and Engineering, Florida Atlantic University, Nov. 1992.