[4] *Alpha architecture handbook*, Digital Equipment Corporation, Maynard, MA, 1992.

[5] T.-Y. Yeh and Y.N. Patt, "Two-level adaptive training branch prediction," *Proc. 24th ACM/IEEE Int'l Symp. and Workshop on Microarchitecture*, pp. 51-61, Nov. 1991.

[6] T.-Y. Yeh and Y.N. Patt, "Alternative implementations of two-level adaptive branch prediction," *Proc. 19th Annual Int'l Symp. on Computer Architecture*, pp. 124-134, May 1992.

[7] S.-T. Pan, K. So, and J.T. Rahmeh, "Improving the accuracy of dynamic branch prediction using branch correlation," *Proc. 5th Int'l Conference on Architectural Support for Programming Languages and Operating Systems*, pp. 76-84, Oct. 1992.

[8] T.-Y. Yeh and Y.N. Patt, "A comparison of dynamic branch predictors that use two levels of branch history," *Proc. 20th Annual Int'l Symp. on Computer Architecture*, pp. 257-266, May 1993.

[9] R. Nair, "Branch behavior on the IBM RISC System/6000," *Research Report RC 17859*, IBM T.J. Watson Research Center, Yorktown Hts., NY, Apr. 1992.

[10] C.G. Ponder, "Studies in branch prediction," *Report UCRL-ID-106077*, Lawrence Livermore National Laboratory, Sept. 1990.

[11] G.F. Grohoski, "Machine organization of the IBM RISC System/6000 processor," *IBM Journal of Research and Development*, vol. 34, no. 1, pp. 37-58, Jan. 1990.

[12] S. McFarling and J. Hennessy, "Reducing the cost of branches," *Proc. 13th Annual Int'l Symp. on Computer Architecture*, pp. 396-403, June 1986.

[13] D.J. Lilja, "Reducing the branch penalty in pipelined processors," *IEEE Computer*, pp. 47-55, July 1988.

Safety Levels–An Efficient Mechanism for Achieving Reliable Broadcasting in Hypercubes

Jie Wu

***Abstract*—We consider a distributed broadcasting algorithm for injured hypercubes using incomplete spanning binomial trees. An injured hypercube is a connected hypercube with faulty nodes. The incomplete spanning binomial tree proposed in this paper is a useful structure for implementing broadcasting in injured hypercubes. It is defined as a subtree of a regular spanning binomial tree that connects all the nonfaulty nodes. We show that in an injured n-dimensional hypercube with m faulty nodes, there are at least $2^n - 2^m$ source nodes (called l-nodes), each of which can generate an incomplete spanning binomial tree. A method is proposed to locate a large subset of the l-node set using the concept of safety level. The safety level of each node in an n-dimensional hypercube can be easily calculated through $n - 1$ rounds of information exchange among neighboring nodes. An optimal broadcast initiated from a safe node is proposed. When a nonfaulty source node is unsafe and there are at most $n - 1$ faulty nodes in an injured n-dimensional hypercube, the proposed broadcasting scheme requires at most $n + 1$ steps.**

***Index Terms*—Binomial trees, broadcasting, fault tolerance, hypercubes.**

I. Introduction

Efficient *broadcasting* [3] of data is one of the keys to the performance of a hypercube system. Basically, broadcasting is the process of transmitting data from one node, called the source node, to all the other nodes once and only once. Broadcasting provides basic functions to implement distributed agreement, clock synchronization, and broadcast-and-aggregate type of algorithms. We define an *injured hypercube* [2] as a connected hypercube with faulty nodes. Broadcasting in an injured hypercube is defined as successful broadcasting of a datum to all the nonfaulty nodes. The concept of *incomplete spanning binomial tree* is introduced to implement the broadcasting process. An incomplete spanning binomial tree in an n-dimensional injured hypercube is a connected subgraph of an n-level spanning binomial tree with the same root node that connects all the nonfaulty nodes in the cube, and its root node is called l-node.

Lee and Hayes [5] proposed the concept of *safe node* which requires a stronger condition than the one that defines l-nodes. (Therefore, the safe node set is a subset of the l-node set.) The safe node set can be decided in $O(n^2)$ rounds of information exchange among neighboring nodes. However, the broadcasting algorithm based on this definition of safe node is applicable to injured hypercubes with no more than $\lceil \frac{n}{2} \rceil$ node failures. That is, there are cases when no safe nodes exist in an injured hypercube with more than $\lceil \frac{n}{2} \rceil$ faulty nodes. Wu and Fernandez [10] gave a refined definition of safe nodes by relaxing certain conditions and hence increasing the size of the safe node set and raising the degree of fault tolerance. The process that identifies the node status needs fewer rounds than the one in [5] in general. However it still requires $O(n^2)$ rounds in the worst case.

In this paper, we propose the concept of *safety level*, which is an enhancement of the safe node concept by further weakening its definition. Each node in an n-dimensional hypercube is assigned an integer within the range of 0 to n. A node with safety level n is still called safe node. The safety level is an approximate measure of the number and distribution of faulty nodes in the neighborhood, rather than just the number of faulty nodes. We provide a process that identifies the node status in $n - 1$ rounds of information exchange among neighboring nodes. Simulation results show that the safe node set is very close to the l-node set when $m < n$. A broadcasting scheme is proposed which uses the safety level of each node. It is shown that broadcast from a safe node is both *time* and *traffic* optimal [4], where time is measured by the number of hops (or steps) required to complete a broadcasting and traffic is a measure of the total number of messages transmitted from one node to another in the broadcasting process. Moreover, it is proved that, for each nonfaulty but unsafe node, there is at least one safe neighbor when $m < n$. The same broadcasting scheme can be used by selecting a safe neighbor as the source node. A total of $n + 1$ steps is required in this case.

The proposed method differs from the existing fault-tolerant broadcasting methods which are based on either local information [6] or global information ([1], [7], [9]). Local-information-based broadcasting algorithms normally require routing history as part of message to be broadcast, and results are not optimal. Global-information-based broadcasting algorithms, although having their merits of simplicity and optimality, require a process that collects global information. The broadcasting based on limited information is a compromise of the above two schemes. In the proposed method limited global information is captured in the safety level associated with each node. Since this type of information is easy to update and maintain and the optimality is still preserved, this method is more attractive than the existing ones.

The safety level is the first practical model that captures fault information in terms of the distribution and the number of faults, rather than just in terms of the number of faults. In a separate paper, we show that it can also be used to achieve optimization in routing and

Manuscript received Mar. 5, 1993; revised May 2, 1994.

The author is with the Dept. of Computer Science and Engineering, Florida Atlantic University, Boca Raton, FL 33431; e-mail jie@cse.fau.edu.

IEEECS Log Number C95040.

multicasting [11] in an injured hypercube.

We make the following assumptions for the techniques used in this paper: 1) All the node faults are fault-stop, i.e., there are no malicious faults. 2) Fault detection and diagnosis algorithms exist, but we do not require such an algorithm to be perfect. We do assume that each node know exactly the safety status of all its neighbors. Due to the limitation of space, several lengthy proofs of theorems are not shown in this paper. The reader may refer to [8] for detail.

II. Notation and 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 a has address $a_n a_{n-1} \cdots a_0$ with $a_i \in \{0, 1\}$, $1 \le i \le n$, and a_i is called the ith bit (also called the ith dimension) of the address. We denote node a^i the neighbor of a along dimension i. Every m-dimensional subcube Q_m (or m-subcube) has a unique address $q_n q_{n-1} \ldots q_0$ with $q_i \in \{0, 1, *\}$, $1 \le i \le n$, where exactly m bits take the value *, a don't-care symbol representing either 0 or 1. An n-level binomial tree B_n is constructed out of two B_{n-1}s by adding an edge to make the root of one B_{n-1} become the leftmost offspring of the root of another B_{n-1}. B_1 contains only one node. Clearly, a B_n has the same number of nodes as a Q_n, and B_n is a subgraph of Q_n.

Definition 1. *An incomplete spanning binomial tree in an injured n-dimensional hypercube is a connected subgraph of an n-level spanning binomial tree with the same root node which connects all the nonfaulty nodes in the cube. A node is l-node if it is a root node of an incomplete spanning binomial tree.*

Theorem 1. *The number of l-nodes in an injured n-dimensional hypercube with m (< n) faulty nodes is* $f(n, m) \ge 2^n - 2^m$.

It is conjectured that the determination l-node status of a node is NP-hard. Instead we assign a *safety level* to each nonfaulty node. The safety level associated with a node is an approximate measure of the number and distribution of faulty nodes in the neighborhood, rather than just the number of faulty nodes. Let $S(a) = k$ be the safety status of node a, where k is referred to as the level of safety, and a is called *k-safe*. A faulty node is 0-safe which corresponds to the lowest level of safety, while an n-safe node (also referred to as a *safe node*) corresponds to the highest level of safety. A node with k-safe status is called *unsafe* if $k \ne n$.

Definition 2: *Let* $(S_0, S_1, S_2, \ldots, S_{n-1})$, $0 \le S_i \le n$, *be the ascending safety-level sequence of node a's n neighboring nodes in an n-cube, such that* $S_i \le S_{i+1}$, $0 \le i \le n - 1$. *The safety level of node a is defined as: if* $(S_0, S_1, S_2, \ldots, S_{n-1}) \ge (0, 1, 2, \ldots, n - 1)$[2] *then* $S(a) = n$ *else if* $(S_0, S_1, S_2, \ldots, S_{k-1}) \ge (0, 1, 2, \ldots, k - 1) \wedge (S_k = k - 1)$ *then* $S(a) = k$.

It can be proved that a safe node is a root node of an incomplete spanning binomial tree, and therefore it is an l-node; however, in general, the converse is not true. For example, in an injured Q_4 with faulty nodes 1001, 0011, 0100, and 0110. Node 0000 is an l-node but not a safe node (it is a 2-safe node). We will show that for most cases an l-node is a safe node in an injured hypercube with m (< n) faults. Therefore, we can use the safety level to approximate the l-node status of each node in an injured hypercube. The following algorithm (GS) calculates the safety level of each node in an n-cube.

2 $seq_1 \ge seq_2$ if and only if each element in seq_1 is greater or equal to the corresponding element in seq_2.

```
Algorithm GLOBAL_STATUS (GS)
{Initially all nonfaulty nodes are $n$-safe and round = 1.}
begin
  while round $\le \Delta$
    parbegin
      NODE_STATUS($a(i)$), $0 \le i \le 2^n - 1$
    parend
    round : = round + 1
end.
Procedure NODE_STATUS ($a(i)$)
begin
  at node $a(i)$ determine the ascending status sequence of neigh-
  boring nodes $(S_0, S_1, S_2, \ldots, S_{n-1})$
  if $(S_0, S_1, S_2, \ldots, S_{n-1}) \ge (0, 1, 2, \ldots, n - 1)$
    then mark $a(i)$ as $n$-safe (or safe);
  if $((S_0, S_1, S_2, \ldots, S_{k-1}) \ge (0, 1, 2, \ldots, k - 1)) \wedge (S_{k-1} = k - 1)$
    then mark $a(i)$ as $k$-safe;
end.
```

Fig. 1 shows the safety level of each node in a faulty 4-cube with five faulty nodes (represented as black nodes). Based on the safety level definition, the safety levels of all the nodes that have two (or more) faulty neighbors will be changed to 1 after the first round, as in the case for nodes 0001, 0110, 0111, 1011, 1100, 1101 in Fig. 1. That is, the effect of 0-safe status of faulty nodes will first propagate to their neighbors, then neighbors' neighbors, and so on. For example, after the second round the safety level of node 1001 changes to 2, because this node has three 1-safe neighbors. The safety level of every node remains stable after two rounds, and each value represents the safety level of the corresponding node.

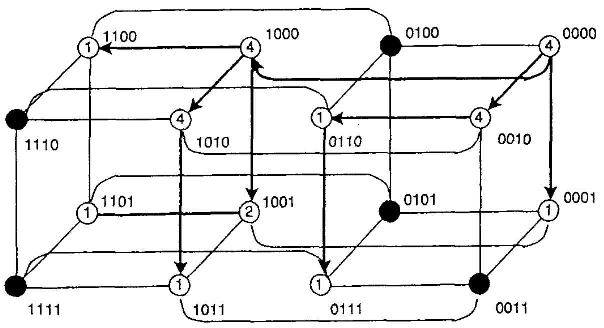

Fig. 1. A 4-dimensional injured hypercube.

Theorem 2. *The GS algorithm identifies a k-safe (k ≠ n) node of an n-cube in k rounds, i.e., at the* kth *round this node reaches a stable status.*

Proof. We prove this theorem using mathematical induction on k, the safety level of a node. When $k = 1$, clearly there are at least two faulty (0-safe) neighboring nodes for any 1-safe node. This node can identify its safety level in one step. Suppose for all k-safe nodes, $k \le i$, exactly k rounds are required for these nodes to stabilize their status. Based on Definition 1 an $(i + 1)$-safe node can identify its status once all its neighbors, which have a safety level lower than $i + 1$, have the stable status. By the induction assumption, exactly after the ith round all those neighbors are stabilized, and it takes one extra step for this $(i + 1)$-safe node to be stabilized. □

COROLLARY. *To identify the status of all the nonfaulty nodes in any faulty hypercube (which might be a disconnected hypercube), the number of rounds (Δ in GS) is $n - 1$, where n is the dimension of the faulty hypercube.*

III. INJURED HYPERCUBE BROADCASTING

In this section, we consider an optimal broadcasting scheme using the incomplete spanning binomial tree structure. Before proceeding, we look at another definition of incomplete spanning binomial trees (and for regular spanning binomial trees). A B_n can be constructed from $B_{n-1}, B_{n-2}, \ldots, B_1, B_0$ by using a node s (the root node) to connect the root nodes of these trees. To construct B_n in an injured n-cube, each B_i should be an incomplete spanning binomial tree of an i-subcube. More specifically, $B_{n-1}, B_{n-2}, \ldots, B_1, B_0$ are incomplete spanning binomial trees in injured subcubes $Q'_{n-1}, Q'_{n-2}, \ldots, Q'_1, Q'_0$, $Q_0 = s$, respectively, where these subcubes constitute a partition of Q_n. The above partition can be generated following procedure: Q_n is split into two $(n-1)$-subcubes, Q_{n-1} $(s \in Q_{n-1})$ and $Q'_{n-1}(s \notin Q_{n-1})$, along the d_0^{th} dimension. Q_{n-1} is further divided into two $(n-2)$-subcubes along the d_1^{th} dimension. This process continues until Q_1 is divided into two 0–subcubes, Q'_0 and $Q_0 = s$, along the d_{n-1}^{th} dimension. The dimension sequence $d_0 d_1 \ldots d_{n-1}$ that determines the partition is called the *coordinate sequence* (*cs*). The set $\{Q'_{n-1}, Q'_{n-2}, \ldots, Q'_0, Q_0 = s\}$ is a partition of Q_n. For example, the splitting process of $Q_3 = {*}{*}{*}$ at $s = 010$ with coordinate sequence $d_0 d_1 d_2 = 213$ can be described as follows:

$$***(Q_3) = *1*(Q_2) + *0*(Q'_2)$$
$$*1*(Q_2) = *10(Q_1) + *11(Q'_1)$$
$$*10(Q_1) = 010(Q_0) + 110(Q'_0)$$

Clearly, $\{Q'_2, Q'_1, Q'_0, Q_0 = s\} = \{*0*, *11, 110, 010\}$ forms a partition of $Q_3 = {*}{*}{*}$. The above process can be continuously applied to each subcube (which is not a single node) in the partition using appropriate coordinate sequences until each subcube is partitioned into a set consisting of single node only.

Let d_j, $0 \le j \le n-1$, be the dimension along which the source node s connects the root node (s^{d_j}) of B_j. $d_0 d_1 \ldots d_{n-1}$ is the coordinate sequence at node s. A j-safe node can generate an incomplete spanning binomial tree for any injured j-subcube. Therefore, it is sufficient to require each s^{d_j} in charge of generating a B_j to have a safety level of no less than j. Based on the definition of safety level, the above condition can always be satisfied if s is safe, because a safe node is a root of an incomplete binomial tree. More specifically, suppose $(S_0, S_1, S_2, \ldots, S_{n-1})$ is the s's status sequence of neighboring nodes in an ascending order and $(d_0, d_1, d_2, \ldots, d_{n-1})$ is the corresponding neighbor dimension sequence. Then we use the neighbor dimension sequence as the coordinate sequence to partition the cube, that is, each s^{d_j} is responsible for generating a B_j in a j-subcube. Obviously, if s is safe, then based on Definition 2 each neighbor s^{d_j} has a safety level of no less than j, and this guarantees a successful generation of an incomplete spanning binomial tree in a j-subcube with node s^{d_j} being the root node. The same procedure can be recursively applied to generate those B_js in each injured j-subcube.

To implement a broadcasting algorithm using neighbors' status, two sequences—*ascending status sequence* $(S_0, S_1, S_2, \ldots, S_{n-1})$ and corresponding *neighbor dimension sequence* $(d_0, d_1, d_2, \ldots, d_{n-1})$—should be kept at each node. Both sequences are available at each node after one application of the GS algorithm. The coordinate sequence can be directly derived from the ascending status sequence. An n-bit control word LABEL is used to represent a subcube in the partition, where each bit is a binary number with each 1 representing an * in the subcube.

ALGORITHM INJURED_HYPERCUBE_BROADCASTING (IHB)

```
{At the source node, $LABEL[j] = 1, 0 \le j < n$.}
begin
  for $j = n - 1$ downto 0
    if $LABEL[d_j] \ne 0$ and $S_j \ne 0$ /* if $S_j = 0$ then the neighbor
    along dimension $d_j$ is faulty */
      then begin
        $LABEL[d_j] := 0$;
        send the broadcast data and $LABEL$ to the neigh-
        bor along the $d_j^{th}$ dimension;
        end
end.
```

Table I shows the ascending status sequence (*ass*) and the corresponding neighbor dimension sequence (*nds*) for each node in the injured Q_4 of Fig. 2. Note that when two or more neighbors have the same safety level, there is more than one possible corresponding neighbor dimension sequence. Table I shows only one possible order for these cases.

THEOREM 3. *The IHB algorithm generates an incomplete spanning binomial tree in an injured Q_n with a safe node being the source node, i.e., it sends broadcast data from a safe node to every nonfaulty node along a shortest path.*

TABLE I
THE *ass* AND *nds* FOR EACH NODE IN THE INJURED Q_4 OF FIG. 1

node	*ass*	*nds*	*node*	*ass*	*nds*
0000	(0,1,4,4)	(3,1,2,4)	0001	(0,0,2,4)	(2,3,4,1)
0010	(0,1,4,4)	(1,3,2,4)	0011	(1,1,1,4)	(2,3,4,1)
0100	(0,1,1,4)	(1,4,2,3)	0101	(0,1,1,1)	(1,2,3,4)
0110	(0,0,1,4)	(2,4,1,3)	0111	(0,0,0,1)	(2,3,4,1)
1000	(1,2,4,4)	(3,1,2,4)	1001	(1,1,1,4)	(2,3,4,1)
1010	(0,1,4,4)	(3,1,2,4)	1011	(0,0,2,4)	(3,4,2,1)
1100	(0,0,1,4)	(2,4,1,3)	1101	(0,1,1,2)	(2,4,1,3)
1110	(0,1,1,4)	(1,2,4,3)	1111	(0,1,1,1)	(1,2,3,4)

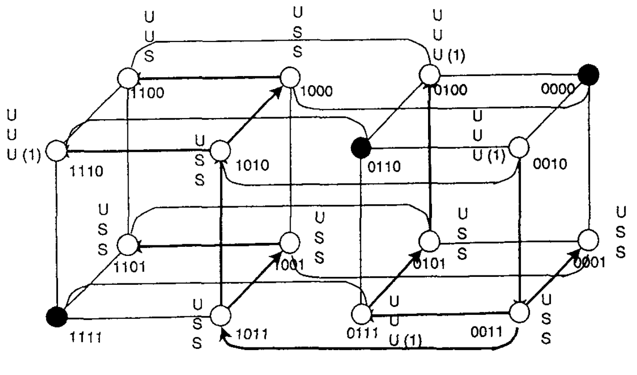

Fig. 2. Safe nodes in an injured Q_4 using different definitions.

PROOF. We prove a stronger statement: The IHB algorithm generates an incomplete spanning binomial tree in a Q_k if the source is an l ($\geq k$)-safe node. Obviously, this theorem is a special case of the above statement. We prove this statement by induction on k. It is clear that the statement is true for $k = 1, 2$. We assume that the statement is true for all $k < i$. For the case that $k = i$, we partition Q_i into $\{Q'_{i-1}, Q'_{i-2}, \ldots, Q'_1, Q'_0, Q_0 = s\}$ using a coordinate sequence $d_0d_1 \ldots d_{i-1}$. Obviously, each s^{d_j}, the neighbor of s along dimension d_j, $0 \leq j \leq i-1$, has a safety level greater than or equal to j. Therefore, using the induction assumption, each s^{d_j} can generate an incomplete spanning binomial tree in Q'_j. Using the definition of binomial tree, we can construct an incomplete spanning binomial tree in Q_i with s being the root node by connecting s to the root node s^{d_j} of each subcube Q'_j. □

The IHB algorithm not only guarantees time optimality in terms of the number of steps but also traffic optimality in terms of the number of links. In this scheme, destination addresses are completely "compressed," and each nonfaulty node receives an address together with the broadcast data once and only once.

THEOREM 4. *Any broadcasting tree generated by the IHB algorithm is a special type of incomplete spanning broadcasting trees, where all the faulty nodes are leaves in the tree.*

The proof of this theorem is straightforward and is based on the definition of incomplete spanning broadcasting trees and the IHB algorithm.

Fig. 1 shows how the IHB algorithm works on a Q_4 with five faulty nodes. By using the GS algorithm, we derive that nodes 0000, 0010, 1000, and 1010 are safe nodes among the nonfaulty nodes. Suppose 0000 is the source node with a LABEL = [1111]. Among all the neighboring nodes, 0010 and 1000 are safe, 0001 is 1-safe, and 0100 is 0-safe. Therefore, at node 0000 the ascending status sequence is (0, 1, 4, 4) with two possible neighbor dimension sequences: (3, 1, 2, 4) and (3, 1, 4, 2). Suppose sequence (3, 1, 2, 4) is used as in Table I, then the neighbor along dimension 4, node 1000, receives the 3-cube 1***, which is represented by the LABEL = [0111]. Node 0010, the neighbor along dimension 2, receives the 2-cube 0*1* represented by LABEL = [0101]. The unsafe node 0001, the neighbor along dimension 1, receives the 1-cube 0*01 represented by a LABEL = [0100]. The faulty node 0100, the neighbor along dimension 3, does not receive any data. In the next step, node 0010, which has subcube 0*1* and has (0, 1, 4, 4) and (1, 3, 2, 4) as its *ass* and *nds*, sends 011*, represented by LABEL = [0001], to neighbor 0110 along dimension 3 (dimension 3 precedes dimension 1 in *nds*), and the remaining faulty 0-cube 0011 is discarded. No data are sent from node 0001 to the faulty node 0101. In the same step, node 1000, which has subcube 1*** and has (1, 2, 4, 4) and (3, 1, 2, 4) as its *ass* and *nds*, sends 1*1*, represented by a LABEL = [0101] to node 1010, 1*01 represented by label [0100] to node 1001, and 0-cube 1100 represented by label [0000] to node 1100. In the third step, 0-cubes represented by [0000] are sent from node 0110 to node 0111, from node 1001 to node 1101. The 1-cube 1*11 represented by [0100] is sent from node 1010 to node 1011. No data are sent from node 1010 to the faulty node 1110. In the last step, since node 1111 is faulty, no data are sent to it. Assume that each message transmission takes one time unit. The total broadcasting time for the above example is 3.

THEOREM 5. *In an injured n-cube with fewer than n faulty nodes, every unsafe node has at least one safe neighbor.*

In the injured 4-cube of Fig. 2 with four faulty nodes, {0011, 1010}, {0101, 1100}, {0011, 0101}, {1010, 1100} are safe neighbor sets for unsafe nodes 0010, 0100, 0111, and 1110, respectively. Suppose unsafe node 0010 is the source node and node 0011 is the selected safe neighbor to be new source node. The broadcasting can be completed in five steps as shown in Fig. 2. Therefore, a broadcast from an unsafe node requires one extra step.

It is clear that the safe node set is a subset of l-node set. An unsafe node is either an l-node or a non l-node. The union of the safe node set and the unsafe equals the union of the l-node set and the non l-node set. Based on Theorems 4 and 5, we conclude that for any injured Q_n with less than n faulty nodes, a broadcasting from an l-node requires n steps if this node is safe or $n + 1$ steps if this node is unsafe. A broadcasting from any non l-node (which is unsafe) requires $n + 1$ steps.

IV. COMPARISONS

In this section, we compare the proposed method with other approximations of l-nodes using the concept of a safe node. The following two measures are used:

1) The number of steps required to determine the status of a node.
2) The size of the subset of l-nodes identified as safe nodes.

The following are two other definitions of safety status of a node:

1) (Lee and Hayes [5]) A nonfaulty node is unsafe if and only if there are at least two unsafe or faulty neighbors.
2) (Wu and Fernandez [10]) A nonfaulty node is unsafe if and only if either of the following conditions is true:
   (a) There are two faulty neighbors or
   (b) there are at least three unsafe or faulty neighbors.

In general, it is difficult, if not impossible, to compare the size of safe node sets because each safe node set depends on the distribution of faulty nodes. However, it is clear that, for each distribution of faulty nodes, the safe node set obtained using the definition in this paper contains the set using the definition in [10], which in turn contains the set using the definition in [5]. The example in Fig. 2 can be used to illustrate this point. We use U and S to represent unsafe and safe status of each node, respectively. Each node has three safety statuses based on three different definitions (with the status based on the one in [5] at the top, the one in [10] at the middle, and the one in this paper at the bottom). The l-node set in this example is {0001, 0011, 0101, 1000, 1001, 1010, 1011, 1100, 1101}. Using the safety definition proposed in this paper, we obtain the safe node set which is the same as the l-node set. Using the definition in [10], we have {0001, 0011, 0101, 1000, 1001, 1010, 1011, 1101} as the safe node set with the absence of node 1100. The safe node set is empty using the definition in [5].

A simulation study was conducted on Q_6s and Q_7s, with the number of faulty nodes ranging from 1 to 10. The size of the safe nodes set under different definitions is compared with the size of the l-node set. Table II lists the average percentages of l-nodes identified as safe nodes under three different definitions. The numbers under the columns Wu, Wu-Fe, and Lee-Hayes represent the percentages of l-nodes identified as safe nodes under the definition in this paper, the Wu and Fernandez' definition, and the Lee and Hayes' definition, respectively. The simulation results show that when the number of faulty nodes is smaller than the dimension of the hypercube, the size of the safe node set under the safety definition in this paper is very close to the size of the l-node set. The safety definition used in this paper is superior to the ones in [5] and [10] in terms of the size of the safe node set.

TABLE II
THE PERCENTAGES OF l-NODES IDENTIFIED AS SAFE NODES IN (a) Q_6S AND (b) Q_7S

faults	*Wu*	*Wu-Fe*	*Lee-Hayes*
1	100.000	100.000	100.000
2	100.000	100.000	100.000
3	99.942	99.943	95.642
4	99.836	99.836	66.694
5	99.216	99.001	27.110
6	98.109	96.699	10.398
7	96.576	91.158	4.380
8	93.604	79.236	2.226
9	88.544	59.897	0.790
10	80.989	45.461	0.426

(a)

faults	*Wu*	*Wu-Fe*	*Lee-Hayes*
1	100.000	100.000	100.000
2	100.000	100.000	100.000
3	99.827	99.827	98.741
4	99.824	99.824	88.769
5	99.820	99.820	53.685
6	99.491	99.323	22.492
7	99.048	98.431	8.302
8	98.744	96.591	2.890
9	97.889	91.997	1.250
10	96.636	82.503	0.696

(b)

V. CONCLUSIONS

In this paper, we have studied distributed broadcasting in injured hypercubes using incomplete spanning binomial trees. We have determined a lower bound on the number of l-nodes, each of which can generate an incomplete spanning binomial tree. A method has been proposed to locate a large subset of the l-node set using the concept of safety level. An optimal broadcasting initiated from a safe node has also been proposed. When this scheme is applied to an unsafe source node and there are at most $n - 1$ faulty nodes in an injured n-cube, at most $n + 1$ steps are required to complete a broadcast.

REFERENCES

[1] Y. Chang, "Fault tolerant broadcasting in SIMD hypercubes," *Proc. Fifth IEEE Symp. on Parallel and Distributed Processing*, pp. 348-351, 1993.

[2] M.S. Chen and K.G. Shin, "Depth-first search approach for fault-tolerant routing in hypercube multicomputers," *IEEE Trans. Parallel and Distributed Systems*, vol. 1, no. 2, pp. 152-159, Apr. 1990.

[3] S.L. Johnson and C.-T. Ho, "Optimal broadcasting and personalized communication in hypercubes," *IEEE Trans. Computers*, vol. 38, no. 9, pp. 1,249-1,268, Sept. 1989.

[4] Y. Lan, A.H. Esfahanian, and L.M. Ni, "Multicast in hypercube multiprocessors," *J. Parallel and Distributed Computing*, vol. 8, pp. 30-41, 1990.

[5] T.C. Lee and J.P. Hayes, "A fault-tolerant communication scheme for hypercube computers," *IEEE Trans. Computers*, vol. 41, no. 10, pp. 1,242-1,256, Oct. 1992.

[6] Z. Li and J. Wu, "A multidestination routing scheme for hypercube multiprocessors," *Proc. 1991 Int'l Conf. Parallel Processing*, vol. II, pp. 290-291, Aug. 1991.

[7] C.S. Raghavendra, P.J. Yang, and S.B. Tien, "Free dimensions—an effective approach to achieving fault tolerance in hypercubes," *Proc. 22nd Int'l Symp. Fault-Tolerant Computing*, pp. 170-177, 1992.

[8] J. Wu, "Broadcasting in injured hypercubes using limited global information," TR-CSE-92-39, Dept. Computer Science and Eng., Florida Atlantic Univ., Nov. 1992.

[9] J. Wu, "An optimal fault-tolerant nonredundant broadcasting scheme in injured hypercubes," *J. Parallel and Distributed Computing*, vol. 22, pp. 295-313, 1994.

[10] J. Wu and E.B. Fernandez, "Reliable broadcasting in faulty hypercube computers," *Microprocessing and Microprogramming*, vol. 39, pp. 43-53, 1993.

[11] J. Wu and K. Yao, "Multicasting in injured hypercubes using limited global information," to appear in *IEEE Trans. Computers*.