# Reliable Broadcasting in Faulty Hypercube Computers

Jie Wu and Eduardo B. Fernandez

Department of Computer Science and Engineering
Florida Atlantic University
Boca Raton, FL 33431

## Abstract

*We propose a non-redundant broadcasting algorithm for faulty hypercube computers. The concept of unsafe node is introduced to identify those non-faulty nodes that will cause a detour or backtracking because of their proximity to faulty nodes. It is assumed that each healthy node, safe or unsafe, knows the status of all the neighboring nodes. The broadcasting is optimal, namely, a message is sent to each node via a Hamming distance path if the broadcasting is initiated from a safe node. We also show that when the source node is unsafe and there is an adjacent safe node then the broadcasting can be achieved with only one more time step than the fault-free case.*

## 1 Introduction

The hypercube [11] has been a dominating topology used because of its strong connectivity, regularity, symmetry, and ability to embed many other topologies. Numerous research projects have considered hypercube design aspects [13] and hypercube applications [14], [10]. These have resulted in several commercial products, such as the Intel iPSC [6], FTS T-series [2], and NCUBE [4].

Efficient *broadcasting* of data [7], [15], 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 ($s$), to all the other nodes. By fault-tolerant broadcasting we mean the successful broadcasting of a message in the presence of faulty components (links and/or nodes).

We consider in this paper a fault-tolerant broadcasting scheme based on non-redundant broadcasting, where each node will be visited by exactly one copy of the broadcast data. In this case normally a *spanning broadcasting tree* is first constructed, then the broadcast data is sent from the root of the tree, the source node, to all the other nodes along the paths in the tree. The methods in this category can be further divided into approaches using global network information, limited global information, and local network information. The methods can also be categorized by the type of faults to be tolerated, which can be node failures or link failures or combinations of both. In this paper, we consider broadcasting using limited global information and where the hypercubes are subject only to node failures.

The standard algorithm for non-redundant broadcasting was given by Sullivan and Bashkow [16] and uses a binomial tree as broadcasting tree. Using only local network information, namely, the status of neighboring links or nodes which can be either faulty or nonfaulty, Al-Dhelaan and Bose [1] proposed binomial-tree based broadcasting with local network information which can tolerate at least one fault in each path in the binomial tree. This approach was enhanced by Wu and Fernandez [18] whose algorithm guarantees an optimal number of time units for completing the broadcast. Using global network information, Wu [17] proposed a reliable broadcasting algorithm which is based on a variation of the binomial tree structure. Using limited global network information by classifying nonfaulty nodes into safe and unsafe (a node is unsafe if it has unsafe or faulty nodes in its neighborhood), Lee and Hayes [8] proposed an efficient broadcasting algorithm which can be used for hypercubes with fewer than $\lceil \frac{n}{2} \rceil$ node failures. Their algorithm requires the same number of time units as the fault-free case when the source node is safe, and one more time unit than the fault-free case when the source is a unsafe node with at least one safe neighbor. In this paper, we propose an enhanced broadcasting algorithm based on Lee and Hayes' algorithm. The concepts of safe and unsafe nodes are still used but they are defined differently. It is shown that the proposed algorithm can tolerate more node failures while still keeping the simplicity of Lee and Hayes' algorithm.

1060-9857/92 $3.00 ©1992 IEEE

This paper is organized as follows. Section 2 defines basic notation and preliminaries. Section 3 proposes an efficient non-redundant fault-tolerant broadcasting algorithm based on a new classification of safe and unsafe nodes. Performance aspects are discussed in Section 4. Section 5 presents some conclusions.

## 2 Notation and Preliminaries

An $n$-dimensional hypercube ( $n$-cube ) $Q_n$ contains $2^n$ nodes. Every node $a$ has address $a_n a_{n-1} \cdots a_1$ with $a_i \in \{0,1\}$, $1 \leq i \leq n$, and $a_i$ is called the $i$ *th* bit (also called the $i$ *th* dimension) of the address. Every $m$-dimensional subcube $Q_m$ $n \leq m$, has a unique address $q_n q_{n-1} \ldots q_1$ with $q_i \in \{0,1,*\}$, $1 \leq i \leq n$, where exactly $m$ bits take the value $*$, and $*$ is a don't care symbol. A hypercube itself can be considered as a special subcube where all the bits take the value $*$. Each node is also a special subcube in which no bit takes the value $*$. A *coordinate sequence* (*cs*): $i_1 i_2 \ldots i_m$ is a permutation of those bit positions that take values $*$ in $Q_m$. Let $Q_m^i = q_n q_{n-1} \cdots \overline{q}_i \cdots q_1$ denote the neighboring $m$-subcube of $Q_m = q_n q_{n-1} \cdots q_i \cdots q_1$, along the *ith* dimension, where $q_i \neq *$. $(Q_m)_q^i$ denotes a substitution of the *ith* bit of $Q_m$ by $q \in \{0,1,*\}$.

For example, if $n = 3$ and $Q_1 = **0$ then $Q_1^1 = **1$ and $(Q_1)_0^2 = *01$. $cs = 23$ is a coordinate sequence of subcube $**1$ (the bit position is counted from the right).

**Definition 1** [9]: The *splitting process* of $Q_m$ with respect to coordinate sequence $i_1 i_2 \ldots i_m$ at node $s = s_n s_{n-1} \ldots s_1$ is defined as follows:

$$\begin{aligned} Q_m &= Q_{m-1} + Q'_{m-1} \\ Q_{m-1} &= Q_{m-2} + Q'_{m-2} \\ &\vdots \\ Q_1 &= Q_0 + Q'_0 \end{aligned}$$

where $Q'_{m-j} = Q_{m-j}^{s_{i_j}}$, and $Q_{m-j} = (Q_{m-j+1})_{s_{i_j}}^{i_j}$, for $1 \leq j \leq m$.

The above sequence defines a splitting process of $Q_m$ as follows: $Q_m$ is split into two $(m-1)$-subcubes, $Q_{m-1}(s \in Q_{m-1})$ and $Q'_{m-1}(s \notin Q'_{m-1})$ along the $i_1 th$ dimension. $Q_{m-1}$ is further divided into two $(m-2)$-subcubes along the $i_2 th$ dimension. This process continues until $Q_1$ is divided into two 0-subcubes, $Q'_0$ and $Q_0 = s$ along the $i_m th$ dimension. Clearly, the set $\{Q'_{m-1}, Q'_{m-2}, \ldots, Q'_0, Q_0\}$ is a partition of $Q_n$. For example, the splitting process of $Q_3 = ***$ at $s = 010$ with $cs : i_1 i_2 i_3 = 213$ can be described as follows:

$$\begin{aligned} ***(Q_3) &= *1*(Q_2) + *0*(Q'_2) \\ *1*(Q_2) &= *10(Q_1) + *11(Q'_1) \\ *10(Q_3) &= 010(Q_0) + 110(Q'_0) \end{aligned}$$

Clearly, $\{Q'_2, Q'_1, Q'_0, Q_0\} = \{*0*, *11, 110, 010\}$ forms a partition of $Q_3 = ***$. and $Q_0 = 010$

*Broadcasting* is a process which sends data from one node to all other nodes in the network. Normally it is required that the broadcast data be sent to each node efficiently. This can be done by selecting a shortest path between the source node and the destination node. In hypercubes the shortest paths between two nodes are those paths which have a length equal to the Hamming distance between these two nodes' addresses. Therefore in hypercubes the shortest path is also called the *Hamming distance path.* Note that in hypercubes with no faulty components (links or nodes), shortest path and Hamming distance path are the same. In faulty hypercubes, however, a shortest path could be longer than a Hamming distance path. *Non-redundant broadcasting* is a special broadcasting such that the broadcast data visits each node exactly once. A non-redundant broadcasting process requires to find a *spanning broadcasting tree* in the network. The broadcasting that uses the above splitting process at each node forms a special spanning broadcasting tree called a *binomial tree* [16]. The broadcasting tree in Figure 1 is a binomial tree with 011 as its root.

When a destination subcube $Q'_i$ is sent to a node $a$, it suffices to provide only the locations of $*$ and all the values of non-$*$'s can be directly derived from $a$, since all the non-$*$'s in the address of $Q'_i$ take the same values (0 or 1) as $a$ in the corresponding location. Therefore in the implementation of the broadcasting algorithm, an n-bit *control word* LABEL is used to perform the splitting process and to control the broadcasting process at each node. Each 1 in LABEL corresponds to $*$ in that location and each 0 represents a non-$*$ value. The general binomial-tree-based broadcasting algorithm, which is similar to the one in [8], can be described as follows:

```
Algorithm BROADCASTING
{ $i_1 i_2 \ldots i_n$ is a cs, where $n$ is the size of the hypercube}
begin
  if current_node = source_node then
    for $j = 1$ to $n$  $LABEL[j] := 1$;
  for $j = 1$ to $n$
    if $LABEL[i_j] \neq 0$ then
      begin
```

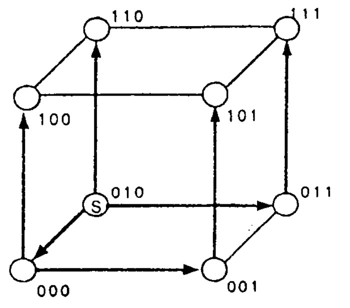

Figure 1: A broadcasting process on $Q_3$ with respect to coordinate sequence 213 at node 010

```
        LABEL[$i_j$] := 0;
        send the broadcast data and LABEL to the
        neighboring node along the $i_j$th dimension;
        end;
end.
```

In the above algorithm, broadcasting to all the neighbors of a given node is performed by a sequence of splitting and sending subcubes. A subcube is split by assigning the value 0 to a particular bit in LABEL and that new label together with the broadcast data is immediately sent to the appropriate neighbor. The splitting process continues on the new LABEL which corresponds to a new subcube one dimension smaller than the previous subcube.

Suppose 213 is used as the *cs* for $Q_3$ : $***$ with 010 as the source node. Then the splitting process generates the partition set $\{*0*, *11, 110, 010\}$ at 011. Then $*0*, *11, 110$ will be sent to 000, 011, 110, respectively. 13, a subsequence of 213, will be used as the *cs* at 000 to further split and broadcast the subcube $*0*$. And 3 will be used as the *cs* at 011 to split and broadcast the subcube $*11$. The broadcasting process of the above example is shown in Figure 1.

In algorithm BROADCASTING the coordinate sequence $i_1 i_2 \ldots i_n$ at the source node is *global* in the sense that coordinate sequences at all the other nodes are subsequences of it. It is possible to allow coordinate sequences at different nodes independent of each other and this type of broadcasting will be adopted in the fault tolerant broadcasting algorithm discussed in the next section.

## 3 Broadcasting in Faulty Hypercubes

We start with an illustration of problems of general broadcasting in hypercubes with node failures. Figure

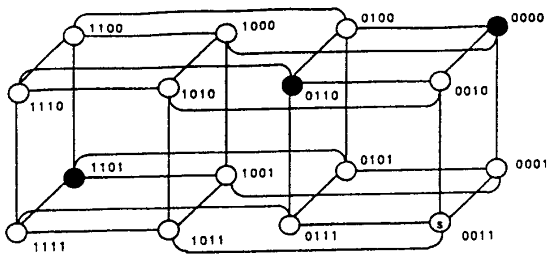

Figure 2: A $Q_4$ with three faulty nodes

2 shows a $Q_4$ with three faulty nodes: 0000, 0110, and 1101. In the figure a black node represents a faulty node and a white node represents a non-faulty node. Suppose node 0011 is the source node and the *cs* at 0011 is 1432, that is $***0, 1**1, 01*1, 0001$ will be broadcast at 0010, 1011, 0111, and 0001. Let's look for example at the broadcasting of $***0$ at 0010. It is clear that there is no shortest path from 0010 to 0100, since there is a faulty node in each of the two shortest paths: 0110 in the path 0010 $\rightarrow$ 0110 $\rightarrow$ 0100; 0000 in the path 0010 $\rightarrow$ 0000 $\rightarrow$ 0100. Therefore the broadcast data can reach node 0100 from 0010 only through a detour path, say 0010 $\rightarrow$ 1010 $\rightarrow$ 1000 $\rightarrow$ 1100 $\rightarrow$ 0100. This process not only lenghtens the broadcasting process, measured in time steps, but also complicates it. The algorithm BROADCASTING defined earlier is no longer applicable for this case.

If 4213 is used as the cs at the source node 0011, then $1***, 0*0*, 0*10, 0111$ will be broadcast at 1011, 0001, 0010, and 0111, respectively. Suppose the broadcasting of $0*0*$ at 0001 is decided by a local $cs = 31$, then 0100 can be reached through a shortest path: 0011 $\rightarrow$ 0001 $\rightarrow$ 0101 $\rightarrow$ 0100. The above example shows that it is very important to select an appropriate *cs* at each node. It is clear that no subcube of more than one dimension should be sent to a node with more than one faulty adjacent node. In the above example node 0010 is such type of node, and therefore 1, the dimension along which 0011 and 0010 are connected, should be placed as the last dimension or the next to last dimension in the *cs* at the source node. We call this type of node *unsafe*, which is defined more precisely as follows.

**Definition 2.** A nonfaulty node in a hypercube is called *unsafe* if one of the following two conditions is true [1]:

[1]In Lee and Hayes' definition a node is unsafe if and only if there are at least two unsafe or faulty neighbors.

- There are at least two faulty neighbors.
- There are at least three unsafe or faulty neighbors.

A subcube is called ***unsafe*** if all its nodes are faulty or unsafe; otherwise the subcube is called *safe*.

In the example shown in Figure 2, 0100 and 0010 are the only unsafe nodes among the non-faulty nodes. Therefore, $Q_4$ in Figure 2 is a safe hypercube.

Before proceeding, we consider an algorithm that identifies all the unsafe nodes in a hypercube. The termination condition in the algorithm is that the global status reaches a stable condition, that is, it becomes stable. The global status consists of the status of all the nodes. Initially all the nonfaulty nodes take the safe node status. A global state has not reached a stable condition if there is at least one node changing status from safe to unsafe in the last round of calculation.

```
Algorithm GLOBAL_STATUS
{Each node keeps a list FAULTY of neighboring faulty
nodes and a list UNSAFE of neighboring unsafe nodes.
Initially all the UNSAFE lists are empty.}
begin
  while the global state is not stable
    parbegin
    NODE_STATUS($a(i)$), $0 \leq i \leq 2^n - 1$;
    parend;
    {Each node determines its status. One calculation
    of the above statement is considered as one round.}
    end.
Procedure NODE_STATUS($a(i)$)
  for $j = 1$ to $n$
  begin
    check status $st$ of the neighboring node $a(i)^j$
    along dimension $j$;
    if $st = unsafe$ and $a(i)^j$ is not in UNSAFE
      then Add $a(i)^j$ to UNSAFE;
    if |FAULTY| $\geq 2$ or | FAULTY| + |UNSAFE| $\geq 3$
      then Mark the current node $a(i)$ as unsafe;
end.
```

The GLOBAL_STATUS algorithm can be easily extended to a version using decentralized control, where a set of processes, each of which represents a node in a hypercube, cooperate in deciding each node's status.

Based on the analysis of several cases we conjecture that there is no assignment of $n-1$ faulty nodes to a $Q_n$ such that $Q_n$ is an unsafe hypercube. However, there exists an assignment of $n$ faults to an n-dimensional hypercube ($Q_n$) that can make $Q_n$ an unsafe hypercube. For example, suppose n faulty nodes are placed in nodes with exactly one 1 in their addresses, namely, 100..00, 010...00, 001...00, ..., 000..10, 000..01. Let $S_i$ be a set of those nodes with exactly $i$ of 1s in their addresses. It is clear that $S_0 =$ 000...00 is directly connected to all the nodes in $S_1$ and each node in $S_2$ has two neighboring nodes in $S_1$. Therefore all the nodes in both $S_0$ and $S_2$ become unsafe in the first round. Since in general each node in $S_i, 3 \leq i \leq n$, has exactly $i$ neighboring nodes in $S_{i-1}$, in the second step all the nodes in $S_3$ become unsafe. This process continues until the $(n-1)$th step when the nodes in $S_{n-1}$ make all the nodes in $S_n$ unsafe.

The proposed fault-tolerant broadcasting algorithm works in the following way: assume the source node is safe, the first n-2 dimensions in the *cs* are those along which safe nodes are connected. The last two dimensions are selected based on the priority order of neighboring nodes: safe, unsafe, and faulty. Based on the definition of safe node, the last and last but one dimensions are the dimensions along which unsafe nodes could be connected. However only along the last dimension the source node could connect to a faulty node. For the subsequent broadcasting at the other nodes, say $a$, if $a$ is safe then the above procedure will be applied; if $a$ is unsafe then it is responsible to broadcast the data on a subcube with no more than one dimension.

```
Algorithm FT_BROADCASTING
{ Each node keeps a $cs$ : $i_1 i_2 ... i_n$ }
begin
  if $current\_node = source\_node$ then
    for $j = 1$ to $n$  $LABEL[j] := 1$;
  for $j = 1$ to $n$
    if $LABEL[i_j] \neq 0$ then
      if the neighboring node along dimension $i_j$ is
        not in FAULTY or UNSAFE then
        begin
        $LABEL[i_j] := 0$;
        send the broadcast data and LABEL via link $i_j$;
        end;
  for $j = 1$ to $n$
    if $LABEL[i_j] \neq 0$ then
      if the neighboring node along dimension
        $i_j$ is UNSAFE then
        begin
        $LABEL[i_j] := 0$;
        send the broadcast data and LABEL via link $i_j$.
        end;
end.
```

In the above algorithm, *cs* can be global in the sense that each node keeps the same copy. This also implies *cs* being static. Each node also can use a local *cs* which can be dynamically changed to meet certain criteria, such as to balance the traffic load. We also assume

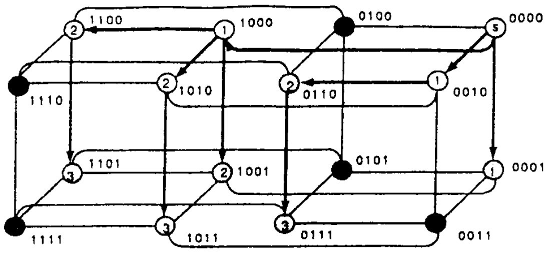

Figure 3: A broadcasting from node 0000 on a $Q_4$ with five faulty nodes.

that a broadcast is initiated when the global status is stable and this status remains stable during one broadcasting period. The enforcement of such assumption [5], [12] is beyond the scope of this paper.

Figure 3 shows how the FT_BROADCASTING algorithm works on a $Q_4$ with five faulty nodes: 0100, 0101, 0011, 1110, and 1111. By applying GLOBAL_STATUS, we derive that $Q_4$ is a safe cube with 0000, 0010, 1000, and 1010 being the safes nodes among the non-faulty nodes. Suppose 0000 is the source node with a LABEL = [1111]. Among all the neighboring nodes, only 0010 and 1000 are safe, therefore a 3-cube and a 2-cube will be sent to these two nodes. Assume that 4 is before 2 in the cs, then 1000 receives the 3-cube $1{*}{*}{*}$, which is represented by the LABEL= [0111] and 0010 receives the 2-cube $0*1*$ represented by LABEL = [0101]. The unsafe node 0001 will receive a 1-cube $0*01$ represented by a LABEL = [0100]. The faulty node 0100 will never receive any data. In the next step, 0010 sends $011*$, represented by [0001] to 0110 and the remaining 0-cube 0011 will be discarded; 1000 sends $1*1*$, represented by a LABEL= [0101] to 1010, $110*$ by [0001] to 1100 (suppose in the *cs* 3 is before 1), and 1001 by [0000] to 1001. In the third step, 0-cubes represented by [0000] will be sent from 0011 to 0111, from 1100 to 1101, and the 1-cube $1*11$ represented by [0100] will be sent from 1010 to 1011. In the last step, since 1111 is faulty no data will be sent to it. Assume that each message transmission takes one time unit. The total broadcasting time for the above example is 3.

**Theorem 1** Algorithm FT_BROADCASTING broadcasts data from a safe node to all the non-faulty nodes optimally, or the data is sent to each node following a Hamming distance path.

*Proof:* We prove the theorem by induction on the size n of the hypercube.

When $n = 0$ and $n = 1$ the theorem clearly holds. Assume the theorem holds for all $n < k$, when $n = k + 1$ and the source is safe. Based on the definition of safe node there are at least $n - 2$ safe neighboring nodes. The $n-2$ subcubes from dimensions $k-1$ to 2 will be sent to these nodes. Based on this assumption, the broadcast data will be transmitted to nodes in these subcubes optimally. The remaining 1-cube will be sent to a safe or unsafe neighbor, and the 0-cube will be sent to the last neighbor if it is not faulty, otherwise it will be discarded. The neighboring node $a$ that receives the 1-cube will split this cube into two nodes, one is itself and the other is $a^i$ the neighbor of $a$ along dimension $i$. If $a^i$ is non-faulty then it will receive the broadcast data from $a$, otherwise no data will be transmitted. Clearly all the nodes in those 0-cube and 1-cube broadcasts which have unsafe root nodes broadcast the data following shortest paths or Hamming distance paths. □

**Corollary :** The number of time steps used in FT_BROADCASTING for a safe node will be no more than $n$, where $n$ is the dimension of the hypercube.

In general, FT_BROADCASTING generates an *incomplete binomial tree*, which is a subgraph of a binomial tree that contains all the healthy nodes, that is, all those missing nodes (mostly leaves) are faulty nodes. The broadcasting tree in Figure 3 is an incomplete binomial tree with five missing nodes, which correspond to five faulty nodes.

FT_BROADCASTING can be modified to be applied in the case that the source is unsafe and there is a neighboring safe node; then the broadcasting data can be first sent to this safe node after which the data is broadcast with the safe node being the source node. Therefore this algorithm can be directly applied at the neighboring safe node; however, the data should not be sent back to the original source node. More specifically, suppose the source node $s$ is unsafe and there is a safe neighboring node $s^i$ along dimension $i$. First of all, $s$ inquires from $s^i$ the location of a faulty or unsafe node other than $s$ that is adjacent to $s^i$ (there is at most one such a node). Assume that $j$ ($j \neq i$) is the dimension along which a faulty or unsafe node is adjacent to $s^i$ (in the case that there is no such $j$ then it will be randomly chosen at $s$), $s$ sends the data to $s^i$ and $s^j$. Then $s^i$ will be the source node to broadcast the data. The cs associated with $s^i$ is selected such that dimensions $i$ and $j$ are placed as the last two dimensions in the sequence. Moreover, the data will not be sent back to $s$. Clearly, the broadcasting in this case requires a

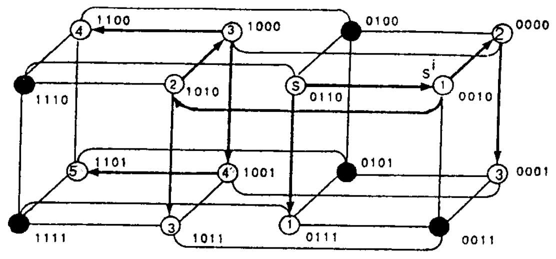

Figure 4: A broadcasting from unsafe node 0110 on the $Q_4$ of Figure 3

maximum of $n+1$ steps, not including the inquiry step before broadcasting. Suppose the unsafe node $s = 0110$ is the source node in the $Q_4$ in Figure 3. First of all, node 0110 obtains from $s^i = (0110)^3 = 0010$ the location of 0010's faulty or unsafe neighbor, which is 0011 in his case, connected along dimension $j = 1$. Then 0110 sends the broadcast data to $(0110)^j = 0111$ and to the safe neighbor 0010. The $Q_4$ is partitioned at 0010 into $\{ 1{*}{*}{*}, 0{*}0{*}, 011{*}, 0011 \}$ and 0010 will be the source node for broadcasting the data. However, the data will not be sent back to the original source node 0110 since the broadcasting in subcube $011*$ has already been completed. The corresponding broadcasting tree is shown in Figure 4 and five steps are used to complete the broadcasting.

**Theorem 2:** When the source node is unsafe and there is at least one safe neighbor, the above modified FT_BROADCASTING algorithm requires a maximum of $n+1$ steps to broadcast data in an $n$-dimensional hypercube.

In general broadcasting initiated from unsafe nodes in a safe hypercube is also possible. In fact, it is possible even to broadcast from unsafe nodes in an unsafe hypercube. However, in the latter case the broadcasting algorithm becomes much more complicated. This algorithm can be sketched as follows: The broadcasting data should be routed to one of the nearby safe nodes. Apparently, such path exists as long as there is no network partition and the hypercube is safe, which ensures the existence of at least one safe node. Nodes on the routing path are all unsafe and should be kept in a list, say history, to be sent together with the broadcasting data to the safe node. Then this safe node acts as the source node to initiate a broadcasting to all the other nodes except those in the history list to avoid redundant broadcasting. Clearly, the key issue here is how to find a nearby safe node. *Depth first search* [3] methods could be used in the absence of global network information. However, this modified broadcasting algorithm becomes less efficient with a possible large amount of nodes in the history list. In the extreme case of an unsafe hypercube with no safe nodes, the broadcasting is completed after searching all the possible paths for a safe node which doesn't exist and all the unsafe nodes will be kept in the history list.

To determine the application range of this method the following theorem provides an estimated upper bound in terms of the number of faults it can tolerate together with their probability.

**Theorem 3:** When the number of faulty nodes reaches $2^{n-1}$, where $n$ is the dimension of the hypercube, the percentage of safe hypercube is no less than $\frac{n}{C_{2^n}^{2^{n-1}}}$.

*Proof*: The assignment of $2^{n-1}$ faulty nodes to a $(n-1)$-subcube will generate a safe n-dimensional hypercube. Clearly, there are $n$ such type of assignments by selecting different dimensions to split the hypercube into two $(n-1)$-subcubes. □

## 4 Performance Analysis

We include the following three measurements in the performance analysis of the proposed broadcasting scheme:

1. Feasibility checking at the source node.
2. Percentage of safe hypercube for a given number of faulty nodes.
3. The number of steps (both for the average and the worst cases) required to determine the status of the hypercube (safe or unsafe).

Feasibility checking at the source node determines whether it is applicable to use the proposed broadcasting method. Actually, the checking process can be simply expressed as:

```
{ at the source node }
if s is safe or one of the neighbors is safe
then it is feasible to use the proposed algorithm
else it is unfeasible.
```

The percentage of safe hypercube under a given number of faulty nodes determines the application

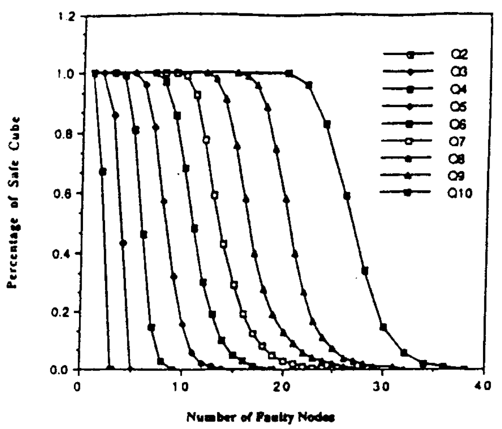

Figure 5: Percentage of safe hypercube vs. number of faulty nodes

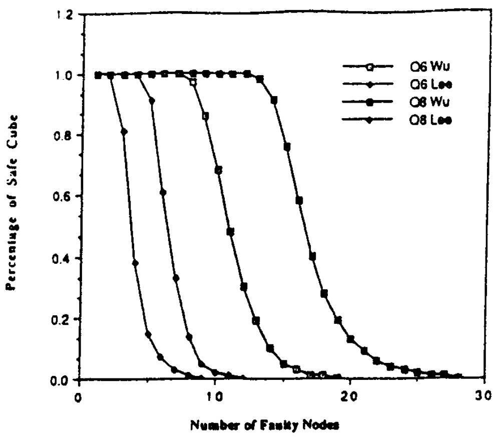

Figure 6: Comparison between two methods on 6 and 8 dimensional hypercubes

range of the algorithm. Figure 5 shows simulation results of the percentage of safe hypercube vs. number of faulty nodes in hypercubes when the dimension varies from 2 to 10. Obviously, one hundred percent of safe hypercube is expected when the number of faulty nodes is less than the dimension of the hypercube. This percentage decreases after the number of faulty nodes exceeds the cube dimension [2]. Simulation results also show a zero percentage of safe hypercube when the number of faulty nodes reaches about $2^{n-1}$, which is half the number of nodes.

Figure 6 shows simulation results which compare the proposed safe cube definition with the one proposed by Lee and Hayes [8] in terms of percentage of safe hypercube on 6-dimensional and 8-dimensional hypercubes. Based on the Lee and Hayes definition of safe hypercube, simulation results show a one hundred percent of safe hypercube when the number of faulty nodes is less than 4 and 5 for 6 and 8 dimensional hypercubes, respectively. These results match the conjectured bound $\lceil \frac{n}{2} \rceil$ [8]. This percentage decreases after the number of faulty nodes exceeds these bounds. The results again confirm the wider range of applications of the proposed safe cube definition compared to the one by Lee and Hayes.

The results in Figure 7 show the average number of steps required to determine the status of hypercubes, safe or unsafe, under different number of faulty nodes when the hypercube dimension varies from 2 to 10. Obviously, one step is required when there is one faulty node or no faulty node. The results show also the average maximum number of steps required for each node to determine the status of a hypercube for a given number of faults under different fault distributions. The number of steps increases as the number of faults increases until it reaches its maximum under a particular number of faulty nodes, 20 for 9-dimensional hypercubes. The number of steps monotonically decreases after further increase of faulty nodes. Clearly when faulty nodes reach $2^n - 1$ only two steps are required. One step is required when all the nodes are faulty. One interesting observation is that the maximum average number of steps is very close to the dimension of the hypercubes.

[2]When the number of faulty nodes reaches n, which is the dimension of the hypercube, the percentage of safe hypercube is still close to one hundred. Actually this percentage can be estimated as no greater than $1 - \frac{2^n}{C^n_{2^n}}$.

## 5 Conclusions

In this paper, we have proposed a broadcasting method for faulty hypercube computers. The method extends the approach proposed by Lee and Hayes where the concept of unsafe node is used. We also show that the method can be applied to all faulty hypercubes with less than $n - 1$ faulty nodes, where n is the dimension of the hypercube. The method can also be applied to some faulty hypercubes with the number of faults ranging from $n$ to $2^n$. A simple feasibility check is also given. It has been proved that when the source node is safe the broadcasting is optimal, namely, the broadcast data is sent to each node via a Hamming distance path. When the source node is unsafe and there is a adjacent safe node then the

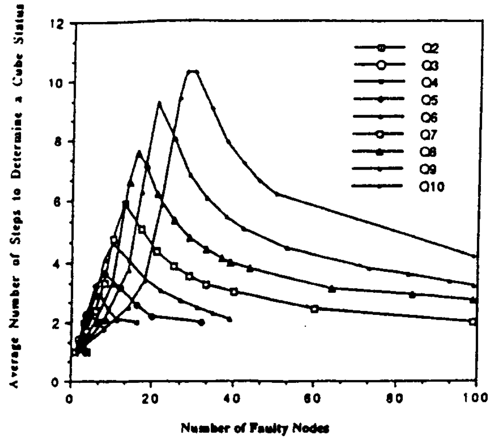

Figure 7: Average number of steps to determine the status of hypercube vs. number of faulty nodes

broadcasting can be achieved with only one more time step than the fault-free case.

## References

[1] A. Al-Dhelaan and B. Bose. Efficient fault tolerant broadcasting algorithm for the hypercube. *Proc. of 4th Conf. on Hypercube Concurrent Computers and Applications.* 1989, 123-128.

[2] D. Bergmark, J. M. Francioni, B. K. Helminen, and D.A. Poplawski. On the performance of the FPS T-series hypercube. *Hypercube Multiprocessors 1987.* M. T. Heath, editor, SIAM, Philadelphia, 1987.

[3] M. S. Chen and K. G. Shin. Depth-first search approach for fault-tolerant routing in hypercube multicomputers. *IEEE Trans. on Parallel and Distributed Systems.* 1, (2), April 1990, 152-159.

[4] NCUBE Company. *NCUBE 6400 Processor Manual.* 1990.

[5] F. Cristian, H. Aghili, R. Strong, and D. Dolev. Atomic broadcasting: from simple diffusion to byzantine agreement. *Proc. of 15th International Conferences on Fault-Tolerant Computing Systems.* 1985, 200-206.

[6] T. Durham and T. Johnson. *Parallel Processing: The Challenge of New Computer Architecture.* Ovum Ltd., London, England, 1986.

[7] S. L. Johnsson and C.-T. Ho. Optimal broadcasting and personalized communication in hypercubes. *IEEE Trans. on Computers.* 38, (9), Sept. 1989, 1249-1268.

[8] T. C. Lee and J. P. Hayes. Routing and broadcasting in faulty hypercube computers. *Proc. 3rd Conf. on Hypercube Concurrent Computers and Applications.* Vol. I, Jan. 1988, 346-354.

[9] Z. Li and J. Wu. A multidestination routing scheme for hypercube multiprocessors. *Proc. 1991 International Conf. on Parallel Processing.* Vol. II, Aug. 1991, 290-291.

[10] L.M. Ni, C.T. King, and P. Prinss. Parallel algorithm design consideration for hypercube multiprocessors. *Proc. of the 1987 International Conf. on Parallel Processing.* Aug. 1987, 717-720.

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

[12] F. B. Schneider. Implementing fault-tolerant services using the state machine approach: A tutorial. *ACM Computing Surveys.* 22, (4), Dec. 1990, 299-319.

[13] C. L. Seitz. The cosmic cube. *Comm. ACM.* 28, (1), Jan. 1985, 22-33.

[14] Y. L Shih and J. Fier. Hypercube systems and key applications. in Parallel Processing for Supercomputers and Artificial Intelligence, edited by K. Hwang and D. DeGroot, McGraw-Hill Publishing Company, 1989, 203-244.

[15] Q.F. Stout and B.A. Wager. Intensive hypercube communication: prearranged communication in link-bound machines. *Journal of Parallel and Distributed Computers*, 1990.

[16] H. Sullivan, T. Bashkow, and D. Klappholz. A large scale, homogeneous, fully distributed parallel machine. *Proc. 4th Annual Symposium on Computer Architecture.* March 1977, 105-124.

[17] J. Wu. Fault-tolerant nonredundent broadcasting in hypercubes. to appear in the Proceedings of the 1992 International Conference on Parallel Processing.

[18] J. Wu and E. B. Fernandez. Broadcasting in faulty cube-connected-cycles with minimum recovery time. to appear in the Proceedings of CONPAR92.