OREGON STATE
UNIVERSITY
COMPUTER
SCIENCE
DEPARTMENT

Efficient Fault Tolerant Broadcasting Algorithm for the Hypercube

Abdullah Al-Dhelaan
Bella Bose

OACIS
&
Department of Computer Science
Oregon State University
Corvallis, Oregon 97331

89-70-1

Efficient Fault Tolerant Broadcasting Algorithm for the Hypercube[1]

Abdullah Al-Dhelaan[2]
Bella Bose

Oregon Advanced Computing Institute (OACIS) and
Department of Computer Science
Oregon State University
Corvallis, OR 97331
(Tel. No. 503-754-3273)

dhelaan@cs.orst.edu
bose@cs.orst.edu

1 This paper is to appear in The Fourth Conference on Hypercube Concurrent Computers and Applications, Monterey, California, March 6-8, 1989.

2 The first author is also supported by King Saud University, Riyadh, saudi Arabia.

Abstract

One of the most popular topologies is the hypercube, that has n = 2^k processors, numbered 0 to 2^k-1 and connected in such a way that there is a link between any two if and only if they differ in one bit. Its popularity is due to the fact that a large number of processors can be interconnected using a small number of communication links while at the same time keeping the communication delay between processors at minimum.

Broadcasting is a procedure by which a processor can pass a message to all other processors in the network non redundantly; it is extremely important for diagnosis of the network, distribution agreement or clock synchronization.

In this paper we present some interesting features and properties of the hypercube and then we describe the algorithm for broadcasting in the hypercube, by Sullivan and Bashkow. Finally, we develop a simple yet efficient algorithm for broadcasting in the hypercube in the presence of some faulty processors.

I. Introduction

Recently, the prospect of interconnecting together many small computers has become more attractive and feasible. Small computers are decreasing in cost and will very likely continue to do so. On the other hand, very large machines remain costly. It may be cost effective to interconnect many small computers to form a large powerful system [1].

One of the most popular interconnection topologies is the hypercube [2,3]. The hypercube is a network of a loosely coupled processors connected in such a way that two processors are linked if and only if their binary representation differ in exactly one bit position. i.e., the indices of neighboring processors differ by a power of 2. It is a network with powerful interconnection features introduced under different names (Cosmic cube [2], Boolean n-cube, n-cube [3], etc.).

Informally, an n-dimensional hypercube, Q_n, is the composition of two (n-1)-dimensional hypercubes with links between corresponding processors in each of them. Fig. 1 and 2 show a Q_3, and a Q_4, hypercubes respectively.

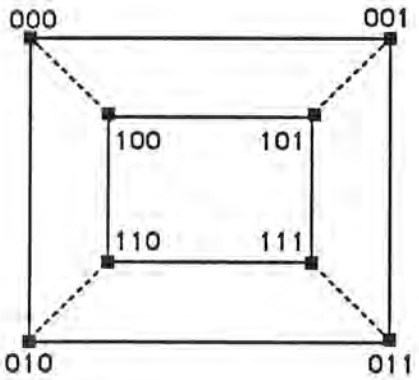

Fig. 1. 3-Dimensional Hypercube, Q_3

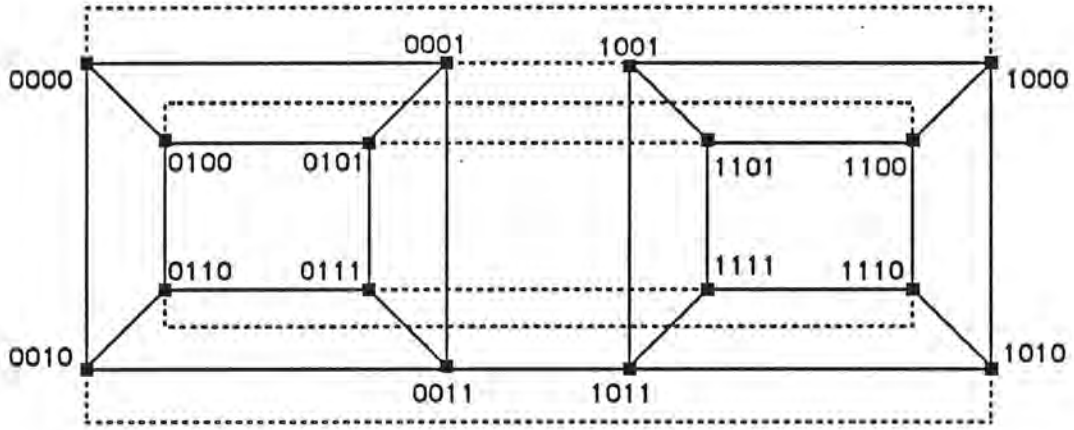

Fig. 2. 4-Dimensional Hypercube, Q_4

The hypercube is known to have the following characterizations :

1. Each processor has a local memory and no shared memory is used.

2. Processors communicate by sending messages direct or through some intermediate processors.

3. Synchronization occurs by the availability of data and messages.

The hypercube gains its popularity due to the fact that a large number of processors can be interconnected using a small number of communication links while at the same time keeping the communication delay between processors at minimum. Furthermore, it has some other attractive features such as:

1. Regularity and high potential for the parallel execution of various algorithms.

2. Its architecture allows high level of concurrency and efficiency.

3. The number of links among processors is small allowing us to have a very large hypercube with a reasonable number of links. At present machines with up to 16384 processors are available [4].

4. The hypercube can be decomposed completely into subhypercubes (hypercubes of smaller dimension).

5. Most other networks can be directly mapped into a hypercube.

Broadcasting is a procedure by which a processor can pass a message to all other processors in the network non redundantly; this message can either be information or control. Broadcasting is extremely important for diagnosis of the network, distribution agreement or clock synchronization.

The problem is that, it is extremely difficult to broadcast in the presence of some faulty processors in the system. There are two different approaches:

1. Each node keeps limited information about the faulty nodes in the system.

2. Send multiple copies of the message through disjoint paths. The nodes that receive the message identify the original message from the multiple copies by using some scheme e.g. majority voting.

Sullivan and Bashkow [5] developed an algorithm for broadcasting in the hypercube. This algorithm was developed on the assumption of having no faulty processors. Ramanathan and Shin [6], developed another one for the hypercube in the presence of faults which uses the second approach mentioned above.

In this paper, we first briefly describe the broadcasting algorithm for the hypercube by Sullivan and Bashkow [5]. This algorithm was developed on the assumption of having no faulty processors. Then we develop a simple yet efficient algorithm for broadcasting in the hypercube in the presence of some faulty processors. Our algorithm uses the first approach where each processor keeps a small amount of information about others.

The rest of the paper is organized as follows: In section II, we introduce some notations that we will need throughout the paper. Since our algorithm is based on Sullivan and Bashkow algorithm [5], their algorithm is briefly described in section III. The main results -broadcasting in the hypercube in the presence of some faulty processors- is given in section IV. The paper concludes with section V.

II. Notations

We start with some definitions and introduce the notations that are needed throughout the paper.

Definition 1:

A n-dimensional hypercube , denoted as n-cube or Q_n, is defined recursively as follows:

1) Q_0 is a trivial graph with one node, and

b) $Q_n = K_2 \times Q_{n-1}$

where K_2 is a complete graph with two nodes and x is the product operation on two graphs [7].

Definition 2:

The ith neighbor of the processor x is the processor y, where x and y differ only in the ith bit.

Example 1:

In a hypercube with eight nodes, i.e. Q_3, the 0th, 1st or 2nd neighbors of node 000, are nodes 001, 010 or 100 respectively.

Definition 3:

The ith link of a given processor with address x is the link connecting x to its ith neighbor.

Example 2:

For the processor 000, in a hypercube with eight processors; the 0th, 1st and 2nd link are these connecting to processor 001, 010 and 100 respectively.

Definition 4:

The broadcasting tree is defined to be the tree rooted at the source node and whose edges are the arcs used to broadcast the message. Every branch of this tree is also a smaller broadcasting tree.

We refer to the processors of a multiprocessor as nodes, and the communication links connecting these processors as links. The processors communicate by sending messages over the links and this could be direct or

indirect, i.e. through some intermediate processors. In our algorithm, we use two types of weights to control the flow of the message in the system:

- Singular, consisting of a single value like 1, 2 or 3.

- Paired, consisting of a pair of two values, like (1,2) or (0,1). The first element is the actual weight used to control the flow similar to Sullivan- Bashkow algorithm; the second element is used for re-routing the message to the otherwise unreached processors.

We define the procedure Send (message,i,j) to mean send the message with the weight i to your jth neighbor.

Example 3:

Consider a hypercube of 3-dimension with eight nodes. The processor 000 can send a message to its 2nd neighbor, i.e. processor 100, via the 2nd link with weight 1 by executing:

Send (message,1,2)

In all the figures in this paper the labels on the links represent the weights which are sent on these links. In an n-dimensional hypercube, Q_n, we define n to mean the dimension and $N = 2^n$ to mean the number of nodes. Also for the broadcasting tree T we define N(T) to be the number of nodes in such tree.

III. Broadcasting in the Hypercube

Sullivan and Bashkow have devised an algorithm for broadcasting in the hypercube [5]. This algorithm sends the message to all other nodes nonredundantly, meaning that the broadcasted message is sent to each processor exactly once. The algorithm takes $\log_2(N)$ steps to broadcast the message. It works by sending a weight along with each message; this weight, is used to decide how the algorithm should continue broadcasting the message from the receiving node. The algorithm starts at the source node using the steps in Fig. 3.

```
Generate the message
FOR i = 0 TO ( log2(N) -1) DO
    Send (message, i, i)
```

Fig. 3. The steps for the source processor in the hypercube

Afterwards, each processor needs to follow the steps that are shown in Fig. 4.

```
Extract the message and process locally
FOR j = 0 TO (weight -1) DO
    Send (message, j, j)
```

Fig. 4. The steps for other processors in the hypercube

The route that the broadcasted message will take can be shown using a tree where the nodes and arcs of the tree correspond to the nodes and links of the hypercube respectively. Furthermore, the root of the tree represents the source, i.e. originator of the broadcasted message.

The following examples 4 and 5, illustrate how this algorithm for broadcasting in a hypercube with no faulty processors works.

Example 4:

Consider broadcasting in a 3-dimensional hypercube, Q_3. Fig. 5 shows how a message would be broadcasted from node 011 to all other processors in the network. It is interesting to note that all the nodes in the rightmost, middle and leftmost subtrees have addresses that differ in bit 0, bit 1 and bit 2 respectively from the originator's.

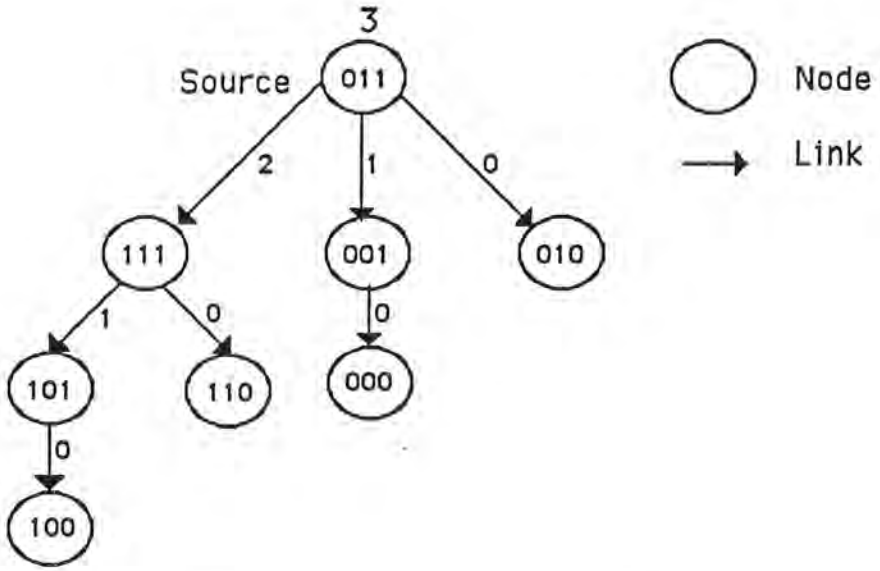

Fig. 5. Broadcasting in a Q_3 from node 011.

Example 5:

Fig. 6 shows the broadcasting of a message from node 101 to all other processors in the hypercube with eight nodes. Note that the numbers on the links represents the weights sent on these links.

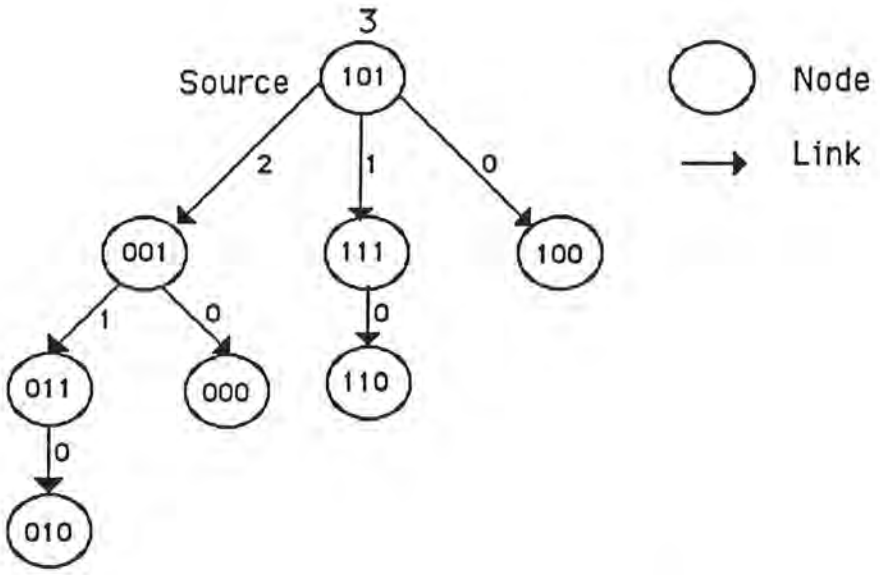

Fig. 6. Broadcasting in a Q_3 from node 101.

We next prove some lemmas that are needed to prove that the algorithm sends the message to each node in the system nonredundantly thus is correct.

Lemma 1:

The broadcasting tree for a Q_n hypercube contains 2^n nodes.

Proof:

Consider the broadcasting tree for a hypercube with 2^n nodes, i.e. Q_n. The source node will connect to n other processors. Let us call the subtree whose root is the ith neighbor of the source T_i. It is clear that:

- All the nodes in T_0 differ only in bit 0 from the source so $N(T_0) = 1$.

- All the nodes in T_i differ in bit i and may be bit $j < i$. So $N(T_i) = 1X2^i$.

The number of nodes in the broadcasting tree $= 1 + \sum_{i=0}^{n-1} N(T_i)$

$$= 1 + \sum_{i=0}^{n-1} 2^i = 2^n$$

So the number of nodes in the broadcasting tree is equal to 2^n nodes.

Lemma 2:

The nodes in the broadcasting tree are distinct.

Proof:

Using lemma 1 we know that the size of the broadcasting tree is always a power of 2. We can proof that these nodes are distinct by using mathematical induction on the number of nodes in the broadcasting tree.

1) Basis: It is easy to see that the lemma holds for n = 0, 1 or 2.

2) Inductive step: Assume that the nodes in any broadcasting tree with k nodes or less are distinct.

3) In a broadcasting tree with 2k nodes it is easy to see that the nodes in each subtree T_i differ in the ith bit from the source. So these subtrees are disjoint. Since each of these subtrees is with k nodes or less and they are disjoint. All the nodes in the broadcasting tree are disjoint.

Theorem 3:

The algorithm sends the message to all nodes in the system nonredundantly.

Proof:

The fact that the algorithm sends the message to all nodes in the hypercube nonredundantly can easily be deduced from the following facts:

a) In the broadcasting tree, each node receives the message from exactly one node, its father.

b) The maximum number of nodes the Q_n hypercube can have is 2^n nodes.

c) The broadcasting tree for a Q_n hypercube has 2^n nodes, from lemma 1.

d) All of the nodes in the broadcasting tree are distinct, from lemma 2.

IV. Broadcasting in the presence of faulty processors

In this section we introduce a new broadcasting algorithm that tolerate the existence of some faulty processors. The algorithm works by sending a weight with the message, just like that in the previous section. When it finds a faulty processor, it sends the message to all of the faulty processor descendents through other lower neighbors.

We give two example 3 and 4 then explain how the algorithm works. Finally, we give the formal algorithm.

Example 3:

Fig. 7. shows how a message would be broadcasted from node 011 in a 3-dimensional hypercube, Q_3, where node 111 is faulty.

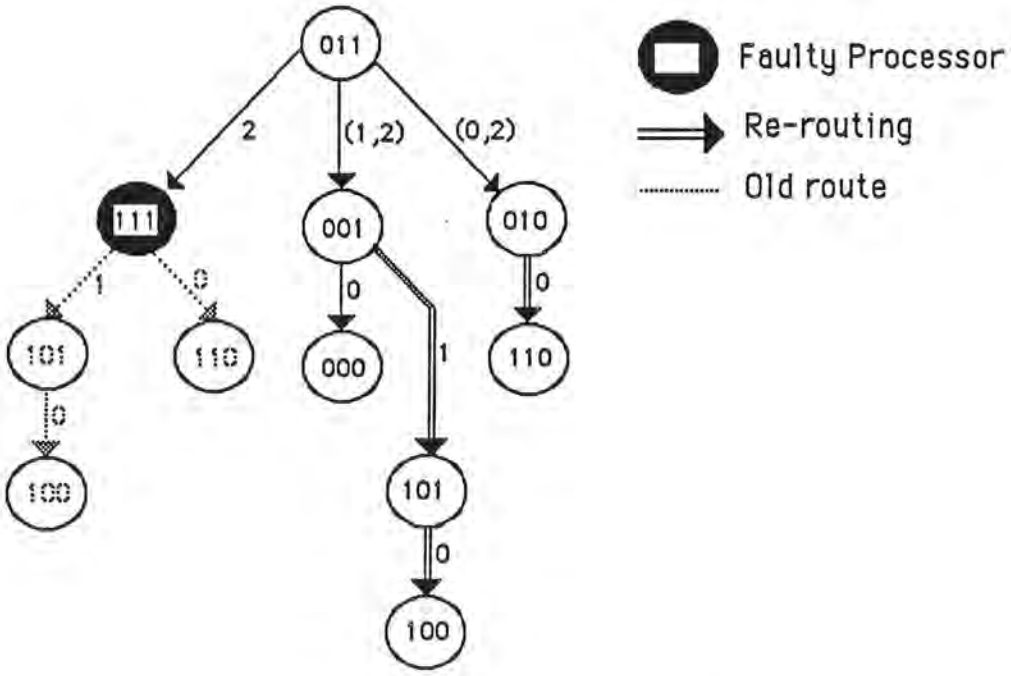

Fig. 7. Broadcasting in the presence of a single faulty processor.

Because the node 111 is faulty, all of its descendent nodes 110, 101 and 100, will not receive the message. Since node 111 is the 2nd neighbor of node 011, the node 011 will re-route the message to these unreached nodes by sending a paired weight to its 0th and 1st neighbors, i.e. nodes 010 and 001 respectively.

When any node receive a paired weight (i,j) it interprets it as:

1. Take i as the weight and do what you would usually do.

2. Send the message with a singular weight of i to your jth neighbor. i.e. execute Send (message,i,j).

Example 4:

Fig. 8 shows how a message would be broadcasted from node 101 in a 3-dimensional hypercube, Q_3, where nodes 111 and 011 is faulty.

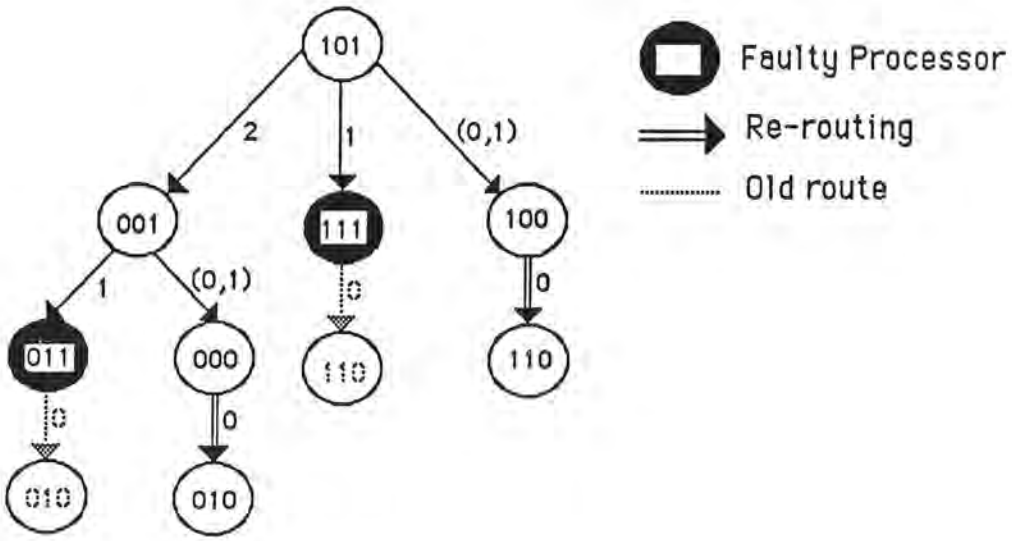

Fig. 8. Broadcasting in the presence of a single faulty processor.

The main idea is that, for any node the descendants of its ith neighbor are all nodes that differ from the node in the ith bit and possibly any jth bit where $j < i$. So we can send the message to all lower ranked neighbors with paired message and then to their ith neighbors with the appropriate singular weight.

The broadcasting tree is normally higher towards high ranked neighbors, since the ith neighbor has more descendant than the jth neighbor for all $i > j$. The algorithm will send the message through all neighbors with lower rank than the faulty processor. This is very important because we need the height of the tree to be as smaller as possible as this will increase the efficiency of the algorithm and effect the number of links the message needs to travel through. We give the formal algorithm next.

The Algorithm:

The algorithm starts at the source node using the steps in Fig. 9.

```
Generate the message
FOR j = 0 TO (log2(N) -1) DO
    IF for some i > j the ith neighbor is faulty
      THEN Send (message, j, (j,i))·
      ELSE Send (message, j, j).
```

Fig. 9. The steps used by the source processor.

then each processor needs to follow the steps that are shown in Fig. 10.

```
Extract the message and process locally
IF the node weight is paired
   THEN BEGIN
            Let the paired weight be (a,b)
            Send (message, b, a).
            Set weight to a.
        END
FOR j = 0 TO (weight -1) DO
   IF for some i > j the ith neighbor is faulty
     THEN Send (message, j, (j,i)).
    ELSE Send (message, j, j).
```

Fig. 10. The steps used by each processor.

The algorithm works if there is only one single faulty processor or if the fault degree of each node is at least n-1. In other words, each non-faulty processor is connected to at most one faulty processors in the network.

We introduce a basic lemma then prove that the algorithm does what it is intended to do.

Lemma 4:

In the broadcasting tree, any faulty node with weight w will disconnect 2^w-1 nonfaulty nodes.

Proof:

The tree that contains the faulty node and all of its descendents is equivalent to a broadcasting tree for a Q_w hypercube; thus should have 2^w nodes according to lemma 1. One of these nodes is the faulty node so the number of isolated nodes will be 2^w-1 nodes.

Theorem 5:

Our broadcasting algorithm tolerate the existence of some faulty nodes and broadcast the message to all of the nonfaulty nodes in the system nonredundantly.

Proof:

Consider broadcasting in a hypercube with 2^n nodes. Fig. 11 shows a general fragment of the broadcasting tree with a faulty node.

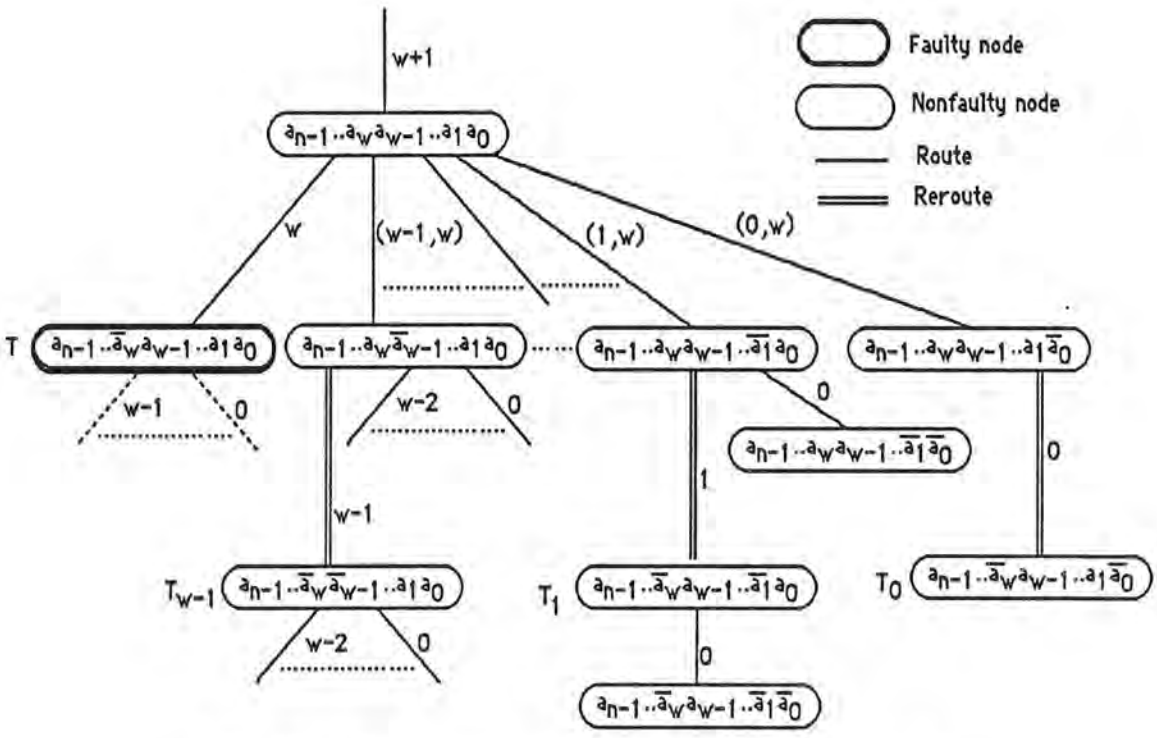

Fig. 11. A fragment of the broadcasting tree with faulty node

When the node $a_{n-1}..a_w a_{w-1}..a_1 a_0$ finds that its wth neighbor is faulty it causes the smaller broadcasting trees $T_0,....,T_{w-1}$ to be generated and attached to the original broadcasting tree that is used by the algorithm in the previous section.

The trees $T_0,...,T_{w-1}$ are also broadcasting trees, so by lemmas 1 and 2 each of them has 2^i distinct nodes.

$$\text{The number of node in the } T_i \text{ trees} = \sum_{i=0}^{w-1} N(T_i)$$

$$= \sum_{i=0}^{w-1} 2^i = 2^w - 1 \text{ nodes}$$

Lemma 4 tells us that this equal to the number of isolated nodes; thus our algorithm delivers the message to all the otherwise isolated nodes.

V. Conclusion:

Broadcasting is extremely important for diagnosis of the network, distribution agreement or clock synchronization. In this paper, we elaborated on the hypercube topology and mentioned some of its interesting features and characteristic. Then we briefly described Sullivan-Bashkow algorithm for broadcasting in the hypercube with no faulty nodes. Finally, we developed an efficient fault tolerant broadcasting algorithm for the hypercube that tolerates the existence of some faulty nodes. In our algorithm each node keeps limited information about its faulty neighbors in the system.

References:

[1] R.A. Finkel and M. H. Solomon, "Processor Interconnection Strategies," IEEE Transactions on Computers, C-29, May 1980, pp. 360-371.

[2] C. L. Seitz, "The cosmic cube," Commun. Ass. Comput. Mach., vol. 28, Jan. 1985, pp. 22-33.

[3] M. Pease, "The Indirect Binary n-Cube Microprocessor Array," IEEE Transactions on Computers, C-26, May. 1977, pp. 458-473.

[4] R. M. Chamberlain, "Gray codes, Fast Fourier Transformations and Hypercubes," Parallel Computing, 6, 1988, pp. 225-233.

[5] H. Sullivan and T.R. Bashkow, "A large scale homogeneous, fully distributed parallel machine, I," in Proc. Fourth Symp. Comput. Architecture, Mar. 1977, pp. 105-117.

[6] P. Ramanathan and K. Shin, "Reliable Broadcast in Hypercube Multicomputers," IEEE Transactions on Computers, Dec. 1988, pp. 1654-1657.

[7] N. Deo, Graph Theroy with applications to Engineering and Computer Science, Prentice-Hall, 1974.