Broadcasting on Uni-directional Hypercubes

Huang-Ming Huang & Chang-Biau Yang
Department of Applied Mathematics
National Sun Yat-sen University
Kaohsiung, Taiwan 804
Republic of China

Abstract

In this paper, we solve the broadcasting problem for the even dimensional uni-directional hypercube (UHC). In the constant evaluation model, the complexity of one of our all-port broadcasting trees, is $n+1$, and it is optimal. Whereas the best one of our one-port broadcasting trees needs $\frac{4}{3}(n-n \bmod 6)+\frac{3}{2}(n \bmod 6)$ steps. These algorithms can be extended to solve the odd dimensional case. We also propose an all-port fault-tolerant broadcasting tree whose height is $\frac{3}{2}n+\frac{n \bmod 4}{2}$.

1 Preliminary

An n-dimensional hypercube can be modeled as a graph $Q_n(V,E)$ with 2^n nodes and $n2^{n-1}$ edges. Each node represents a processor (or processing unit) and each edge represents a communication link between a pair of processors. The nodes are assigned with binary numbers from 0 to 2^n-1 such that identifiers of any two neighbors differ only in one bit position. The port i of a node v stands for the link which connects v with the node whose identifier differs from v in the ith bit. Note that the rightmost bit position is 0.

The uni-directional hypercube (UHC)[2] is an orientation of a hypercube; that is, each edge in a hypercube is assigned with a fixed direction, either incoming or outgoing. The directions of the edges are determined according to the following polarity function:

$$polarity(B,i) = \text{sign of } (-1)^{(b_{n-1}+b_{n-2}+\cdots+b_1+b_0+i)},$$

where $B=(b_{n-1}b_{n-2}\cdots b_1b_0)$ is the binary representation of the identifier of a node in a UHC, and i represents the port number.

If the polarity function is positive, port i is an out-port of node B; otherwise port i is an in-port of node B. *Out-port* (*in-port*) i of a node v may be either port $2i$ or port $2i+1$, depending on which one is the out-port (in-port). For example, the graph shown in

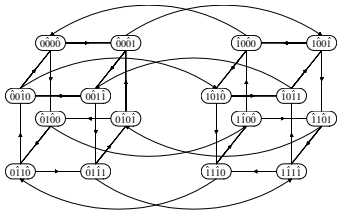

Figure 1: The 4-dimensional uni-direction hypercube.

Figure 1 is the 4-dimensional uni-direction hypercube and the link from 0000 to 0100 is out-port 1 (port 2) of 0000.

For convenience, we will denote a node with hat symbols to represent its out-port, such as $\hat{0}0\hat{0}\hat{0}$ which means ports 1 and 3 are the out-ports of 0000. The superdimension j of a node v, denoted as $SD_j(v)$, represents bits $2j$ and $2j+1$ of v together with their polarity. For instance, $SD_2(0\hat{0}1\hat{0}0\hat{0}1\hat{0}) = 1\hat{0}$. Sometimes, we denote a node S as $\langle s_{\frac{n}{2}-1}, s_{\frac{n}{2}-2}, \cdots, s_j, \cdots, s_1, s_0\rangle$, where s_j is superdimension j of node S and is represented by two bits. $\bar{s}_j$ represents the result by complementing the out-port bit of s_j and exchanging the polarity of these two bits, such as $\bar{s}_j = 1\hat{0}$ if $s_j = \hat{0}0$.

The UHC discussed above is also referred as the positive UHC. By graph duality, there is another class of uni-directional hypercubes (referred as the negative UHC) with the opposite polarity function:

$$polarty(B,i) = \text{sign of } (-1)^{(b_{n-1}+b_{n-2}+\cdots+b_1+b_0+i+1)}.$$

In fact, most properties of the positive UHC and the negative UHC do not make any difference. Therefore, we only refer UHC as positive UHC in the rest of this paper except particularly pointed out.

Although the degree of the UHC equals only a half of the hypercube's degree, the diameter of the UHC is only $n+1$ when n is even, and $n+2$ when n is odd. An n-UHC can be decomposed into a positive $(n-1)$-UHC and a negative $(n-1)$-UHC[2].

While a processor can communicate with all its ports concurrently, we call it *all-port* communication. If a processor can only send and receive on one of its ports at any time, and the ports on which a processor sends and receives can be different[7] , it is called *one-port* communication.

The evaluation of communication cost is also an important issue. In this paper, we adopt the model proposed by Fraigniaud [4, 5]. The time T_{OTO} required for sending a message from one processor to one of its neighboring processors is assumed to the sum of a start-up time β and a propagation time $L\tau$ proportional to the length of message L ($1/\tau$ is the bandwidth), i.e. $T_{OTO} = \beta + L\tau$. This evaluation model is said to be linear. If $T_{OTO} = 1$, it is called the constant evaluation model. The constant evaluation model is widely used in most published papers. The authors of those papers, however, did not consider the length of message would play an important role in the communication cost. If the length of message is long, it may dominate the communication cost.

This paper is organized as follows: In section 2, we prove that the even dimensional UHC is node symmetric. The broadcasting trees are described in section 3 to 5. Next, we present our all-port fault-tolerant broadcasting tree in section 6. In section 7, we analyze the performances of our algorithms. Some concluding remarks and unsettled problem will be given in section 8.

2 Node Symmetric Property of the UHC

A graph is node symmetric if given any two nodes v_1 and v_2, there exists an automorphism θ such that $\theta(v_1) = v_2$[1]. The node symmetric property is very important for an interconnection network. If a graph is node symmetric, then it looks the same from every node in the graph. Thus, the routing and broadcasting algorithms for the network will not have to be modified for the change of the starting node. In fact, an n-UHC is node symmetric only if n is even.

Theorem 1 *An even dimensional UHC is a node symmetric graph.*

Beside the node symmetric property, there is still another property which is important for our subsequent algorithms.

Theorem 2 *In an even dimensional positive UHC, if we interchange bits $2i$ and $2i+1$ along with their polarity in all node identifiers, for all $0 \leq i < \frac{n}{2}$; in other words, each node $(\hat{b}_{n-1}b_{n-2}\cdots\hat{b}_{2i+1}b_{2i}\cdots\hat{b}_1b_0)$ becomes $(b_{n-2}\hat{b}_{n-1}\cdots b_{2i}\hat{b}_{2i+1}\cdots b_0\hat{b}_1)$ and each node $(b_{n-1}\hat{b}_{n-2}\cdots b_{2i+1}\hat{b}_{2i}\cdots b_1\hat{b}_0)$ becomes $(\hat{b}_{n-2}b_{n-1}\cdots\hat{b}_{2i}b_{2i+1}\cdots\hat{b}_0b_1)$, the resulting graph will become an even dimensional negative UHC.*

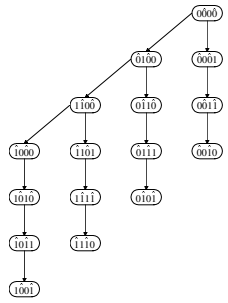

Figure 2: The 4-BT$_1$ rooted at $(0\hat{0}0\hat{0})$.

3 Broadcasting Trees

The divide-and-conquer strategy is widely used to construct broadcasting trees in most interconnection networks. There is no exception to the UHC. Therefore, we divide an n-UHC into four $(n-2)$-UHCs. The data in the source node is transmitted first to one node in each $(n-2)$-UHC, and this node becomes a source node in the $(n-2)$-UHC. Then this method is recursively applied. The node which first receives message from the source node in each $(n-2)$-UHC is called a *sub-root*. Without loss of generality, we use the leftmost two bits of identifiers to divide an n-UHC. The sub-roots in the four $(n-2)$-UHCs are identified as 00, 10, 11 and 01, which represents the difference in the leftmost two bits between the sub-roots and the source node. Because the source node are embedded in a cycle of length four via its ports $n-1$ and $n-2$ and each node in the cycle is in a distinct $(n-2)$-UHC, the message can be broadcast to the nodes in the cycle via the leftmost out-ports of these nodes. To be precise, the message is sent through the path $(0\hat{0}0\cdots0) \rightarrow (\hat{0}10\cdots0) \rightarrow (1\hat{1}0\cdots0) \rightarrow (\hat{1}00\cdots0)$. Thus we achieve our goal to transmit message to one node in each $(n-2)$-UHC. And the tree, denoted as n-BT$_1$, is recursively constructed. For example, the graph in Figure 2 is the 4-BT$_1$ rooted at $0\hat{0}0\hat{0}$.

Instead of using the cycle mentioned above to transmit message to the other three $(n-2)$-UHCs (except the $(n-2)$-UHC the source node is in), we can also use two paths to do so. The source $(0\hat{0}0\hat{0}0\cdots0)$ sends the message via two paths $(0\hat{0}0\hat{0}0\cdots0) \rightarrow (\hat{0}1\hat{0}00\cdots0) \rightarrow (1\hat{1}0\hat{0}0\cdots0)$ and $(0\hat{0}0\hat{0}0\cdots0) \rightarrow (\hat{0}0\hat{0}10\cdots0) \rightarrow (1\hat{0}0\hat{1}0\cdots0)$. The subcube $10x\cdots x$ does not receive message from $(11000\cdots0)$ any more, it receives mes-

sage from $(00010\cdots0)$ instead. This broadcasting tree of the 6-UHC is shown in Figure 3. In this paper, we denote this tree as n-BT_2.

Theorem 3 *The height of the n-BT_1 is $\frac{3}{2}n$ and that of the n-BT_2 is $n+1$, where n is even.*

For the all-port communication, the start-up time of the communication cost is bounded by the tree height h since messages have to be routed through at least h links from the root to the leaf nodes of the broadcasting tree. Consequently, the broadcasting on the n-UHC under BT_1 and BT_2 would take $\frac{3}{2}n$ and $n+1$ start-up time steps respectively.

Theorem 4 *For the all-port communication in the n-UHC, the broadcasting time using BT_1 and BT_2 are bounded by $\frac{3}{2}n$ and $n+1$ respectively, where n is even. For the one-port communication, the broadcasting time using BT_1 and BT_2 is bounded by $\frac{3}{2}n$.*

Because the diameter of the n-UHC is $n+1$, it is impossible to construct a tree with height less than $n+1$. Thus, for the constant evaluation model, our broadcasting algorithm under BT_2 is optimal. From Theorem 4, we may doubt if the lower bound of start-up time of one-port broadcasting in the n-UHC is $\frac{3}{2}n$. The answer is no. For example, it needs 8 steps to broadcast if we prune node (101111) in 6-BT_2 and glue it as a child of (111111). As a result, we develop another type of broadcasting tree, BT_3, which is based on the tree we mentioned above. Instead of dividing an n-UHC into four $(n-2)$-UHCs as in BT_1 and BT_2, we now divide an n-UHC into 64 $(n-6)$-UHCs. The root broadcasts the message to the sub-roots using the tree, referred as 6-BT_3. 6-BT_3 is similar to 6-BT_2 except (101111) is no longer a child of (101110) but of node (111111). If each subcube can not be further divided into 64 units (i.e. each subcube contains less than 64 nodes), we use the BT_1 to broadcast to the remaining nodes.

Theorem 5 *One-port broadcasting in the n-UHC using BT_3 needs $\frac{4}{3}(n-(n \bmod 6))+\frac{3}{2}(n \bmod 6)$ steps for start-up time, where n is even.*

4 Construction Algorithms for the Broadcasting Trees

In the section, we shall present the distributed construction algorithms for the broadcasting trees we proposed in the previous section.

Compared with BT_2 and BT_3, BT_1 is a very simple structure. We can easily find a children function which generates all children of a node in BT_1. Without loss of generality, we assume the root is the node with all bits being zero. The children of i is given in Figure 4.

In the function, l is the first superdimension, scanning from right to left, that is not $\hat{0}0$ or $0\hat{0}$. Note that the rightmost superdimension is superdimension 0.

For BT_2, it is much more complicated. We do not provide the children function here, because the algorithm using the children function is not so easy . At first, we shall describe some characteristics of BT_2.

Proposition 1 *In BT_2, there does not exist a path $v_0e_0v_1e_1\cdots e_{k-1}v_k$ such that e_i, e_{i+1} and e_{i+2} are out-port j of v_i, v_{i+1} and v_{i+2} respectively, and $j \neq 0$.*

Proposition 2 *In BT_2, if a node receives a message, which its parent receives from in-port j, from in-port i and $j > i+1$, then the node should send the message through all out-ports with out-port numbers smaller than $i+2$.*

Proposition 3 *In BT_2, a node k is a leaf if and only if the last three arcs of the path from root to node k are in-port 0 of the last three nodes.*

As we can see from the above propositions, they are just to follow the philosophy of BT_2 construction method. Messages in an n-UHC should be sent to each (n-2)-UHCs in two steps except those in 2-UHCs. Hence a message can be broadcast with a tag which informs the children where the message was from according to the above propositions. The algorithm is as follows:

Algorithm 1 *Distributed BT_2 construction algorithm*

```
if the current node is the source node then
    send message with tag n to all out-ports;
else
    receive a broadcasting message from in-port i;
    if tag > i + 1 then
        send message with tag i to out-port j, ∀0 ≤
        j ≤ i + 1;
    else if tag = i = 0 then
        send message with tag −1 to out-port 0;
    else if tag = i then
        send message with tag i to out-port j, ∀0 ≤
        j ≤ i − 1;
    else if tag = i + 1
        send message with tag i to out-port j, ∀0 ≤
        j ≤ i;
    else if tag = −1
        stop;
    end;
```

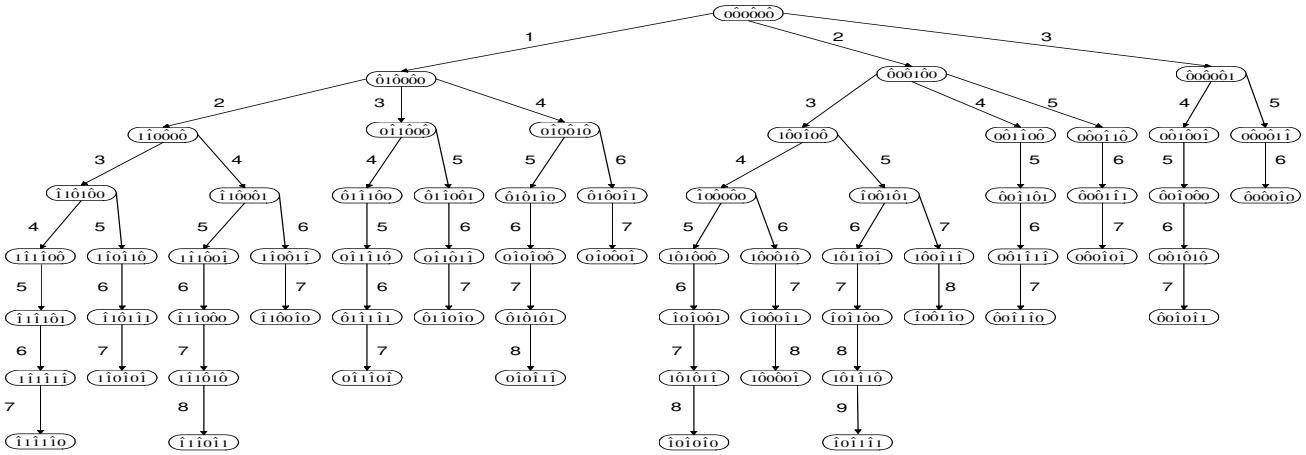

Figure 3: The 6-BT_2 rooted at $(0\hat{0}0\hat{0}0\hat{0})$.

$$children(i) = \begin{cases} \langle i_{\frac{n}{2}-1}, \cdots, \bar{i}_k, \cdots, i_0 \rangle, \forall 0 \le k \le \frac{n}{2}-1 & \text{if } i \text{ is root,} \\ \langle i_{\frac{n}{2}-1}, \cdots, \bar{i}_k, \cdots, i_0 \rangle, \forall 0 \le k \le l & \text{if } i_l \in \{\hat{0}1, 1\hat{0}, \hat{1}1, 1\hat{1}\} \text{ ,} \\ \langle i_{\frac{n}{2}-1}, \cdots, \bar{i}_k, \cdots, i_0 \rangle, \forall 0 \le k < l & \text{if } i_l \in \{\hat{1}0, 0\hat{1}\} \text{ and } l \neq 0, \\ \emptyset & \text{if } i_l \in \{\hat{1}0, 0\hat{1}\} \text{ and } l = 0, \end{cases}$$

Figure 4: The children function of n-BT_2.

end;
end

Note that in this algorithm, the identifier of the source node is no longer needed. The tag, in fact, takes the place of the source identifier. Therefore, this algorithm does not need any extra field encapsulated in the message, compared with the broadcasting algorithm in hypercubes.

Since the n-BT_3 is based on the 6-BT_3, we split the binary representation of a node into several sections, each containing 6 bits except the rightmost section, i.e. section 0. Again we suppose the root is $(0 \cdots 0)$. Each node contains a table which describes the basic information of a 6-BT_3. Given a node in 6-BT_3, the table suggests to which out-port the node should begin sending message. If the current node is the root, it sends the message via all its out-ports. If a node receives a message from other node, the algorithm for this node is as follows:

Algorithm 2 *Distributed BT_3 construction algorithm*

receive a message from in-port i;
scan the binary representation from right to left, find the first section k whose content is not zero;
if $(k \neq 0)$ or $(n \bmod 6 = 0)$ then
    look up the table using the content of section k to find out to which out-port j it should start sending message;
    if $k \neq 0$ then
        send the message to the out-port l, $\forall 0 \le l \le j + 3(k-1) + (n \bmod 6)/2$;
    else
        send the message to the out-port l, $\forall 0 \le l \le j + 3k$;
else
    use the content of section k to send the message according to the algorithm of BT_1;
end;
end

When we consult the table,the polarity should be consistent. For instance, if the first section whose content is not zero is $\hat{1}0\hat{0}1\hat{0}0$, then $0\hat{1}1\hat{0}0\hat{0}$ rather than $1\hat{0}0\hat{1}0\hat{0}$ should be used to look up the table.

5 Broadcasting in the Odd Dimensional UHC

In the previous sections, we only discuss broadcasting in the even dimensional UHC, since the even di-

mensional UHC is vertex symmetric. Now we describe how to broadcast in the odd dimensional UHC.

An n-UHC can be decomposed into a positive $(n-1)$-UHC and a negative $(n-1)$-UHC, thus we can divide an odd dimensional n-UHC by bit $n-1$. If port $n-1$ of the source node v is an out-port, the source node sends the message to $v \oplus 2^{n-1}$; otherwise it sends message via the path $(v \rightarrow v \oplus 2^{n-2} \rightarrow v \oplus (2^{n-1}+2^{n-2}))$. Then we can applied broadcasting trees of the even dimensional UHC in previous sections to broadcast. The cost of these algorithms will increase one or two steps, depending on whether port $n-1$ of the source node is an out-port.

6 A Fault-Tolerant Broadcasting Tree

As the number of components of a distributed system increases, the probability that some component works incorrectly cannot be neglected. In this paper, we will consider both link failures and processor failures. A link or a processor is *faulty* if it cannot transmit any message. Note that in our model, a processor or a link either transmits or does not transmit messages; in other words, it does not corrupt messages. All faults are assumed to be permanent (or at least are considered to be permanent during the whole execution of the communication algorithm). The algorithm does not test the status of the processors and links; that is, the source node has to send multiple copies of each message to each node.

It has been proved by Edmonds [3] that every digraph possesses as many arc-disjoint spanning trees rooted at any node as the arc-connectivity of the digraph. Here for a digraph $D=(V,A)$, *arc-connectivity* represents the minimum number of elements in an arc set A' such that $D'=(V,A-A')$ is not a strongly connected digraph. It is easy to see that the arc-connectivity of an n-UHC is at most $\frac{n}{2}$, where n is even. Thus the n-UHC has at most $\frac{n}{2}$ arc-disjoint spanning trees rooted at any node. In this section, we will propose an explicit construction of $\frac{n}{2}$ arc-disjoint spanning trees in an n-UHC, where n is even. This family of trees is denoted as n-ADST. Besides, we identify each spanning tree by the out-port number where it is derived from. For example, the left subtree in the 4-ADST in Figure 5 is called subtree 1.

Let node $(0\cdots0)$ be the root. At the beginning, we only consider subtree $\frac{n}{2}-1$. First, the root sends message via the path $(0\hat{0}0\cdots0) \rightarrow (\hat{0}10\cdots0) \rightarrow (1\hat{1}0\cdots0) \rightarrow (\hat{1}00\cdots0)$. Then nodes $(\hat{0}10\cdots0)$, $(1\hat{1}0\cdots0)$ and $(\hat{1}00\cdots0)$ serve as the roots of the subcubes $(01b_{n-3}\cdots b_1b_0)$, $(11b_{n-3}\cdots b_1b_0)$ and $(10b_{n-3}\cdots b_1b_0)$ respectively. Afterwards, these three nodes broadcast the message as in BT_1. At the last step, each node with identifier $(\hat{1}0b_{n-3}\cdots b_1b_0)$ or $(0\hat{1}b_{n-3}\cdots b_1b_0)$ sends message via its out-port $\frac{n}{2}-1$. Subtree j, $0 \le j \le \frac{n}{2}-1$, can be obtained by rotating the identifiers of the nodes in subtree $\frac{n}{2}-1$ right by $n-2j-2$ bits. For given a node i in subtree j, let l be the first superdimension position , in the sequence $j+1, j+2, \cdots, \frac{n}{2}-1, 0, 1, \cdots, j-1$, such that $SD_l(i) \notin \{\hat{0}0, 0\hat{0}\}$. If there does not exist such a superdimension position, $l=-1$. For instance, let $n=10$ and $j=2$: then for $(\hat{1}1\hat{0}0\hat{1}0\hat{0}0\hat{0}0)$ and $(\hat{0}0\hat{0}0\hat{1}0\hat{0}0\hat{1}1)$, $l=4$ and 0 respectively. Thus the children of i in subtree j is given in Figure 6.

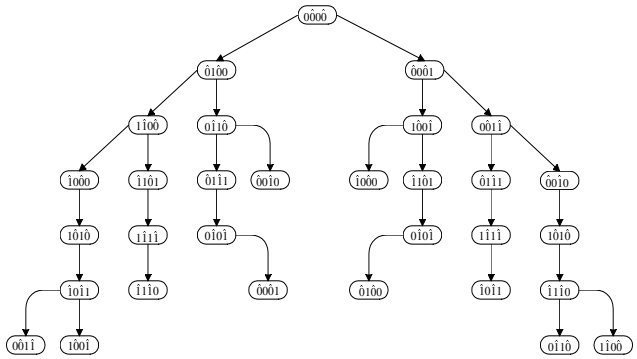

Figure 5: The 4-ADST rooted at $(0\hat{0}0\hat{0})$.

Theorem 6 *The $n/2$ subtrees in the n-ADST graph are arc-disjoint, where n is even.*

Theorem 7 *Two paths between the source and any other node in two subtrees of the n-ADST, n is even, are node-disjoint.*

Theorem 8 *The height of the n-ADST is $\frac{3}{2}n + (n \bmod 4)/2$, where n is even.*

7 Performance Analysis

Let the height of a broadcasting tree be h. Thus the cost of sending a message of length L under all-port communication is $h(\beta + L\tau)$. Now, in case of long message, we can use the pipeline technique proposed by many authors [7, 4, 5, 8] to speed up broadcasting under the all-port communication. First we cut the message into L/B packets of size B, then the message can be sent one after another in pipeline. The broadcasting time is $h(\beta + B\tau)$ for the first packet to reach the farthest node, plus $(L/B-1)(\beta+B\tau)$ for the rest packets to reach the farthest node. Hence the total time is $(h-1+\frac{L}{B})(\beta+B\tau)$. Minimize the total time

$$\begin{cases} \langle i_{\frac{n}{2}-1}\cdots\overline{i_j}\cdots i_0\rangle & \text{if } i \text{ is root,} \\ \langle i_{\frac{n}{2}-1}\cdots\overline{i_k}\cdots i_0\rangle & \text{if } i_j \in \{\hat{1}1,1\hat{1},\hat{0}1,1\hat{0}\} \text{ and } i_l \notin \{\hat{1}0,0\hat{1}\}, \\ \quad \forall k \in \{l, l-1, \cdots, j+1\} & \\ \langle i_{\frac{n}{2}-1}\cdots\overline{i_k}\cdots i_0\rangle & \text{if } i_j \in \{\hat{1}1,1\hat{1},\hat{0}1,1\hat{0}\} \text{ and } i_l \in \{\hat{1}0,0\hat{1}\} \text{ and } l \neq j-1, \\ \quad \forall k \in \{l-1, l-2, \cdots, j+1\} & \\ \langle i_{\frac{n}{2}-1}\cdots\overline{i_k}\cdots i_0\rangle & \text{if } i_j \in \{\hat{1}0,0\hat{1}\} \text{ and } i_l \notin \{\hat{1}0,0\hat{1}\}, \\ \quad \forall k \in \{l, l-1, \cdots, j+1, j\} & \\ \langle i_{\frac{n}{2}-1}\cdots\overline{i_k}\cdots i_0\rangle & \text{if } i_j \in \{\hat{1}0,0\hat{1}\} \text{ and } i_l \in \{\hat{1}0,0\hat{1}\}, \\ \quad \forall k \in \{l-1, l-2, \cdots, j+1, j\} & \\ \emptyset & \text{if } (i_j \in \{\hat{1}1,1\hat{1},\hat{0}1,0\hat{1}\} \text{ and } i_l \in \{\hat{1}0,0\hat{1}\} \text{ and } l = j-1) \text{ or} \\ & \quad (i_j \in \{\hat{0}0,0\hat{0}\} \text{ and } l \neq -1). \end{cases}$$

Figure 6: The children function of n-ADST.

by selecting the packet size, we will get the minimal time is $(\sqrt{(h-1)\beta}+\sqrt{L\tau})^2$ when B is $\sqrt{\frac{L\beta}{(h-1)\tau}}$.

Now we consider the all-port fault-tolerant broadcasting. Let f denote the number of faults in the n-UHC. If $f = n/2-1$, then the message is duplicated $n/2$ times, and each copy is sent through one subtree in the n-ADST. The time complexity is $(\frac{3}{2}n+\frac{n \bmod 4}{2}-1+\frac{L}{B})(\beta+B\tau)$, where n is even and B is the packet size. If $f < n/2-1$, then we cut the message into $n/2$ blocks, $M_0, M_1, \cdots, M_{n/2-1}$. Each subtree j transmits the message which is the composition of blocks M_k, $\forall k \in \{j, j+1, \cdots, (j+f) \bmod (n/2)\}$. If there is no fault, each node can receive $f+1$ copies of the original message, because each block is transmitted via $f+1$ subtrees. Thus each node can receive at least one copy of the message if there are f faults. Since each subtree transmits the message of length $2(f+1)L/n$, the time complexity of the algorithm becomes $(\frac{3}{2}n+\frac{n \bmod 4}{2}-1+\frac{2(f+1)L}{Bn})(\beta+B\tau)$.

If we do not consider the fault-tolerant tree, the formula discussed above is still valid in the one-port communication, except the term h should be replaced by the total start up time needed. We haven't found a scheduling discipline for the n-ADST to broadcast under the one-port communication. In other words, we do not solve the one-port fault-tolerant broadcasting problem. Table 1 is a summary of the complexities of our broadcasting trees.

8 Conclusion

In this paper, we solve the broadcasting problem for the even dimensional uni-directional hypercube (UHC). In the constant evaluation model, the complexity of our all-port broadcasting tree, BT_2, is $n+1$. And it is optimal. Whereas the best of our one-port broadcasting tree, BT_3, needs $\frac{4}{3}(n-n \bmod 6)+\frac{3}{2}(n \bmod 6)$ steps. These algorithms can be extended to solve the odd dimensional case. We also propose an all-port fault-tolerant broadcasting tree whose height is $\frac{3}{2}n+\frac{n \bmod 4}{2}$.

Beside broadcasting and fault-tolerant broadcasting which are described in the paper, we also explored the gossiping problem and the DESCEND/ASCEND algorithms for the even dimensional UHC[6]. The complexities of these algorithms for UHC are the same as those of hypercubes under the one-port half-duplex model.

Given any node v in the n-UHC, where n is even, the number of the elements in V' is $2^{\frac{n}{2}-1}$, where $V' = \{u|$ the distance between u and v is $n+1.\}$. Thus, the lower bound is greater than or equal to $n+2$, i.e. $d+1$, where d is the diameter of the UHC. As a result, we do not know if our best one-port broadcasting algorithm is optimal. Since the height of the n-BT_2 is only $n+1$, yet the height of the n-ADST is $\frac{3}{2}n+\frac{n \bmod 4}{2}$, is it possible to improve it? Another problem we have not solved is the one-port fault-tolerant broadcasting for the UHC. Is it possible to find a scheduling discipline which allows one-port fault-tolerant broadcasting in our n-ADST? These are worth future investigation.

References

[1] J. A. Bondy and U. S. R. Murty, *Graph theory with applications*. The Macmillan Press LTD, 1976.

[2] C. H. Chou and D. H. C. Du, "Uni-directional hypercubes," *Proceedings of Supercomputing'90*, pp. 254–263, 1990.

[3] J. Edmonds, "Edge-disjoint branchings, combinatorial algorithms," *Combinatorial Algorithms* (R. Rustin, ed.), New York: Algorithmic Press, 1972.

[4] P. Fraigniaud, "Asymptotically optimal broadcasting and gossiping in faulty hypercube multicom-

Communication model	Routing	Constant evaluation model	Linear evaluation model	
			Optimal packet size	Minimum time
one-port	BT_1	$\frac{3}{2}n$	$\sqrt{\frac{L\beta}{(\frac{3}{2}n-1)\tau}}$	$(\sqrt{(\frac{3}{2}n-1)\beta}+\sqrt{L\tau})^2$
	BT_2	$\frac{3}{2}n$	$\sqrt{\frac{L\beta}{(\frac{3}{2}n-1)\tau}}$	$(\sqrt{(\frac{3}{2}n-1)\beta}+\sqrt{L\tau})^2$
	BT_3	$x_1(\approx \frac{4}{3}n)$	$\sqrt{\frac{L\beta}{(x_1-1)\tau}}$	$(\sqrt{(x_1-1)\beta}+\sqrt{L\tau})^2$
all-port	BT_1	$\frac{3}{2}n$	$\sqrt{\frac{L\beta}{(\frac{3}{2}n-1)\tau}}$	$(\sqrt{(\frac{3}{2}n-1)\beta}+\sqrt{L\tau})^2$
	BT_2	$n+1$	$\sqrt{\frac{L\beta}{n\tau}}$	$(\sqrt{n\beta}+\sqrt{L\tau})^2$
	BT_3	$x_2(\approx \frac{7}{6}n)$	$\sqrt{\frac{L\beta}{(x_2-1)\tau}}$	$(\sqrt{(x_2-1)\beta}+\sqrt{L\tau})^2$
	ADST	$\frac{3}{2}n+\frac{n \bmod 4}{2}$	$\sqrt{\frac{2(f+1)L\beta}{(x_3-1)n\tau}}$	$(\sqrt{(x_3-1)\beta}+\sqrt{\frac{2(f+1)L\tau}{n}})^2$

Table 1: Complexities of Broadcasting in the n-UHC.

$x_1 = \frac{4}{3}(n - n \bmod 6) + \frac{3}{2}(n \bmod 6)$
$x_2 = \frac{7}{6}(n - n \bmod 6) + \frac{3}{2}(n \bmod 6)$
$x_3 = \frac{3}{2}n + \frac{n \bmod 4}{2}$
L : message length.
f : number of faults.
β : start-up time.
τ : a constant parameter.

puters," *IEEE Trans. Computers*, Vol. 41, No. 11, pp. 1410–1419, Nov. 1992.

[5] P. Fraigniaud, "Complexity analysis of broadcasting in hypercubes with restricted communication capabilities," *Journal of Parallel and Distributed Computing*, Vol. 16, pp. 15–26, 1992.

[6] H. M. Huang, "A study on uni-directional hypercubes," Master's thesis, National Sun Yat-sen University, Kaohsuing, Taiwan, June 1994.

[7] S. L. Johnsson and C. T. Ho, "Optimum broadcasting and personalized communication in hypercubes," *IEEE Trans. Computers*, Vol. 38, No. 9, pp. 1249–1268, Sept. 1989.

[8] Q. Stout and B. Wagar, "Intensive hypercube comminication, prearraged communication in link-bound machines," *Journal of Parallel and Distributed Computing*, Vol. 10, pp. 161–181, 1990.