Communications in a Tree Architecture

Srikanth Kambhatla

Oregon Graduate Institute
Department of Computer Science
and Engineering
19600 N.W. von Neumann Drive
Beaverton, OR 97006-1999 USA

Technical Report No. CS/E 90-018

September, 1990

COMMUNICATIONS IN A TREE ARCHITECTURE

Srikanth Kambhatla
Oregon Graduate Institute
Beaverton, OR 97006

ABSTRACT

Broadcasting and point-to-point communication are common primitives in parallel computations. Efficient implementations of these primitives are of interest in order to improve the performance of these computations. In this paper, we concentrate on the tree architecture and give lower bounds and algorithms for 1) singlenode broadcasts 2) multinode broadcasts 3) scattering and 4) total exchange.

1. Introduction

Tree architectures arise naturally in real time measurement and control systems [1], divide and conquer strategies [2], and searching applications [3]. The popularity of the tree architecture stems from the following properties [4]:

1) It has the least number of arcs possible for the interconnection of a given set of nodes.
2) Trees have important properties for path control, especially in packet switching networks.
3) Trees have preorder, inorder and postorder capabilities.
4) They can implement lookahead logic used in carry lookahead adders, priority circuits and shift-registers.
5) They can naturally implement arithmetic operations and operations that collect data, such as a maximum of a set of numbers.
6) The tree architecture is *strongly inductive.*

Some of the common data communication patterns in parallel numerical methods are [5,6]:

1) *Singlenode broadcasting*: which involves transfer of the same data from one processor to all other processors in the network.
2) *Multinode broadcasting*: which involves concurrent broadcasting from all nodes to all other nodes in the network.
3) *Scattering*: which is singlenode broadcasting with the difference that different packets are transferred to different nodes.
4) *Total exchange*: which involves transfer of a different packet from each node to every other node.

Broadcasting finds applications in a variety of linear algebra algorithms [7], such as matrix-vector multiplication, matrix-matrix multiplication, LU-factorization, and also in database queries and transitive closure algorithms [8]. Scattering and total exchange operations occur in transposing a matrix and in the conversion between different data structures [7].

In this paper, we give lower bounds and algorithms for the communication primitives mentioned above, for the single I/O (where each processor can send or receive at most one message at a time) and multiple I/O (where each processor can simultaneously do I/O operations on all the ports available) cases for the tree architecture. The lower bound results are summarized in table 1.

Table 1.
Lower bounds for some Tree communications.

Communication task	Single I/O	Multiple I/O
Single node b'cast	2logn	logn
Multi node b'cast	(n-1)	(n-1)/3
Scattering	(n-1)	(n-1)/3
Total Exchange	n^2	n^2

The paper is organized as follows. The next section gives some basic definitions and notations used in the rest of the paper. Section 3 deals with the singlenode broadcast operation while the multinode broadcast is described in section 4. Sections 5 and 6 discuss the scattering and total exchange operations respectively. Section 7 describes the related work. Finally, section 8 gives the conclusions.

2. Preliminaries

A *tree* is a Hasse diagram <S,R> where S is the set of nodes and R defines the connectivity among the nodes such that every node (except one, called the *root*) has exactly one arc incident into it. Trivially each pair of nodes has a least upper bound (LUB), which is either one of the nodes if they are in a directed path in the tree, or is the node at which point directed paths from the two nodes meet. Thus, the tree is an upper lattice. The path from any node to any other passes through the LUB [4].

Let *n* be the number of nodes in the tree. In this paper, we will assume a complete binary tree. The number of edges is given by (n-1). The *height* of the tree, *h*, is equal to (log(n+1) - 1). (We imply a base of 2 whenever log is used). We define the nodes labeled cl and cr in the tree in fig 1 as the *corner nodes.* In particular cl is the *left corner node* and cr is the *right corner node.* We use the term *lchild* to denote a node's left child and *rchild* to for the node's right child.

Let d_{ij} be the length of the shortest path between nodes i and j. We then define the *diameter* of the tree as max $\{d_{ij} \mid i,j \in S\}$, the maximum distance between any two nodes. Clearly, this is 2*h. Thus, the worst case communication delay between any two nodes is of the order of the diameter of the graph.

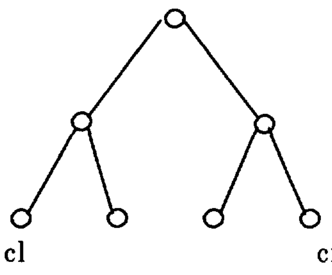

Figure 1. Corner nodes in a tree

The *node connectivity* of a graph is defined as the minimum number of nodes which need to be removed for the graph to be disconnected. Equivalently, it can also be defined as the minimum number of disjoint paths between any two nodes. The *fault tolerance* of the graph is then defined as (node connectivity - 1). Clearly, the node connectivity of a tree is 1 (removing the root partitions the tree) implying that it is 0-fault tolerant.

Labeling of the tree nodes: The nodes are labeled in increasing numbers starting at 0 from left to right in each level and from level 1 to log(n+1). It is clear that under this scheme, a node i has as its children nodes labeled 2*i+1 and 2*i+2. Conversely, the parent of a node j is given by $\lfloor (j-1)/2 \rfloor$.

A simple routing algorithm between any two nodes is given by the algorithm A0 below. We measure the time required for different algorithms in terms of the number of timesteps required, where a timestep is the amount of time for an I/O operation.

Algorithm A0 - Routing from node *source* to node *dest*.

```
/* each node follows this algorithm based on the destination node number in the packet */
if (dest == source) done;
else
        /* find LUB */
        while (dest > source) do {temp = dest; dest = dest/2;}
        if (dest < source) send to parent;
        else if (odd(temp)) send to rchild; else send to lchild;
```

3. Single node broadcast

Single I/O:

Result 1. *2h timesteps are required to broadcast a message from the root of a complete binary tree of height h in the single I/O case.*

Proof: The proof is by induction.
Basis (h=1): with single I/O operation per unit time, we trivially require two time steps for the broadcast.
Induction step: Suppose the proposition holds for a tree of height h. For a tree of height h+1, the routing involves the following steps:
step 1. send the message to its right child.
steps 2 to 1+2h: The right child is a tree of height h. So it will take 2h steps to finish sending the message to all its children. At the start of this phase, the number of links to be traversed on the left of the original root is a left child (tree of height h) and the link connecting the root to the left child. Thus after time steps 1+2h, one link remains to be traveled on the left child.
step 1+2h+1: complete the traversal.
Therefore, it requires 2(h+1) time steps to broadcast a message from the root. □

Result 2. *A lower bound for single node broadcast with single I/O is 2h timesteps.*
Proof: Result 1 assures us that the time taken by the root is 2h. If we take the corner nodes which need to traverse the maximum path in order to accomplish the broadcast, we notice that the number of messages is again at least 2h. □

Algorithm A1 - Single node broadcast with single I/O.

```
If (I am the originator node)
        if (parent exists) send to parent;
        if (rchild exists) send to rchild;
        if (lchild exists) send to lchild;
else (if data received on channel c)
        if (c != parent && parent exists) send to parent;
        if (c != rchild && rchild exists) send to rchild;
        if (c != lchild && lchild exists) send to lchild;
```

Result 3. *Algorithm A1 requires a maximum of 3h-1 time steps for a single node broadcast.*
Proof: The worst case results when the broadcasting node is either cl or cr. The time for the data to propagate to the root is h. One time step is required for the message to reach the other child of the root node, which is a full binary tree of height h-1. The traversal of that subtree takes 2(h-1) time steps (result 1). The nodes on the subtree containing the originating corner node is subsumed by concurrent transmissions during this time. Therefore, the overall time = h + 1 + 2(h-1) = 3h-1 = 3log(n+1)-4, where n is the number of nodes in the tree. □

Multiple I/O:

Result 4. *Broadcast of a message from the root of a complete binary tree of height h in the multiple I/O case requires h timesteps.*
Proof: is again by induction and is similar to the proof of result 1. □

Result 5. *A lower bound for single node broadcast with multiple I/O is h timesteps.*
Proof: Under the possibility of multiple I/O operations at any given time, the minimum time is achieved when the originating node is the root, for which the bound is defined by result 5. □

Algorithm A2 - Single node Broadcast with multiple I/O.

```
If (I am the originator)
        simultaneously send to the following if they exist: parent, lchild and rchild;
else (if data received on channel c)
        simultaneously send to the following if they exist and are != c :
        parent, lchild and rchild;
```

Result 6. *An upper bound for singlenode broadcast with multiple I/O as implemented in A2 is 2h timesteps.*
Proof: The worst case results when the message needs to traverse the diameter of the tree. This happens if the broadcasting node is one of the corner nodes. Irrespective of the amount of concurrency in sends, these nodes take h time steps to reach the root from where we require another h time steps to reach the other corner node. □

4. Multinode Broadcast

Single I/O:

Result 7. *A lower bound for multinode broadcast with single I/O is (n-1) timesteps.*
Proof: Each node receives one message from every other node (which is n-1 messages in all). Therefore, the lower bound for single I/O is (n-1). □

The following scheduling disciplines are possible for any given node with single I/O:
1. sPRL: send to parent, then to rchild, and finally to lchild.
2. sPLR: send to parent, then to lchild, and finally to rchild.
3. sLRP: send to lchild, then to rchild, and finally to parent.
4. sRLP: send to rchild, then to lchild, and finally to parent.
5. sRPL: send to rchild, then to parent, and finally to lchild.
6. sLPR: send to lchild, then to parent, and finally to rchild.

Result 8. *The time for multinode broadcast when all the nodes follow any of the above schedules is the same. Further, that time is not optimal.*

proof: First, we make three observations:
1. When all the nodes send messages to their rchild, only half the nodes in the tree receive one message and the other half do not receive any message at all.
2. When all the nodes send messages to their lchild, the result is analogous.
3. When all the nodes send messages to their parents, half the nodes in the tree receive two messages at once while the other half do not get any. Since, the nodes can receive only one message at a time (because of single I/O), half the nodes are engaged for two time steps while the other half send the message in one time step and idle in the other.

Thus, operations (1) and (2) require one time unit each while operation (3) needs two time steps. Any multinode broadcast algorithm which adopts any of the scheduling algorithms above for all the nodes is essentially composed of the three operations mentioned above. Thus all such algorithms need the same number of time steps. Further, in all such algorithms half of the nodes do not receive any message in any given time step. Hence, they require at least twice the number of time steps which might be required in the best possible algorithm. □

Result 9. *For a tree interconnect with single I/O, a maximum of* $\lceil(5^*n)/8\rceil$ *nodes (within an accuracy of 1) can receive messages at any time step, with any combination of the above schedules at different nodes in the tree.*
Proof: A tree of n nodes has 1 root, $\lceil(n/2)\rceil$-1 internal nodes and $\lceil n/2\rceil$ leaves. In any operation, the root and the $\lceil(n/2)\rceil$-1 internal nodes can send one message each such that the receiving node receive one and only one message each. This ensures that the message sent will always be received in the time step. This is trivially achieved when the root and all the internal nodes send messages on their right links.

There remain $\lceil(n/2)\rceil$ nodes at the leaves which have not sent any message so far. The tree configuration is such that two leaf nodes always have a common parent. Thus, if it has to be ensured that the parent receives the message in the same time step, only one of its two children can send a message. This reduces the number of children which can send a messages to (1/2) * $\lceil(n/2)\rceil$. Further, half of the parents in the penultimate level will be receiving messages from their parents. This implies that the number of leaf nodes which can send messages to their parents is further reduced by half and becomes $\lceil(n/8)\rceil$.

Thus, the total number of messages which can be exchanged at any given step is 1 + $\lceil(n/2)\rceil$ - 2 + $\lceil(n/8)\rceil$ which is about $\lceil(5n)/8\rceil$. □

Algorithm A3 gives a simple scheme for implementing multinode broadcast with single I/O. A3 is essentially a multi-single node broadcast, with all the nodes following the sLRP scheduling discipline.

There are several ways in which the upper bound on concurrency defined by result 9 can be met. The approaches differ in the specific (5n)/8 nodes which can receive messages. While such algorithms can be derived, it has been our experience that the approach does not gain us much over the simple multinode broadcast algorithm A3. The reason is that although most of the exchanges are accomplished in less than h steps, the few

Algorithm A3 - simple multinode broadcast with single I/O.

```
Each node repeats 1 thru 3, (2*h) times:
        1. update left child
        2. update right child
        3. update parent
```

exchanges which remain, necessitate a similar figure for the number of timesteps.

Multiple I/O:

Result 10. *A lower bound for the multinode broadcast with multiple I/O is (n-1)/3 timesteps.*
Proof: Each node is to receive (n-1) messages, and it can receive at most three of those at any time step. □

We *emulate* a ring in the tree and circulate the data from each node. In order to achieve this, we need to re-label the nodes of the tree. We follow an in-order labeling of the nodes as shown in fig 2. Then, the ring emulation is achieved by sending messages to successive nodes. An algorithm performing these actions is given as algorithm A4 below.

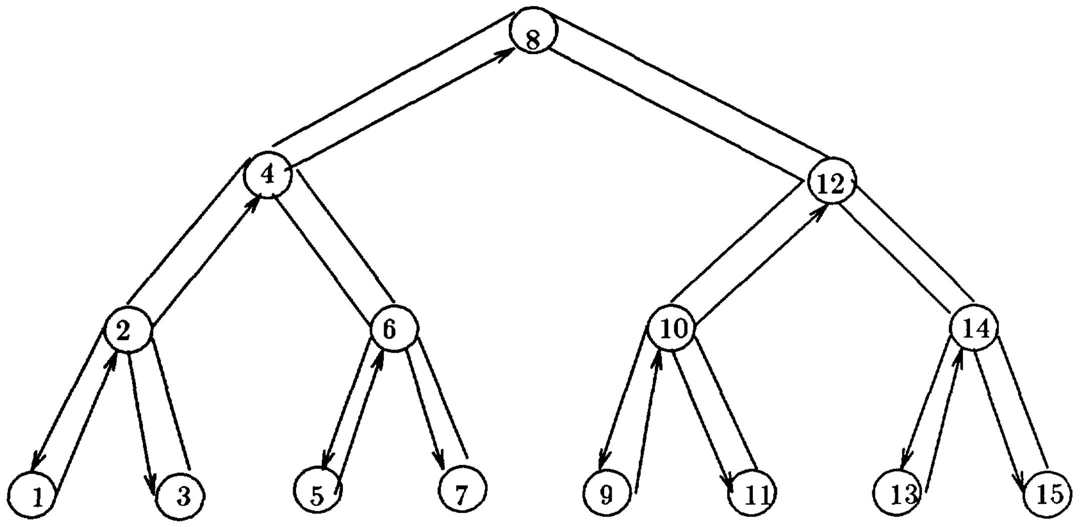

Figure 2. Ring emulation in a tree.

Although Figure 2 achieves the emulation, the number of hops between neighbors of the ring varies within the ring. In particular, this maximum number grows logarithmically with the size of the tree. The problem persists with preorder and postorder ring

Algorithm A4 - Multinode Broadcast with multiple I/O.

Emulate a ring in the tree.
Each node starts by sending its data to its neighbor in the ring in one direction.
In the subsequent steps, propagate any data which the node gets from the other direction.

emulations as well. Shaw [9] describes a particular ring emulation where the maximum distance between any two neighbors is exactly three. With such a *bounded neighbor* emulation, we can achieve a multinode broadcast with multiple I/O in a maximum of 3*(n-1) timesteps. The specific labeling of nodes is given in figure 3 for a tree of 15 nodes.

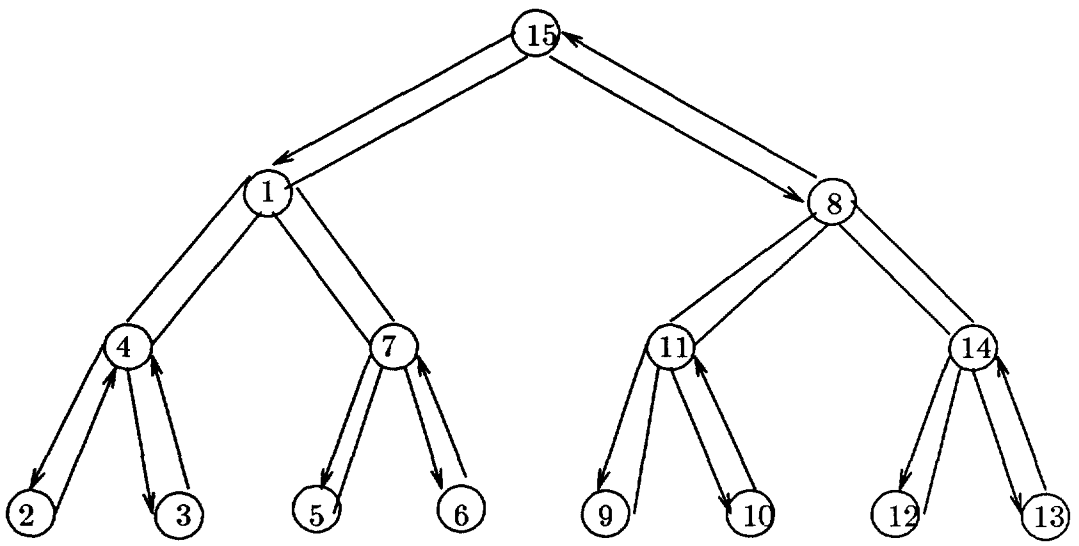

Fig 3. Bounded neighbor emulation of Shaw.

5. (Single node) Scattering

Single I/O:

Result 11. *A lower bound for single node scattering with single I/O is (n-1) timesteps.*
Proof: The source node needs to send (n-1) messages. □

Scattering of data can be done in several ways. One way is to simply use the routing algorithm A0 repeatedly. The worst case performance of such a scheme results when the source node for the scatter is one of the corner nodes. If such were the case, the total number of hops would certainly be greater than (n/4)*(2h) = O(nlogn) even with pipelining (there are $\lceil n/4 \rceil$ nodes at a distance 2h away). We therefore, retain our ring emulation algorithm, by which we can achieve the scattering operation in 3*(n-1) timesteps.

We have two options in the implementation of the scatter. First, we might decide to send data in one direction only and pipeline the data for the other nodes along the way. We call this the option **A**. Thus, the farthest node in this scheme would be the neighboring node in the other direction. In our next option, which we call the option **B**, we can alternately send data to the successively closer nodes in both directions (the farthest node in this scheme would be the node diametrically opposite to the sender). However, due to the single I/O communication model, we can send data in the same direction only half the number of times. Algorithm A5 implements option A.

Algorithm A5 - Scattering with single I/O.

```
Emulate a ring in the tree.
The source node i sends out the data for the node (i-1) mod n along the longer path
and pipelines the data for successively nearer nodes.
```

Result 12. *For a ring network, options A and B cost the same.*
Proof: First we take the case when n is even. The number of nodes which need to receive the information is odd. Considering option B, there exists a unique node which is the farthest distance away. We send data to that node in the clockwise direction (say), in timestep 1. We now have two nodes equally far off in both directions. In step 2, we send data off to the farthest node in the anticlockwise direction. We alternate these steps. The messages sent off in times 1 and 2 reach their destinations at the time (n/2). Subsequently, two nodes will receive their data at each time step. Since there are n/2 nodes in the clockwise direction (and one less in the other direction), the total amount of time taken is (n/2)+(n/2-1) = n-1, which is the same as option A. Fig 4 illustrates the routing.

We now consider the case when n is odd. We now have two nodes which are farthest from the source. Data arrives at one farthest node at time $\lceil n/2 \rceil$. Thereafter the situation is same as before, i.e. two nodes receive data at each timestep. Thus, the time for option B is now 1 + time for (n-1) nodes. Option A also takes exactly one time step more, because there is exactly one node extra in the ring. Thus, both options are equivalent. □

Corollary. *In a ring emulation in a tree, option B is always better.*
Proof: In a ring network, the message sent in time 1 will reach the neighboring node at time 2. However, this does not hold when the dilation factor is greater than one (as in the ring emulation). For example in tree, it takes 3 hops to reach the nearest neighbor. The time taken for a message to reach a node diametrically opposite in our ring emulation is 3*(n/2). But, during this time, the source node has time to send off 3*n/2

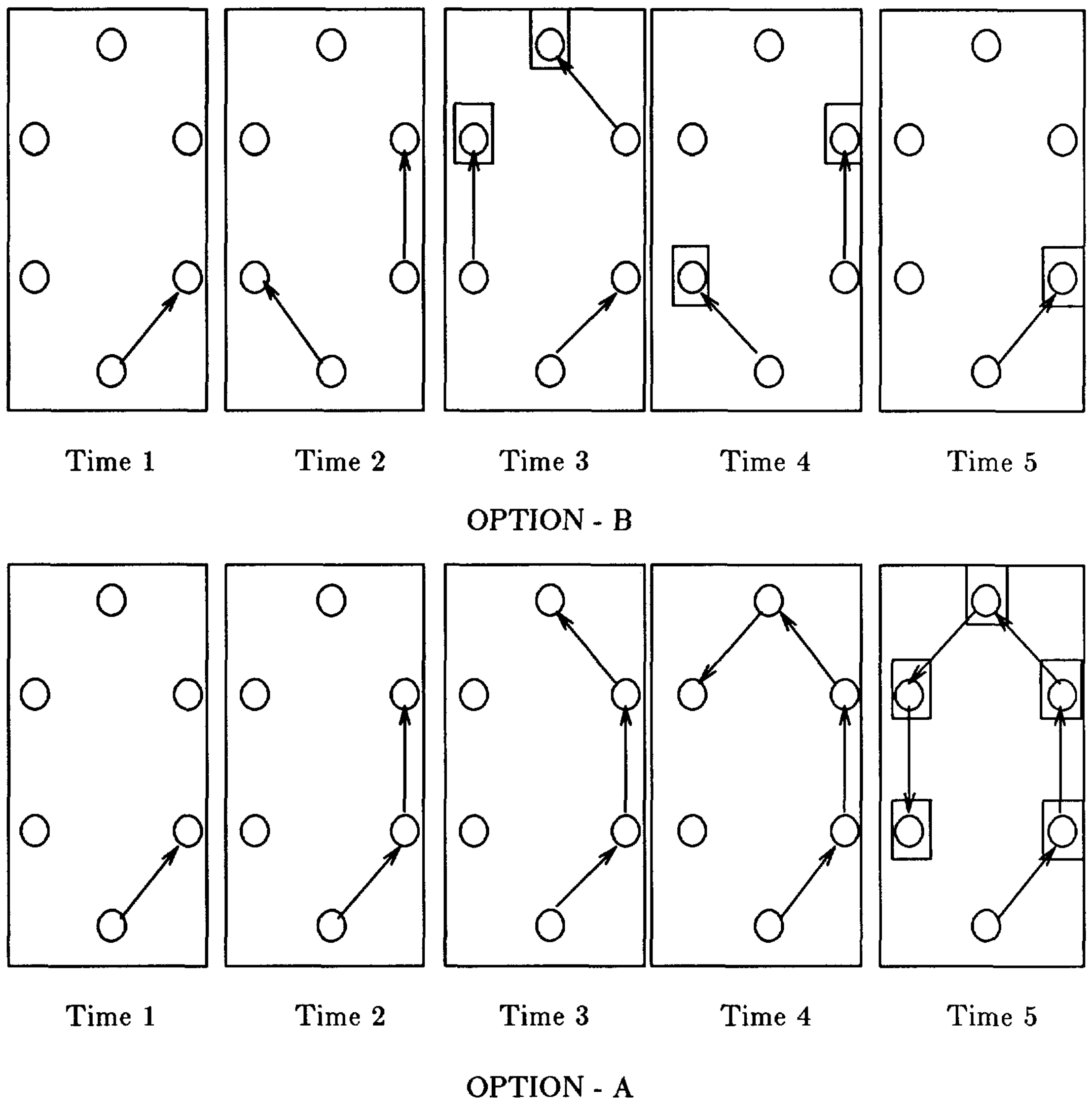

Figure 4. Options A and B when n is even.
Note: A node with a square represents the arrival of data for that node.

messages which is more than (n-2) messages that it had to send. Thus, by the time the message reaches the farthest node in the ring, all messages to the other nodes have been dispatched. The time at which the node (n/2 - 1) gets its message is 3*(n/2 - 1) + 2 (two is added because messages in both directions are sent alternately starting at 0). But this figure is less than 3*(n/2). The time required for a message to reach its nearest neighbor in the emulated ring is more than the time required for another message in the same direction to get started. This means that the time for broadcast by option B is the time required for the message to reach the diametrically opposite node, which is 3*(n/2). But by scheme A, the time for broadcast is 3(n-1). □

Multiple I/O:

Result 13. *A lower bound for scattering with multiple I/O is (n-1)/3 timesteps.*
Proof: The source needs to send (n-1) messages and can send 3 messages at each time step. □

The approach adopted is the option B above with the difference that data is now sent simultaneously in both directions. The maximum time required is 3*(n/2).

Algorithm A6 - scattering with multiple I/O.

```
Emulate a ring in the tree.
Simultaneously send to nodes (n/2) and (n/2 -1) (if n is even; and the corresponding nodes when n is
odd) in the two different directions of the ring.
Pipeline data for the next farthest nodes.
```

---

6. Total Exchange

Result 14. *A lower bound for any total exchange algorithm with single I/O is O(n^2) timesteps. This result holds in the multiple I/O case as well.*
Proof: Each node needs to transfer (n-1) messages. We can make cuts at different edges and see the number of messages which need to cross the edge. If we make a cut on the edge connecting the root to its lchild, we see that the number of messages which need to cross the link to the subtree having the lchild, N, given by: $N = ((n-1)/2 + 1) * ((n-1)/2)$ is clearly the maximum among such numbers. This therefore is the minimum time any total exchange algorithm needs.

Since there is only a single edge connecting the root and its lchild, O(n^2) messages need to travel the edge sequentially even in the multiple I/O case. □

Algorithm A7 - Total exchange.

```
All the nodes follow the algorithms A5 and A6 for single I/O and multiple I/O respectively.
```

---

7. Related work

Johnsson and Ho [6] describe the broadcast and personalized communications in the hypercube. They give the lower bounds and also describe algorithms which achieve the communication tasks. Saad and Schultz [5] describe the same operations for a broadcast bus, shared memory model, ring, mesh, hypercube and switch models of parallel architectures. Fraigniaud *et. al.* [10] demonstrate the optimality of a scattering algorithm for a ring of processors. Although there has been some work reported on individual instances of these communication primitives (mostly single node broadcasts) on specific interconnects, a comprehensive treatment of communication tasks has been quite limited.

8. Conclusions

We have presented the lower bounds and algorithms achieving various forms of broadcasts and personalized communications in a tree architecture. In each case, we assumed two models of communication. The lower bound results are summarized in table 1.

Acknowledgements
The author wishes to thank Micheal Wolfe for his comments on the paper.

References

1. H. F. Li and C. C. Lau, "A Distributed Multiprocessor Traffic Control System," *Proceedings COMPSAC 78*, pp. 259-264, November 1978.
2. E. Horowitz and A. Zorat, "The Binary Tree as an Interconnection Network: APplications to Multiprocessor Systems and VLSI," *IEEE Transactions on Computers*, pp. 247-253, April 1981.
3. J. Bentley and H. T. Kung, "A Tree Machine for Searching Problems," *Proceedings of the IEEE International Conference on Parallel Processing*, 1979.
4. G. J. Lipovski and M. Malek, *Parallel Computing theory and comparisions,* John Wiley & Sons, 1987.
5. Y. Saad and M. H.Schultz, "Data Communications in Parallel Architectures," *Parallel Computing*, vol. 11, pp. 131-150, 1989.
6. S. L. Johnsson and C. T. Ho, "Optimum Broadcasting and Personalized Communications in Hypercubes," *IEEE Transactions on Computers*, vol. 38, 9, pp. 1249-1268, 1989.
7. S. L. Johnsson, "Communication Efficient Basic Linear Algebra Computations on Hypercube Architectures," *Journal of Parallel and Distributed Computing*, vol. 4, pp. 133-172, 1987.
8. S. A. Browning, *The tree machine: A highly concurrent computing environment,* Tech Report 1980:TR:3760, Computer Science, California Institute of Technology, 1980.
9. D. E. Shaw, "Organization and Operation of a Massively Parallel Machine," *Advanced Semiconductor Technology and Computer Systems*, Van Nostrand Reinhold Company, 1987. Guy Rabbat (ed.)
10. P. Fraignaud, S. Miguet, and Y. Robert, "Scattering on a Ring of Processors," *Parallel Computing*, vol. 13, pp. 377-383, 1990.