JOURNAL OF PARALLEL AND DISTRIBUTED COMPUTING **30**, 85–90 (1995)

Optimal Communication Algorithms in Distributed Loop Networks

RAJIB K. DAS AND BHABANI P. SINHA[1]

Electronics Unit, Indian Statistical Institute, Calcutta 700035, India

Time optimal algorithms for multiple node broadcast and single node scatter in distributed loop networks have been proposed in this paper. These algorithms involve the minimum number of packet transmissions to effect both multiple node broadcast and single node scatter. © 1995 Academic Press, Inc.

1. INTRODUCTION

In multiprocessor networks, the computation time at each node is often small compared to the interprocessor communication time. As a result, it is necessary to perform internode communications as fast as possible. There are different types of communication requirements in a network [3], e.g., single node broadcast, multiple node broadcast, single node scatter, and multiple node scatter (also known as total exchange).

Distributed loop network is one of the popular network topologies. A distributed loop network [5], denoted by $G(n; 1, s)$, is defined as follows:

(1) The total number of nodes in the network is n;

(2) s is an integer such that $1 < s < n/2$;

(3) If the n nodes are numbered as $0, 1, 2, \ldots, n-1$, then the node i is connected to the nodes $(i \pm 1) \bmod n$ and $(i \pm s) \bmod n$ by symmetric (bidirectional) edges for all i, $0 \le i \le n-1$.

As an example, the graph $G(14; 1, 6)$ has been shown in Fig. 1. The network $G(n; 1, s)$ has many attractive features such as regular structure and constant degree, and is well-studied in the literature [5, 6]. This network is a special case of a well-known class of networks called circulant graphs [1], which have some good properties in the context of optimal fault-tolerant design [2]. By suitably choosing the value of s, we can achieve a diameter on the order of $\sqrt{n}$. Optimal routing algorithms for distributed loop networks have been proposed in [8], for the fault-free case as well as for the case with a single fault.

In this paper, we consider the multiple node broadcast and single node scatter problems in a distributed loop network. Our proposed algorithms are based on the model of multiple link availability (MLA) [3], where we assume that each node can receive or send packets through all of its adjacent links simultaneously. Links are bidirectional and full duplex communication is possible through each link. However, if there are more than one packet to be transmitted through the same link in the same direction at the same time, only one packet can be sent, and the remaining packets have to wait in a queue. We also assume that each packet transmission takes one unit of time; i.e., packets are all of the same length. Packet transmissions are assumed to be error-free. Under this model, our proposed algorithms for both multiple node broadcast and single node scatter are optimal in time. As each node has to store the packets from all the nodes of the network, a buffer of size n is required at each processor node.

In Section 2, we shall first discuss a few basic results and then present our algorithm on multiple node broadcast in optimal time. Section 3 deals with the time-optimal algorithm for single node scatter.

2. MULTIPLE NODE BROADCAST

Since the degree of a node in a distributed loop network is 4, a node can receive at most four packets at a time. Since each node has to receive a total of $(n-1)$ packets for multiple node broadcast, the minimum time for multiple node broadcast is $\lceil (n-1)/4 \rceil$. A time-optimal algorithm for multiple node broadcast should be completed within time $\lceil (n-1)/4 \rceil$, provided of course that the diameter is not more than this quantity. We have shown that the diameter of $G(n; 1, s)$ is less than or equal to $\lceil (n-1)/4 \rceil$ and have developed a multiple node broadcast algorithm which requires $\lceil (n-1)/4 \rceil$ time, and hence is optimal with respect to time. The minimum number of packet-transmissions required for this operation is $n(n-1)$.

Our algorithm for multiple-node broadcast is based on the ideas described in [3]. There, it has been shown that for a time-optimal multinode broadcast algorithm in a hypercube, all that we need is to construct a spanning tree rooted at a certain node satisfying certain properties. Let A_t denote the set of links of the spanning tree those are used for communication at the time instant t, $t = 1, 2, \ldots, T$, such that $\bigcup_{t=1}^{T} A_t$ is the set of all edges in the spanning tree. If the set A_t satisfies the property that for each time t no two links in A_t are of the same type (along the same dimension of the hypercube), then by replicating this tree

[1] E-mail: bhabani@isical.ernet.in.

and the time schedule at each node, a multinode broadcast algorithm can be derived which completes within time T.

In a loop network, we define the type of a link $[v, w]$ connecting v to w as $\Delta(v, w)$, where $w = (v + \Delta(v, w)) \bmod n$. The links can be of four types, namely, $+1$, -1, $+s$, and $-s$.

EXAMPLE 1. Consider the distributed loop network $G(14; 1, 6)$ as shown in Fig. 1. The type of the link $[1, 2]$ is $+1$ and that of the link $[1, 7]$ is $+6$. The types of the links $[13, 0]$ and $[12, 4]$ are also $+1$ and $+6$, respectively. The types of the links $[6, 7]$, $[0, 13]$, $[1, 7]$, and $[6, 0]$ are $+1$, -1, $+6$, and -6, respectively.

In what follows, we shall describe a method for generating $A_{t,0}$'s (0 stands for source node 0) such that for all t, $A_{t,0}$ contains at most one link of each type. For any other source node $u \in \{1, 2, ..., n-1\}$, we use modulo n rotation along the loop to obtain $A_{t,u}$ from $A_{t,0}$ as follows: If $[v, w] \in A_{t,0}$, then the link $[(v + u) \bmod n, (w + u) \bmod n] \in A_{t,u}$.

Let S_k, $1 \le k \le d$, be the set of nodes at a distance k from the node 0, where d is the diameter of the graph $G(n; 1, s)$. A shortest path between any two nodes consists of either of $+s$ and $-s$ links and/or either of $+1$ and -1 links [8]. Hence, any node in S_k, can be written in the form $a_1.s + a_2$, where a_1 and a_2 are integers and $|a_1| + |a_2| = k$. Here $|a_1|$ is the number of $+s$ or $-s$ links and $|a_2|$ is the number of $+1$ or -1 links used in the path. For a node u in S_k, the node $n - u$ is also in S_k. The nodes in S_k can be easily enumerated from those in S_{k-1}. For $k = 1, 2, ..., d$, we generate the elements of S_k in pairs, in the following order:

$$(k, -k), (s + k - 1, -s - (k-1)), (s - (k-1), -s + (k-1)), ..., (j.s + (k-j), -j.s - (k-j)), (j.s - (k-j), -j.s + (k-j)), ..., (k.s, -k.s).$$

Here all numbers are taken modulo n, the node pair $(k, -k)$ are reached using k number of $+1$ and -1 links, respectively, and so on.

It may be noted that in the above method of generation, some elements of S_k may be duplicated and in that case, the corresponding term will be indicated by a dash. To facilitate our later discussions, we construct a $d \times 2d$ matrix M with these generated pairs of S_k's as follows:

1. The elements of S_k are entered in the kth row of the matrix, each pair generated as above being placed in one column. The pairs are placed in the successive columns in the above order of their generation, starting from the column 1.
2. The entries in columns $2k + 1$ through $2d$ are all filled up by a null ($\varnothing$) entry.
3. If an element is already generated in some S_p, $p \le k$, then we keep it out to avoid duplication and mark the corresponding position in S_k by a dash (—).

An illustrative example is given below.

EXAMPLE 2. Consider the network $G(14; 1, 6)$ of Fig. 1. The diameter d of this graph is 3. Hence, M will be a 3×6 matrix with the pairs of S_1, S_2 and S_3 as follows:

S_1: (1, 13)	(6, 8)	$\varnothing$	$\varnothing$	$\varnothing$	$\varnothing$
S_2: (2, 12)	(7, —)	(5, 9)	(—, —)	$\varnothing$	$\varnothing$
S_3: (3, 11)	(—, —)	(4, 10)	(—, —)	(—, —)	(—, —)

Whenever a pair of generated nodes in S_k will be encountered in our discussion, we will treat that as an ordered pair.

Before describing the algorithm for multiple node broadcast, we investigate the following properties of the sets S_k's.

2.1. Properties of the Sets S_k

We observe the following properties of the sets S_k's (all node numbers are taken *modulo n*):

PROPERTY 1. *The pairs generated as above are always in the form $(u, n-u)$. In a generated pair, if only u is present, but not $n - u$, then u must be equal to $n/2$. In other words, the nodes are duplicated in pairs (except the node $n/2$).*

PROPERTY 2. *Let j be an integer, $0 < j < k$ and let $k > 2$. If an element $j.s + (k - j)$ is present in S_k, then S_{k-1} must contain both the nodes $(j-1)s + (k-j)$ and $j.s + (k - j - 1)$. A similar property also holds for the elements of the form $j.s - (k-j)$, $-j.s - (k-j)$, and $-j.s + (k-j)$ in S_k.*

PROPERTY 3. *For $k > 2$, if the node k is present in S_k, then S_{k-1} must contain the node $k - 1$ and if the node k.s is present in S_k, then $(k-1).s \in S_{k-1}$.*

PROPERTY 4. *In the process of sequentially generating the pairs of S_k, each S_k will contain 2k pairs of nodes, until any duplication occurs. If a duplication occurs in some S_p, either k or k.s or both will be absent in S_k, for $k > p$.*

Proof. Omitted due to brevity. Detailed proof is given in [7]. ■

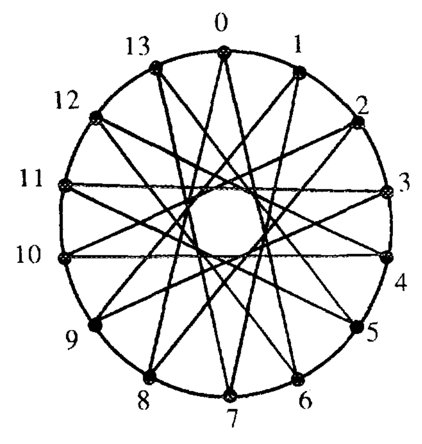

FIG. 1. The distributed loop network $G(14; 1, 6)$.

PROPERTY 5. *If n is even and the node $n/2$ occurs in S_k in any form other than $(k.s)$ then all 4 nodes adjacent to the node $n/2$ must be in S_{k-1}.*

Proof. Omitted due to brevity. Detailed proof is given in [7]. ■

PROPERTY 6. *If for some k, $|S_k| < 4$, then k is the diameter of the graph.*

Proof. Omitted due to brevity. Detailed proof is given in [7]. ■

PROPERTY 7. *The diameter d of $G(n; 1, s)$ is less than or equal to $\lceil (n-1)/4 \rceil$.*

2.2. Generation of $A_{t,0}$

DEFINITION 1. Let (w, x) be a pair in S_k. For (w, x) in any form other than $(k, -k)$ or $(k.s, -k.s)$ we define two pairs (w_{+1}, x_{-1}) and (w_{+s}, x_{-s}) in S_{k-1} such that w and x are connected to w_{+1} and x_{-1} respectively, by links of type $+1$ and -1 and w and x are connected to w_{+s} and x_{-s} respectively, by links of type $+s$ and $-s$. When $(w, x) = (k, -k)$, then we define $(w_{+1}, x_{-1}) = (k-1, -k+1)$ and when $(w, x) = (k.s, -k.s)$ we define $(w_{+s}, x_{-s}) = ((k-1)s, -(k-1)s)$.

DEFINITION 2. We define an ordered set S'_k, by selecting the non-null elements of the k-th row of the matrix M, in order from left to right, excluding the dashes(-).

Remark. The purpose of S'_k is to keep track of the nodes which are not yet included in the spanning tree.

We first give an overview of the generation process. $A_{t,0}$'s are generated in the increasing order of t, starting from $t = 1$. In S_1, there are two pairs $(1, -1)$ and $(s, -s)$. $A_{1,0}$ consists of the links joining the node 0 to the pairs in S_1. We generate the subsequent $A_{t,0}$'s as follows.

We note that we have generated the elements in S_k in the following order:

$(k, -k), (s + k - 1, -s - (k - 1)), (s - (k - 1), -s + (k - 1)), \ldots, (js + (k - j), -js - (k - j)), (js - (k - j), -js + (k - j)), \ldots, (k.s, -k.s)$.

Before any duplication occurs there will be $2k$ such pairs in S_k. We select a set of four nodes (w, x, y, z) from S_k as follows. We select the leftmost pair in S'_k. Let this pair be (w, x). We also select the rightmost pair in S'_k. Let this pair be (y, z). We can always find $A_{t,0}$ connecting the nodes (w, x, y, z) satisfying the requirement of different link types as $\{[w_{+1}, w], [x_{-1}, x], [y_{+s}, y], [z_{-s}, z]\}$. Next we delete the pairs (w, x) and (y, z) from S'_k and select the leftmost and rightmost pairs in the new S'_k. This way we go on until all the nodes in the network are included in the spanning tree.

If $(n - 1) \bmod 4 = 0$, all four link types will be present in each $A_{t,0}$. If $(n - 1) \bmod 4 = 2$, we make all but the last $A_{t,0}$ with four links. If $(n - 1) \bmod 4 = 3$, one of the $A_{t,0}$'s will have three links and the rest four links each. If $(n - 1) \bmod 4 = 1$, we can make either one $A_{t,0}$ with 1 link and the rest with four links each or one $A_{t,0}$ having three links, another $A_{t,0}$ having two links and the rest with four links each. We now consider below a few more points in detail for odd and even values of n.

2.2.1. For Odd n

Since n is odd, the elements of S_k will always occur in pairs. If a duplication occurs in some S_p, for $p \le k$, then some pairs will be absent in S_p and there may be odd number of pairs in S_k. In that case we have to select $A_{t,0}$ by taking one pair from S_k and one pair from S_{k+1}. By Property 3, either the pair $(k + 1, -k - 1)$ or $((k + 1)s, -(k + 1)s)$ will be absent in S_{k+1}.

Consider the case when the pair $(k + 1, -k - 1)$ is absent in S_{k+1}, and S'_k contains one pair, say (w, x). Since S_{k+1} is nonempty S_k must have more than one pair and since we have started to connect the nodes in S_k by selecting the leftmost and rightmost pair, (w, x) cannot be of the form $(k, -k)$ or $(k.s, -k.s)$. We select the rightmost pair from S_{k+1} say (y, z). If the pair (y, z) is of the form $((k + 1)s, -(k + 1)s)$, we construct $A_{t,0}$ as $\{[w_{+1}, w], [x_{-1}, x], [y_{-s}, y], [z_{-s}, z]\}$. Here $(y_{-s}, z_{-s}) = (k.s, -k.s)$ which are already included in the spanning tree. If (y, z) is not $((k + 1)s, -(k + 1)s)$, then we make $A_{t,0} = \{[w_{+1}, w], [x_{-1}, x], [y_{+s}, y], [z_{-s}, z]\}$ provided $(w, x) \ne (y_{+s}, z_{-s})$, otherwise we make $A_{t,0} = \{[w_{+s}, w], [x_{-s}, x], [y_{+1}, y], [z_{-1}, z]\}$.

If the pair $(k + 1, -k + 1)$ is present in S_{k+1} we select the pair (y, z) from the leftmost end of S_{k+1} and proceed as before.

2.2.2. For Even n

After any duplication occurs, the number of elements in S_k will be odd only when $n/2$ is present in S_k, and even otherwise. The generation process is same as that for odd n, except when the node $n/2$ is to be connected to the tree. When the node $n/2$ is to be connected and there remains another pair (w, x) unconnected to the spanning tree, we make $A_{t,0}$ consisting of three links, connecting the nodes $(w, x, n/2)$ and also maintaining the property of different link types. If no other node remains unconnected to the tree, then we make $A_{t,0}$ consisting of only one link connecting $n/2$ to the tree.

We now give an overall idea for the multiple node broadcast algorithm. At each node, a buffer of size n is needed to store data packets from n processors. Each of these buffers has n locations with addresses, ranging from 0 to $n - 1$. The packet originated from node r will be placed in the location $(r - u)$ of the buffer of node u. The overall arrangement can be shown as:

	$P(u)$	$P(u+1)$	$P(u+2)$	... $P(r)$ ...	$P(u-1)$
address	0	1	2	$r - u$	$n - 1$

$P(r)$: Data packet originated from node r.

The sets $A_{1,0}$, $A_{2,0}$, ..., $A_{T,0}$ are determined beforehand once for all, and we store for each instant of time, the address of the buffer from which a data packet is to be sent and the link along which that packet is to be sent. Let $[w, x] \in A_{t,0}$. Then $[w + u, x + u] \in A_{t,u}$. Hence at time t, the node $(w + u)$ must send the packet originated from the node u, i.e., $P(u)$ which is stored in location $(n - w)$ of its buffer, to the node $(x + u)$. To implement this, we need to store the buffer address $(n - w)$ and the link type $\Delta(w + u, x + u)$ which is same as the link type $\Delta(w, x)$. Thus the information regarding the link $[w, x]$ in $A_{t,0}$ is sufficient to effect transmission of data packets from all the nodes in the network. If $A_{t,0} = \{[w_1, w], [x_1, x], [y_1, y], [z_1, z]\}$ we store the tth record consisting of four pairs (b_1, l_1), (b_2, l_2), (b_3, l_3), (b_4, l_4) for any node $(w + u)$ as follows:

$(n - w_1)$	$\Delta(w_1, w)$	$(n - x_1)$	$\Delta(x_1, x)$	$(n - y_1)$	$\Delta(y_1, y)$	$(n - z_1)$	$\Delta(z_1, z)$
b_1	l_1	b_2	l_2	b_3	l_3	b_4	l_4

There will be $\lceil (n - 1)/4 \rceil$ such records. For $t = 0, 1, ..., \lceil (n - 1)/4 \rceil$, each node will fetch the tth record, and transmit the packet in location b_i along the link of type l_i.

3. SINGLE NODE SCATTER

In scattering, a node has to send $(n - 1)$ different packets to each of the other nodes in the network. Since a node can transmit at most four packets at a time, the minimum time required for single node scatter is $\lceil (n - 1)/4 \rceil$. Also, no scattering algorithm can be completed in time less than the diameter of the network. We have already shown that the diameter of $G(n; 1, s)$ is less than or equal to $\lceil (n - 1)/4 \rceil$. We will present now a time-optimal algorithm for single node scatter which requires $\lceil (n - 1)/4 \rceil$ units of time.

To describe our scattering algorithm, we assume that the node 0 is the source node. The packets will be transmitted from the node 0, along a spanning tree T rooted at node 0. T consists of four subtrees T_{+1}, T_{-1}, T_{+s}, and T_{-s} rooted at the nodes $+1$, -1, $+s$, and $-s$, respectively. Each of the four subtrees contains at most $\lceil (n - 1)/4 \rceil$ nodes.

With such a construction of the spanning tree, all the nodes will receive their packets within time $\lceil (n - 1)/4 \rceil$, if the following rule for transmission of packets is obeyed [3].

Node 0 sends packets to distinct nodes in the subtree (using only the links in T), giving priority to nodes farthest away from node 0 (breaking ties arbitrarily).

We also ensure that each packet travels along the shortest path to its destination by making T a shortest path tree.

3.1. Construction of the Spanning Tree

We find the sets S_k's for the graph $G(n; 1, s)$ as before. We maintain the property that if a node u of a generated pair $(u, n - u)$ is in T_{+1}, then the node $(n - u)$ will be in T_{-1} or if u is in T_{+s}, then $(n - u)$ will be in T_{-s}. We divide the total set of $(n - 1)$ nodes into two partitions of nearly equal size: partition I, consisting of the pairs which will be included in the trees T_{+1} and T_{-1}, and partition S, consisting of the pairs which will be included in the trees T_{+s} and T_{-s}.

Before going into the details of partitioning the nodes, we make the following observations on the matrix M.

Observation 1. In row k, the pair in column 1 is of the form $(k, -k)$. So we put all the pairs in column 1 in partition I.

Observation 2. All the pairs of the form $(k.s, -k.s)$ will be put in the partition S.

Observation 3. If a node u of a pair $(u, n - u)$ in S_k, is adjacent to some node u' in S_{k-1} then $(n - u)$ is adjacent to the node $(n - u')$ in S_{k-1}.

The method of grouping the nodes for partition I and partition S is almost identical for odd and even values of n. First, we describe the procedure for odd n.

3.1.1. For odd n

Since n is odd, there will be a total of $(n - 1)/2$ pairs in all the sets S_k's. We collect the pairs for partition I as follows. We leave out the pairs of the form $(k.s, -k.s)$. We take all the pairs in column 1. The maximum number of such pairs is $\lceil (n - 1)/4 \rceil$. If the number of pairs in column 1 is $\lceil (n - 1)/4 \rceil$ then we put all these pairs in partition I and the rest in partition S. Otherwise, from successive columns we select pairs starting at the bottom of that column and move upwards until we get $\lceil (n - 1)/4 \rceil$ pairs (see Example 3). Later, we will show that it is indeed possible to collect $\lceil (n - 1)/4 \rceil$ pairs in this way.

The pairs in partition I are connected in such a way that if one node of a pair is connected to T_{+1}, then the other node of that pair is connected to T_{-1}. Now we have the following lemmas.

LEMMA 1. *Suppose $(u, n - u)$ is a pair in partition I in some column c. Then the pair $(u, n - u)$ can always be connected to the subtrees T_{+1} and T_{-1}.*

Proof. The pairs in column 1 are of the form $(k, -k)$. The node k is included in T_{+1} and the node $-k$ is included in T_{-1}.

Let the pair be of the form $(j.s + i, -j.s - i)$ in S_k. Then the pair $(i, -i)$ is in column 1 of which i is included in T_{+1} and $-i$ is included in T_{-1}, respectively. All the pairs $(p.s + i, -p.s - i)$, where $1 \leq p \leq j$ are in partition I and the node $p.s + i$ can be connected to T_{+1} by a link of type $+s$ and the node $-p.s - i$ can be connected to T_{-1} by a link of type $-s$.

The proof is similar for a pair of the form $(j.s - i, -j.s + i)$. ■

LEMMA 2. *The pairs in partition S can be connected to the subtrees T_{+s} and T_{-s}, after we connect the pairs in partition I to T_{+1} and T_{-1}.*

Proof. Let a pair in partition S be of the form $(j.s + (k - j), -j.s - (k - j))$. The pair $(j.s, -j.s)$ will be in partition S. So the node $j.s$ will be connected to T_{+s} and the node $-j.s$ will be connected to T_{-s}, respectively. Also all the pairs $(j.s + i, -j.s - i)$, $0 < i < k - j$, are in partition S and can be connected to T_{+s} and T_{+s} by links of type $+1$ and -1, respectively.

The proof is similar for a pair of the form $(j.s - (k - j), -j.s + (k - j))$. ■

LEMMA 3. *It is possible to get at least $\lfloor (n - 1)/4 \rfloor$ pairs in partitions I.*

Proof. We have shown that the diameter d of the graph is less than or equal to $\lfloor (n - 1)/4 \rfloor$. Total number of pairs for odd n is $(n - 1)/2$. Since we have to leave out only the pairs of the form $(k.s, -k.s)$ for partition S, the maximum number of pairs that we can have in partition $I = (n - 1)/2 -$ number of such pairs of the form $(k.s, -k.s)$. That is, $|I| \geq (n - 1)/2 -$ number of rows in the matrix $M \geq (n - 1)/2 - d \geq (n - 1)/2 - \lceil (n - 1)/4 \rceil = \lfloor (n - 1)/4 \rfloor$. ■

3.1.2. For Even n

The construction of the spanning tree in this case is slightly different. When we encounter the nodes in pairs the same procedure is followed as before. Only the node $n/2$ will not occur in a pair. The following cases can occur:

Case I: $n/2$ is in row k and is of the form $j.s + (k - j)$ or $j.s - (k - j)$, $0 < j < k$. We make partition I consisting of $\lceil (n - 1)/4 \rceil$ pairs provided that the node $n/2$ is not encountered in the process. Otherwise, partition I will have $\lfloor (n - 1)/4 \rfloor$ pairs and the node $n/2$. If $n/2$ is included in partition I, then we can connect this node to either T_{+1} or T_{-1} tree. On the other hand if $n/2 \in S$, then we can connect this node to either T_{+s} or T_{-s} tree.

Case II: $n/2$ is in row k and of the form $k.s$. In this case $n/2$ appears in column $2k$ and all columns $c > 2k$ will be empty. Here $n/2 \in S$. If the pair $(k.s + 1, -k.s - 1)$ is present in row $(k + 1)$ and is in partition S, to connect this pair properly we have to keep the pair $((k - 1).s + 1, -(k - 1).s - 1))$ in partition S (the node $k.s = n/2$ becomes a leaf node).

EXAMPLE 3. For $n = 14$, $s = 4$, we first construct the matrix M as follows:

$S1$: $\underline{(1, 13)}$ (4, 10)

$S2$: $\underline{(2, 12)}$ $\underline{(5, 9)}$ (3, 11) (8, 6)

$S3$: (—, —) (—, —) (—, —) (—, —) (7, —) (—, —)

The three underlined pairs are in partition I and others are in partition S. The spanning tree is shown in Fig. 2.

Let C_t denote the set of nodes which are destinations of the packets sent from node 0 at time t. After constructing the partitions I and S, the sets C_t's are generated by the following algorithm.

ALGORITHM C.

```
Step 1: The nodes in partition $I$ and $S$ are arranged in
    decreasing order of their distances from node 0.
Step 2: For $t = 0, 1, \ldots, \lceil (n - 1)/4 \rceil - 1$, do the following:
    If $n$ is odd then a pair of nodes from partition $I$ and
    a pair of nodes from partition $S$, taken in the above
    order, constitute $C_t$.
    If $n$ is even, then $C_t$'s are generated in the same way
    as above except when the node $n/2$ is encountered.
    Let the node $n/2$ be encountered at $t = t_1$. If another
    pair of nodes remain to be included in $C_t$'s, then
    $|C_{t_1}| = 3$, otherwise $|C_{t_1}| = 1$. When, $|C_{t_1}| = 3$, if
    $n/2 \in I$, then $C_{t_1}$ consists of $n/2$ and a pair in parti-
    tion $S$. If $n/2 \in S$, then $C_{t_1}$ consists of $n/2$ and a
    pair in partition $I$. ■
```

To effect the transmission of data packets from node 0, each packet will be associated with a *tag* depending on its destination node. If a node is in partition I and appears in

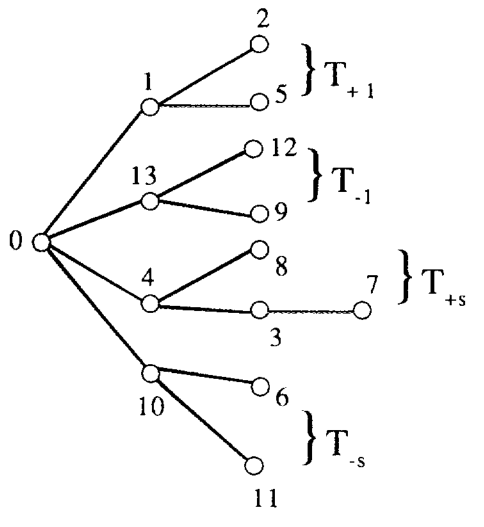

FIG. 2. Spanning tree for single node scatter.

the matrix M in the form $j.s + k$, $j, k > 0$ the packet destined to this node will have a *tag* like this:

link type				partition type
$+s$	$-s$	$+1$	-1	
j	0	k	0	I

Similarly a packet destined to a node $-j.s + k$ in partition S will have a *tag* like this:

link type				partition type
$+s$	$-s$	$+1$	-1	
0	j	k	0	S

The transmission of a packet at any node will be made only along a link of type for which there is a nonzero entry in the tag associated with the packet. Once it is decided to transmit a data packet along a particular link, the corresponding link type entry in its *tag* is decremented by unity.

We have to ensure that the packets destined to the nodes in partition I always travel through T_{+1} or T_{-1} and packets destined to the nodes in partition S always travel through T_{+s} or T_{-s}. We do this by sending every packet destined for a node in partition I along the links of type $+1$ or -1 first, until the corresponding link type's entry in the tag of the packet becomes zero. Then only, we transmit the packet using $+s$ or $-s$ links. The reverse procedure is followed for the packets destined to the nodes in partition S.

4. CONCLUSION

In this paper, we have given time-optimal algorithms for multiple node broadcast and single node scatter in a distributed loop network. Our algorithms are based on the assumptions of multiple link availability (MLA) and full duplex communication along each link. Also, to complete the multiple node broadcast in time $\lceil (n - 1)/4 \rceil$, we need proper synchronization among all the nodes of the network. There is no overhead associated with the running of this algorithm except an $O(n)$ storage requirement at each node. The required information for packet broadcasting will be computed once for all and will be stored in all the nodes of the network. The single node scattering algorithm also runs in time $\lceil (n - 1)/4 \rceil$ and hence is optimal with respect to time. The required information for packet scattering is computed once for all and stored at the source node. The resulting overhead is $O(n)$ storage at the source node. The total exchange or multiple node scatter is another important communication problem. Further work is being carried out to find if there exists a time-optimal algorithm for this problem in a distributed loop network.

Received November 15, 1993; revised July 19, 1994; accepted March 25, 1995

REFERENCES

1. B. Elspas and J. Turner, Graphs with circulant adjacency matrices. *J. Combin. Theory* **9,** 297–307 (1970).
2. S. Dutt and J. P. Hayes, Designing fault-tolerant systems using automorphisms. *J. Parallel Distrib. Comput.* **12,** 249–268 (1991).
3. D. P. Bertsekas, C. Ozveren, G. D. Stamoulis, P. Tseng, and J. N. Tsitsiklis, Optimal communication algorithm for hypercubes. *J. Parallel Distrib. Comput.* **11,** 263–275 (Apr. 1991).
4. D. P. Bertsekas, and J. N. Tsitsiklis, *Parallel and Distributed Computation: Numerical Methods*, Prentice-Hall, Englewood Cliffs, NJ, 1989.
5. J.-C. Bermond, and D. Tzvieli, Minimal-diameter double-loop networks: Dense optimal family. *Networks* **21,** 1–9 (Jan. 1991).
6. D. Tzvieli, Minimal diameter double-loop networks, I. Large infinite families. *Networks* **21,** 387–415 (July 1991).
7. R. K. Das, and B. P. Sinha, *Distributed Loop Networks.* Technical Report, Indian Statistical Institute, 1994.
8. K. Mukhopadhyaya, and B. P. Sinha, Optimal design and routing of distributed loop networks. *Proceedings of International Symposium on Circuits and Systems*, Singapore, Aug. 1991.

RAJIB K. DAS received a B.E. degree in electronics and telecommunication engineering from Jadavpur University in 1988 and an M.Tech. degree in computer science from the Indian Institute of Technology, Madras in 1990. Currently he is working toward his Ph.D. degree in computer science at the Indian Statistical Institute, Calcutta. His research interests include network topology and parallel and distributed processing.

BHABANI P. SINHA received a B.Sc.(Hons.) degree in Physics, B.Tech. and M.Tech. degrees in radiophysics and electronics, and a Ph.D. degree in computer science from the University of Calcutta in 1970, 1973, 1975, and 1979, respectively. In 1976, he joined the faculty of the Indian Statistical Institute, Calcutta, where he has been a Professor since 1987. During 1986–1987, he visited the Department of Computer Science, Southern Illinois University, as an associate professor. With a fellowship from the Alexander von Humboldt Foundation of West Germany, he visited Informatik Kolleg, GMD, Bonn during 1979–1981 and also the Department of Computer Science, University of Saarland, Saarbruecken in 1990. His current research interests include parallel algorithms and architectures, network topology, and computational geometry. He is a senior member of the Institution of Electrical and Electronics Engineers.