Short Notes

An Optimal Broadcasting Algorithm without Message Redundancy in Star Graphs

Jang-Ping Sheu, Chao-Tsung Wu, and Tzung-Shi Chen

Abstract—**Based on the Mendia and Sarkar's algorithm [8], we propose an optimal and nonredundant distributed broadcasting algorithm in star graphs. For an n-dimensional star graph, our algorithm takes $O(n \log_2 n)$ time and guarantees that all nodes in the star graph receive the message exactly once. Moreover, broadcasting m packets in a pipeline fashion takes $O(m \log_2 n + n \log_2 n)$ time due to the nonredundant property of our broadcasting algorithm.**

Index Terms—**Broadcasting, interconnection networks, star graphs.**

I. Introduction

Broadcasting is an important issue for numerous applications in parallel or distributed computing [4]. The communication schemes are characterized by the *time*, the number of time steps required, and the *traffic*, the total number of messages exchanged, to complete the communication [5], [7]. Hence, it is desirable to develop a scheme for broadcasting that minimizes both time and traffic. Broadcasting algorithms in star graphs have been studied in much research [1], [3], [8], [9]. For an n-dimensional star graph, using a sequence generated through a permutation network, the broadcasting algorithm proposed in [1] takes at most $3(n \log_2 n - n/2)$ time. The algorithm in [3], broadcasts a message by a scheme named recursively doubling and takes $O(n \log_2 n)$ time. The algorithm proposed in [8] works by recursively partitioning the original star graph into smaller star graphs with the same sizes for broadcasting in $O(n \log_2 n)$ time. All of the above algorithms are optimal on the one-port communication model; however, they have the same drawback that some message redundancy incurs the additional amount of traffic in star graphs. The algorithm in [9] solves this drawback, i.e., it sends or receives no redundant messages, and only needs to take $2n - 3$ time steps on the multi-port communication model.

Our purpose in this paper is to develop an optimal broadcasting algorithm without message redundancy in star graphs on the one-port communication model. The proposed algorithm here is based on the algorithm in [8]. The main difference between our algorithm and the algorithm in [8] is that we use a different scheme to recursively decompose the original star graph into smaller disjoint star graphs with the different sizes in order to fulfill the purpose of no message redundancy. Our proposed algorithm takes $O(n \log_2 n)$ time in an n-dimensional star graph and guarantees that all nodes receive the broadcast-message exactly once during the broadcasting; that is, the total amount of traffic is minimal. Moreover, because of the characteristic of message nonredundancy, our algorithm can broadcast a stream of m packets in $O(m \log_2 n + n \log_2 n)$ time in a pipeline fashion.

Manuscript received August 4, 1993; revised August 25, 1994. This work was supported by the National Science Council of the Republic of China under Grant NSC 83-0408-E-008-018.

The authors are with the Department of Computer Science and Information Engineering, National Central University, Chung-Li 32054, Taiwan.

IEEE Log Number 9411969.

II. Preliminaries

In this section, we will introduce some definitions and notations related to star graphs and use them throughout the paper. A permutation of n distinct symbols in the set $\{1, 2, \cdots, n\}$ is represented by $P = s_1 s_2 \cdots s_n$ where $s_i, s_j \in \{1, 2, \cdots, n\}$ and $s_i \neq s_j$ for $i \neq j$ and $1 \leq i, j \leq n$. A star graph with dimension n is an undirected graph in which the nodes correspond to the elements of the permutations of $\{1, 2, \cdots, n\}$ and the edges correspond to the actions of generators. We formally define them as follows [1], [2].

Definition 1 [Generator g_i]:

Given a permutation $P = s_1 s_2 \cdots s_n$, we define the *generator* g_i as the function that interchanges the symbol s_i with the symbol s_1 in P for $2 \leq i \leq n$.

□

Definition 2 [Star Graph]:

An undirected *star graph* with dimension n is denoted by $S_n = (P_n, E_n)$ where the set of vertices P_n is defined as $\{s_1 s_2 \cdots s_n | s_i, s_j \in \{1, 2, \cdots, n\}, s_i \neq s_j \text{ for } i \neq j, 1 \leq i, j \leq n\}$ and the set of edges E_n is defined as $\{(v_1, v_2) | v_1, v_2 \in P_n, v_1 \neq v_2,$ such that $v_1 = g_i(v_2)$ for $2 \leq i \leq n\}$.

□

In other words, any two nodes v_1 and v_2 are connected by an undirected edge if and only if the permutation corresponding to the node v_2 can be obtained from that of v_1 by interchanging the symbol s_i of v_1 with the symbol s_1 of v_1 for $2 \leq i \leq n$. We use the notation S_n to represent an n-dimensional star graph in this paper. Note that star graphs are edge and vertex symmetric [2]. Moreover, S_n is a regular graph with degree $n - 1, n!$ vertices, and $(n-1)n!/2$ edges. Each node in S_n is connected to $n - 1$ adjacent nodes by $n - 1$ edges. The edges are named *dimensions* which are similar to that in hypercubes [6].

Definition 3 [ith Dimension]:

The ith *dimension* of a node u in S_n is defined as the edge along which the node u connects with the node $v = g_i(u)$ for $2 \leq i \leq n$.

□

Definition 4 [Substar Graph] [3]:

Let $p_{i+1} p_{i+2} \cdots p_n$ be a permutation of $n - i$ distinct symbols in $\{1, 2, \cdots, n\}$ for $1 \leq i \leq n$. Then the *substar graph* denoted by $S_i(p_{i+1} p_{i+2} \cdots p_n)$ is a subgraph (V_{S_i}, E_{S_i}) of S_n, where V_{S_i} is the set of all nodes with the same last $n - i$ symbols $p_{i+1}, p_{i+2}, \cdots, p_n$ and E_{S_i} is the set of edges incident with any two of the nodes in V_{S_i}.

□

We can easily examine that the $S_i(p_{i+1} p_{i+2} \cdots p_n)$ is an i-dimensional star graph by the definition of star graphs. Throughout this paper, we use the notations substar S_i to denote the $S_i(p_{i+1} p_{i+2} \cdots p_n)$. For example, the 4-star S_4 is composed of four disjoint 3-substars $S_3(1), S_3(2), S_3(3)$, and $S_3(4)$. Clearly, each 3-substar is isomorphic to one another. We can independently regard each 3-substar as an S_3. Hence the S_n can be simply decomposed into n substars S_{n-1} by varying the last position with n distinct symbols in $\{1, 2, \cdots, n\}$.

III. An Optimal Broadcasting Algorithm

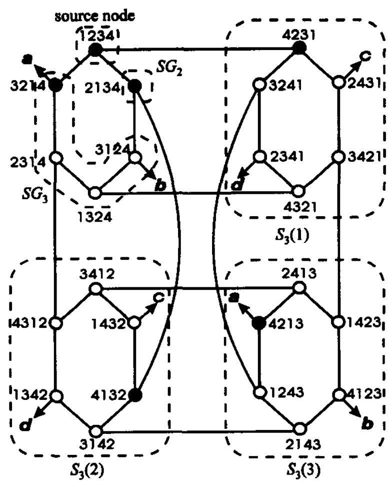
We shall discuss how the proposed distributed algorithm works in details and analyze its time complexity in this section. A node is called a *source node* when this node wants to broadcast a specific message to all the other nodes in an n-star. Since star graphs are vertex symmetric, without loss of generality, we can consider the node $12\cdots n$ as the source node in an n-star.

At the beginning, given an n-star, we briefly describe our broadcasting algorithm proposed here consisting of three phases in below. In our proposed algorithm, while receiving the source message, nodes start with their individual process of broadcasting, performing their three-phase algorithm. In Phase 1, we claim that there exist $n-2$ designated intermediate nodes in the $S_{n-1}(n)$ which can receive the message from the source node. As soon as the source node or one of these intermediate nodes completes the operations in Phase 1, it can at once apply Phase 2 to send the message to one of the $n-1$ substars S_{n-1} except the $S_{n-1}(n)$. After performing Phase 1 and Phase 2, only the intermediate nodes continue to apply Phase 3. In Phase 3, the intermediate nodes in the $S_{n-1}(n)$ should send the message to other nodes in the $S_{n-1}(n)$ and avoid sending a redundant message to the nodes held the message.

The detail descriptions of our proposed algorithm with three phases are given as follows.

Phase 1: In what follows we shall describe how the source node can distribute the message to the intermediate nodes. To make our algorithm optimal, the source node must generate such $n-2$ intermediate nodes in $O(\log_2 n)$ time in the $S_{n-1}(n)$ instead of generating them step by step in $O(n)$ time. These intermediate nodes can be generated by the way proposed in [8]. For this purpose, a variable *Cardinality* in each node to signify the generating order of those intermediate nodes is defined in [8]. The *Cardinality* of the node will be set to 1 as long as it is viewed as a source node. When any other node, the intermediate node, receives the source message, its *Cardinality* will be set to the dimension of the edge along which the intermediate node received the message. The node whose *Cardinality* is i will be denoted by C_i. How Phase 1 works during broadcasting in an S_n is illustrated below. In step i for $1 \le i \le \lceil \log_2(n-1) \rceil$, nodes C_j will send the message to the other intermediate nodes $C_{j+2^{i-1}}$ along the $(j+2^{i-1})$th dimension if $j + 2^{i-1} \le n-1$, for $1 \le j \le 2^{i-1}$. By this way, the intermediate nodes of an S_n can be generated continuously until the last intermediate node C_{n-1} is generated. As a result, the source node in an S_n takes $\lceil \log_2(n-1) \rceil$ time steps to distribute the message to these $n-2$ intermediate nodes. The above described approach is simply called the *intermediate node distribution approach.*

Phase 2: Let $C_1 = 12\cdots n$ be the source node in an S_n. After applying the *intermediate node distribution approach* in Phase 1, each of the $n-2$ intermediate nodes C_j has the symbol j in its first position, $2 \le j \le n-1$. We shall state this property in Lemma 1 which has been proven in [8]. Directly derived from Lemma 1, the source node and the $n-2$ intermediate nodes in this phase can broadcast the message to the other $n-1$ substars S_{n-1} except the $S_{n-1}(n)$ in the S_n along the nth dimension in one time step. Thus, the source node in S_n takes $\lceil \log_2(n-1) \rceil + 1$ time steps to broadcast the message to the other $n-1$ substars S_{n-1} by finishing Phase 1 and Phase 2. Hence, each of the $n-1$ $(n-1)$-substars is able to broadcast the message within itself recursively by applying our three-phase broadcasting algorithm. The source node C_1 in the S_n will terminate in our algorithm when it completes the broadcasting in Phase 2.



● : the node with the source message

Fig. 1. The substar graphs and subgraphs in an S_4.

For example, consider the substar $S_3(4)$ in which there are three nodes held the source message after performing Phase 1: the source node $C_1 = 1234$, and two intermediate nodes $C_2 = 2134$ and $C_3 = 3214$ as shown in Fig. 1. Within Phase 2, the above three nodes can send the message to the respective nodes 4231, 4132, and 4213 along the 4th dimension. Thus the three substars $S_3(1), S_3(2)$, and $S_3(3)$ can recursively broadcast within themselves.

Phase 3: We shall describe how the intermediate nodes in the $S_{n-1}(n)$ deliver the message continuously in this phase. There are $n-1$ nodes in the $S_{n-1}(n)$ holding the source message: the source node and $n-2$ intermediate nodes. We can decompose the substar $S_{n-1}(n)$ into $n-2$ substars $S_{n-2}, n-3$ substars $S_{n-3}, \cdots$, an S_1, and the source node. To make our description simple, the $n-2$ substars $S_{n-2}, n-3$ substars $S_{n-3}, \cdots$, and an S_1 are denoted as symbols $SG_{n-1}, SG_{n-2}, \cdots, SG_3$, and SG_2, respectively. A node which needs to broadcast a message to all the other nodes in an SG_j is also called a *source node* in the $SG_j, 2 \le j \le n-1$. We shall prove it in Lemma 2 that each intermediate node C_j is the source node in the SG_j, for $2 \le j \le n-1$. For avoiding the message redundancy, each source node in $SG_j, 2 \le j \le n-1$, should independently broadcast the message within itself.

We shall describe how the source node broadcasts the message in a particular SG_m as follows. Note that an SG_m has $m-1$ substars S_{m-1}. Given a source node in an SG_m, it is in one of the $m-1$ substars S_{m-1} and needs to broadcast the message to the other $m-2$ substars S_{m-1} in the SG_m. Applying the same *intermediate node distribution approach* as in Phase 1, the source node in SG_m will distribute the message to $m-2$ intermediate nodes and complete the three-phase algorithm. On the other hand, the $m-2$ intermediate nodes will send the message to the other $m-2$ substars S_{m-1} by applying their Phase 2. We shall prove that the message of the source node in an SG_m can be broadcast through the $m-2$ intermediate nodes to other $m-2$ substars S_{m-1} in $\lceil \log_2(m-1) \rceil + 1$ time steps in Lemma 3. Thus each of the $m-2$ substars S_{m-1} with a source node can recursively broadcast the message within itself.

Moreover, the substar S_{m-1} with the source node in the SG_m can be also decomposed into $SG_{m-1}, SG_{m-2}, \cdots, SG_2$, and the source node. Of course, by Lemma 2, each SG_i, $2 \le i \le m-1$, has an intermediate node as its source node. Hence, we can recursively apply the Phase 3 to each intermediate node for sending the message to each SG_i for $2 \le i \le m-1$.

For making the description of Phase 3 clear, consider a 5-star as an example. Fig. 2 shows the $S_4(5)$ out of an S_5 after executing Phase 1 and Phase 2 in our broadcasting algorithm. The SG_4, SG_3, and SG_2 decomposed from the $S_4(5)$ are also shown in Fig. 2. In the $S_4(5)$, the intermediate nodes 21345, 32145, and 41325 are the source nodes in SG_2, SG_3, and SG_4, respectively. In the SG_4, the source node 41325 distributes the message to two intermediate nodes 14325 and 31425. Then the source node 41325 completes its Phase 3 and the two intermediate nodes apply their Phase 2 to send the message to nodes 24315 and 21435 in $S_3(15)$ and $S_3(35)$, respectively. So, each of the $S_3(15)$ and the $S_3(35)$ has a source node to broadcast the message within themselves recursively.

We shall formally describe our three-phase distributed broadcasting algorithm, shown at the bottom of the page, without message redundancy. In our algorithm, nodes held the message send the broadcasting requests to other nodes. Every node individually starts to perform its broadcasting algorithm while receiving a request. A request Broadcast(*Data, Edge, StarDimension, Steps*) consists of the following four parameters.

1) *Data*	Message to be broadcast.
2) *Edge*	Dimension or edge along which request was sent or received.
3) *StarDimension*	Dimension of a substar that the node will be broadcast to.
4) *Steps*	Used for determining the ordering of the intermediate nodes.

The broadcasting in an n-star is simply initialized by the source node with issuing the call: Broadcast(*Message*, n, n, 0), where the *Message* is the message to be broadcast from the source node and n is the allocated dimension of the star graph involved.

Fig. 3 illustrates the broadcasting in an S_4 by applying our algorithm. The number-labeled edges denote the time steps during broadcasting. It can be shown that our algorithm completes the broadcasting of an S_4 in 6 time steps, and receives or sends no redundant messages. Fig. 4 shows the broadcasting tree generated by applying our algorithm in an S_4.

```
Algorithm Broadcast($Data, Edge, StarDimension, Steps$)
begin
    get $Data$ from buffer;
    if ($StarDimension \leq 1$) then terminate;
    /* terminate while broadcasting to itself */
Phase 1:
    begin
      if ($Edge \geq StarDimension$) then $Cardinality = 1$;
      else $Cardinality = Edge$;
      for $i = (Steps + 1)$ to $\lceil \log_2 (StarDimension - 1) \rceil$ do
      begin
          $Dimension = Cardinality + 2^{i-1}$;
          if ($Dimension < StarDimension$) then
              Broadcast($Data, Dimension, StarDimension, i$)
              along the $Dimension$-th dimension;
          end;
      end; /* end of Phase 1 */
Phase 2:
    begin
      Broadcast($Data, StarDimension, StarDimension - 1$, 0)
      along the $StarDimension$-th dimension;
    end; /* end of Phase 2 */
Phase 3:
    begin
      if this node is an intermediate one /* $Cardinality > 1$ */ then
      begin
          $Cardinality = 1$;
          $Steps = 0$;
          $StarDimension = Edge$;
          /* here the intermediate nodes serve as the source nodes in $SG_{Edge}$ */
          for $i = (Steps + 1)$ to $\lceil \log_2 (StarDimension - 1) \rceil$ do
          begin
              $Dimension = Cardinality + 2^{i-1}$;
              if ($Dimension < StarDimension$) then
                  Broadcast($Data, Dimension, StarDimension, i$)
                  along the $Dimension$-th dimension;
          end;
      end;
    end; /* end of Phase 3 */
end; /* end of Algorithm */
```

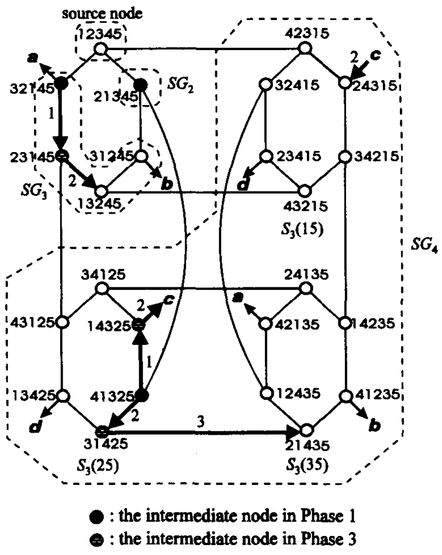


Fig. 2. The Phase 3 in the $S_4(5)$ out of an S_5.

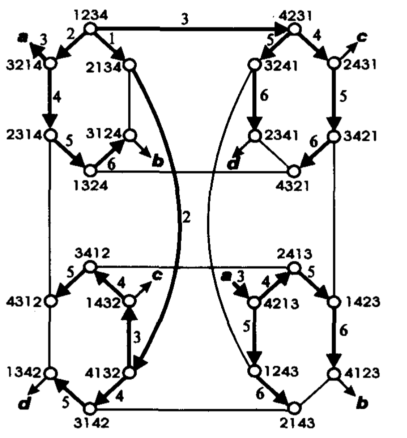


Fig. 3. Time steps of broadcasting in an S_4.

Before we prove that our broadcasting algorithm is nonredundant and optimal, some notations used in the proofs are first defined. Let the function $Last_i(U)$ obtain the last i symbols of $U = u_1u_2\cdots u_{m-i+1}u_{m-i+2}\cdots u_m$ which is a permutation of $\{1, 2, \cdots, m\}$; that is, $Last_i(U) = u_{m-i+1}u_{m-i+2}\cdots u_m$. For the source node $C_1 = u_1u_2\cdots u_m$ in a particular S_m, the $m-2$ intermediate nodes are of the form $u_jx_1\cdots x_{j-1}u_{j+1}\cdots u_m$ where the $x_1\cdots x_{j-1}$ is a particular permutation of symbols $u_1, u_2, \cdots, u_{j-1}$, for $2 \le j \le m-1$. The following Lemma and Corollary will show the two properties about the intermediate nodes in an S_m. Note that not all of the nodes which satisfy the two properties can be the intermediate nodes.

Lemma 1: [8] If $C_1 = u_1u_2\cdots u_m$ is the source node in an S_m, each intermediate node C_j has the symbol u_j in its first position, for $2 \le j \le m-1$, after finishing Phase 1.

Corollary 1: If $C_1 = u_1u_2\cdots u_m$ is the source node in an S_m, then each intermediate node C_j has the property that $Last_{m-j}(C_j) = Last_{m-j}(C_1)$ and $Last_{m-j+1}(C_j) \neq Last_{m-j+1}(C_1)$, for $2 \le j \le m-1$, after finishing Phase 1.

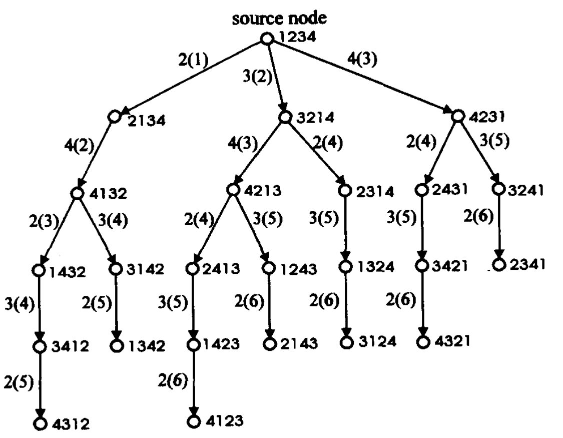


Fig. 4. The broadcasting tree in an S_4.

Proof: The reader is referred to the proof of Lemma 2 in [8] for the proof of this Corollary. □

Lemma 2: If $C_1 = u_1u_2\cdots u_m$ is the source node in an S_m or SG_m, each intermediate node C_j is the source node in the SG_j, for $2 \le j \le m-1$, after finishing Phase 1 or Phase 3.

Proof: We can decompose the substar $S_{m-1}(u_m) = S_{m-1}(Last_1(C_1))$ into a substar $S_{m-2}(Last_2(C_1))$ and the subgraph $S_{m-1}(Last_1(C_1)) - S_{m-2}(Last_2(C_1))$ which is composed of $m-2$ substars S_{m-2} and is denoted as the SG_{m-1}. Each node U in the SG_{m-1} has the property that $Last_1(U) = Last_1(C_1)$ and $Last_2(U) \neq Last_2(C_1)$. It has been shown in Corollary 1 that each intermediate node C_j has the property that $Last_{m-j}(C_j) = Last_{m-j}(C_1)$ and $Last_{m-j+1}(C_j) \neq Last_{m-j+1}(C_1), 2 \le j \le m-1$, when applied Phase 1 or Phase 3 in an S_m or SG_m. Clearly, only the intermediate node C_{m-1} is in the SG_{m-1} and the others are in the $S_{m-2}(Last_2(C_1))$. Thus, the intermediate node C_{m-1} is the source node in the SG_{m-1}. Similarly, the $S_{m-2}(Last_2(C_1))$ can be also decomposed into the $S_{m-3}(Last_3(C_1))$ and the $SG_{m-2} = S_{m-2}(Last_2(C_1)) - S_{m-3}(Last_3(C_1))$ which is composed of $m-3$ substars S_{m-3}. Only the intermediate node C_{m-2} can be the source node in the SG_{m-2}. If we decompose the SG_{m-2} continuously, the $S_2(Last_{m-2}(C_1))$ can be finally decomposed into the source node C_1 and the SG_2 in which the intermediate node C_2 is the source node.

Therefore, each intermediate node C_j is the source node in the $SG_j, 2 \le j \le m-1$ after executing Phase 1 or Phase 3. □

Lemma 3: By applying Phase 3, the source node in a particular SG_m can broadcast the message to the other $m-2$ substars S_{m-1} in $\lceil \log_2 (m-1) \rceil + 1$ time steps, where $2 \le m \le n-1$.

Proof: By Lemma 2, the source node in an SG_m is the node C_m previously used as an intermediate node. Because the two properties in Lemma 1 and Corollary 1 also hold in the SG_m, the source node C_m can distribute the message to $m-2$ intermediate nodes in $\lceil \log_2 (m-1) \rceil$ time steps in Phase 3. The source node C_m terminates its broadcasting algorithm after distributing the message to $m-2$ intermediate nodes. Then the $m-2$ intermediate nodes can send the messages to the other $m-2$ substars S_{m-1} along their mth dimensions in their Phase 2, respectively; it only takes one time step. Thus by applying Phase 3, the source node in the SG_m can broadcast the message to the other $m-2$ substars S_{m-1} in $\lceil \log_2 (m-1) \rceil + 1$ time steps. □

Theorem 1: The proposed broadcasting algorithm can send a message from the source node to each of the other nodes in the S_n exactly once.

Proof: We shall prove this theorem by mathematical induction.

Basis: Clearly, the theorem is true for $n = 1$.

Induction hypothesis: Assume that the theorem holds for $S_n, 1 \le n \le k-1$.

Induction: In this step, we shall prove that the theorem also holds for S_k. Assume that $u_1u_2\cdots u_k$ is the source node in the S_k. After finishing Phase 1 and Phase 2 of our algorithm, we can decompose the S_k as:

$$\begin{aligned} S_k &\rightarrow S_{k-1}(u_k) \cup \left(\bigcup_{1\le i\le k-1} S_{k-1}(u_i) \right) \\ &\rightarrow \left(\bigcup_{2\le i\le k-1} SG_i \cup \text{source node} \right) \\ &\quad \cup \left(\bigcup_{1\le i\le k-1} S_{k-1}(u_i) \right). \end{aligned} \tag{1}$$

Directly derived from Lemma 1, each of the $S_{k-1}(u_i)$ for $1 \le i \le k-1$ has a source node after applying Phase 2. Hence, these $k-1$ substars S_{k-1} in expression (1) hold the criteria by induction hypothesis. We have shown it in Lemma 2 that each of the $k-2$ intermediate nodes is the source node in an SG_i in expression (1), $2 \le i \le k-1$. After applying Phase 3, we have proven that the source node in a particular SG_i can broadcast the message to the other $i-2$ substars S_{i-1} in Lemma 3, $2 \le i \le k-1$. So, we can decompose a particular SG_i as:

$$\begin{aligned} SG_i \rightarrow &\left(\bigcup_{2\le j\le i-1} SG_j \cup \text{source node} \right) \\ &\cup \left(\bigcup_{2\le j\le i-1} S_{i-1}(u_j u_{i+1} u_{i+2} \cdots u_k) \right). \end{aligned} \tag{2}$$

Hence, each S_{i-1} in expression (2) also holds the criteria. Any SG_i in expression (1) can be decomposed into substars with dimensions less than $k-1$. Therefore, by the induction hypothesis we can prove that our algorithm can broadcast the message from the source node in the S_n to each of the other nodes exactly once. □

Theorem 2: Given an n-star, our proposed broadcasting algorithm completes the broadcasting in $\Theta(n \log_2 n)$ time.

Proof: It has been shown that the source node in an S_n broadcasts the message to each of the $n-1$ substars S_{n-1} in the first two phases in $\lceil \log_2 (n-1) \rceil + 1$ time steps. Next, in Lemma 3, the source node in the SG_i by applying Phase 3 can also broadcast the message to the other $i-2$ substars in the $SG_i, 2 \le i \le n-1$. They take at most $\lceil \log_2 (n-2) \rceil + 1$ time steps, which is the same as that taken by the other substars S_{n-1} to broadcast the message to $n-1$ substars S_{n-2}. In other words, our algorithm can reduce the broadcasting problem size in an n-star by at least 1 in $\lceil \log_2 (n-1) \rceil + 1$ time steps recursively. Therefore, the following equation gives the number of time steps required for broadcasting a message in an n-star.

$$\begin{aligned} \text{Total time steps} &= \sum_{i=2}^{n} (\lceil \log_2 (i-1) \rceil + 1) \\ &= (n-1)\lfloor \log_2 (n-1) \rfloor - 2^{\lfloor \log_2 (n-1) \rfloor} \\ &\quad + 2n - 1 \\ &= \Theta(n \log_2 n). \end{aligned}$$ □

TABLE I
AMOUNT OF TRAFFIC COMPARISONS WITH OTHER BROADCASTING ALGORITHMS

dimension of star graphs	T_A	T_B	T_C	total traffic improved $\frac{T_A - T_C}{T_A} \times 100\%$	total traffic improved $\frac{T_B - T_C}{T_B} \times 100\%$
2	1	1	1	0.0000	0.0000
3	9	6	5	44.4444	16.6667
4	51	29	23	54.9020	20.6897
5	291	152	119	59.1065	21.7105
6	1851	921	719	61.1561	21.9327
7	13371	6458	5039	62.3140	21.9727
8	109131	51677	40319	63.0545	21.9788
9	996171	465108	362879	63.5726	21.9796
10	10068171	4651097	3628799	63.9577	21.9797

An optimal broadcasting algorithm in an n-star takes the time complexity $O(n \log_2 n)$ [8]. Hence our proposed broadcasting algorithm which takes $\Theta(n \log_2 n)$ time steps is optimal.

IV. PERFORMANCE ANALYSIS

In multicomputer systems or communication networks, interprocessor communication occurs when a node sends/receives a message through the edge to/from its neighbor, respectively. While running an application, communication with high probability may cause message transmission delay due to the heavy traffic incurred by a broadcasting scheme. Especially, it is the case that a node wants to broadcast a lot of messages or packets divided by a large amount of data to the other nodes [6]. Thus the performance of such an application which uses the broadcasting as a basic step highly depends on the amount of traffic incurred by the broadcasting algorithm. Because our proposed algorithm has the nonredundant property, this yields the minimum traffic for broadcasting a message from the source node to all the others.

We exhibit the performance of our algorithm by comparing the total amount of traffic with two other algorithms proposed by Akl, Qiu, and Stojmenovic [3], and Mendia and Sarkar [8] as follows. We compare with those algorithms since they are all time optimal on the one-port communication model and run in a recursive manner as our algorithm does. First, given an n-star for $n \ge 2$, the total amount of traffic for the algorithm proposed in [3] can be directly derived and be expressed as below.

$$T_A = \sum_{i=2}^{n} (3i-5)(i-1)! \quad .$$

Next the total amount of traffic for the algorithm proposed in [8] can be also directly derived and be expressed as below.

$$T_B = \sum_{i=2}^{n} \frac{(2i-3)n!}{i!}.$$

Finally, our proposed algorithm always produces the minimum amount of traffic

$$T_C = n! - 1.$$

Obviously, $T_A \ge T_C$ and $T_B \ge T_C$. In terms of the total amount of traffic illustrated in Table I, our algorithm has greater improvement over the algorithms in [3] and [8]. There exists a large amount of traffic in their algorithms since they use intermediate nodes repeatedly in the process of broadcasting. Hence our algorithm can significantly improve the performance when applications use the broadcasting frequently.

There is another merit of our algorithm while a node needs to broadcast m packets to all the others. By applying the broadcasting algorithms in [3] and [8], broadcasting the first packet to all the others requires to take $O(n \log_2 n)$ time. Then, the second packet can be consecutively broadcast until the source node completes the broadcasting of the first packet in a recursive manner. This is because their algorithms have to perform the broadcasting one by one; that is, it can not be performed in a pipeline fashion. In total, broadcasting the m packets can be completed in $O(mn \log_2 n)$ time. In contrast, because of the message nonredundancy in our algorithm, the source node can start to broadcast the next packet while the first two phases of our algorithm are completed. Since performing the first two phases of our broadcasting algorithm takes $O(\log_2 n)$ time, the m packets can be broadcast in $O(m \log_2 n + n \log_2 n)$ time in a pipeline fashion. Hence our nonredundant broadcasting algorithm takes less time and produces less traffic than the redundant ones for broadcasting a stream of packets.

V. Conclusions

In this paper, we proposed a distributed broadcasting algorithm with time and traffic optimum in star graphs on the one-port communication model. By recursively partitioning a star graph into smaller disjoint substar graphs, our algorithm can broadcast a message to all the other nodes in the given n-star in $O(n \log_2 n)$ time. We also showed the traffic improvement of our algorithm over two other algorithms proposed by [3] and [8]. Besides, our algorithm is more efficient than the above algorithms while broadcasting a stream of packets.

References

[1] S. B. Akers, D. Harel, and B. Krishnamurthy, "The star graph: An attractive alternative to the n-cube," in *Proc. 1987 Int. Conf. Parallel Process.*, Aug. 1987, pp. 393–400.
[2] S. B. Akers and B. Krishnamurthy, "A group-theoretic model for symmetric interconnection networks," *IEEE Trans. Comput.*, vol. 38, pp. 555–565, Apr. 1989.
[3] S. G. Akl, K. Qiu, and I. Stojmenovic, "Fundamental algorithms for the star and pancake interconnection networks with applications to computational geometry," *Networks*, vol. 23, no. 4, pp. 215–225, July 1993.
[4] R. Dechter and L. Kleinrock, "Broadcast communications and distributed algorithms," *IEEE Trans. Comput.*, vol. 35, pp. 210–219, Mar. 1986.
[5] S. M. Hedetniemi, S. T. Hedetniemi, and A. L. Liestman, "A survey of gossiping and broadcasting in communication networks," *Networks*, vol. 18, no. 4, pp. 319–349, 1988.
[6] S. L. Johnsson and C.-T. Ho, "Optimum broadcasting and personalized communication in hypercubes," *IEEE Trans. Comput.*, vol. 38, pp. 1249–1268, Sept. 1989.
[7] X. Lin and L. M. Ni, "Multicast communication in multicomputer networks," *IEEE Trans. Parallel, Distrib. Syst.*, vol. 4, pp. 1105–1117, Oct. 1993.
[8] V. E. Mendia and D. Sarkar, "Optimal broadcasting on the star graph," *IEEE Trans. Parallel, Distrib. Syst.*, vol. 3, pp. 389–396, July 1992.
[9] J. P. Sheu, W. H. Liaw, and T. S. Chen, "A broadcasting algorithm in star graph interconnection networks," in *Proc. 1992 Int. Conf. Parallel, Distribut. Syst.*, Hsinchu, Taiwan, Dec. 1992, pp. 204–210.

Performance Characterization of the Tree Quorum Algorithm

Her-Kun Chang and Shyan-Ming Yuan

***Abstract*—The tree quorum algorithm, which logically organizes the sites in a system to a tree structure, is an efficient and fault-tolerant solution for distributed mutual exclusion. In this paper, the performance characteristics of the tree quorum algorithm is analyzed. A refinement algorithm is proposed to refine a logical tree structure by eliminating nodes or subtrees which do not improve the performance. Thus the refined tree performs better than the original.**

***Index Terms*—Distributed mutual exclusion, tree quorum algorithm, availability, communication cost.**

I. Introduction

A distributed system consists of a set of sites which are loosely coupled by a computer network. One advantage of distributed systems is resource sharing. That is the resources in a distributed system can be shared among the sites in the system. Examples of sharable resources are memory, peripheral, CPU, clock, etc. The sites in a distributed system may issue requests to a shared resource at arbitrary time. When two or more sites intended to access the same resource, a conflict occurs. A mechanism is required to synchronize conflicting requests so that at most one site is allowed to access the resource at any time instant. This problem is known as distributed mutual exclusion [1]–[11]. A survey of various algorithms for mutual exclusion can be found in [6] and a simple taxonomy for distributed mutual exclusion algorithms was reported in [7].

A central controller can be used to control mutually exclusive access to a shared resource. All requests intended to the resource are sent to the controller and scheduled by the controller. Using a central controller is simple and easy to implement. However, the controller is vulnerable to site failure. When the controller fails, no access to the resource is allowed, i.e., the entire system is *halted*. It is desirable to reduce the probability that the system is halted by using more than one sites to participate the decision making. For example, majority consensus [11] can be used to achieve mutual exclusion wherein a site is allowed to access the resource if it can get permissions from a majority of all participating sites.

Majority consensus can tolerate at most $N/2$ sites failures, where N is the number of participating sites. On the other hand, the communication overhead is costly, since at least N messages ($N/2$ for request and $N/2$ for reply) are required to be exchanged. Several algorithms try to reduce the communication overhead by imposing logical structures to the sites [1], [4]. The tree quorum algorithm [1], which logically organizes the sites to a tree structure, can reduce the number of messages exchanged to $O(\log N)$ in the best case. In this paper, the performance characteristics of the tree quorum algorithm is analyzed.

The *availability*, which is defined to be the steady-state probability that the system is up (not halted), is usually used to evaluate a distributed algorithm. Another important performance measure for a distributed algorithm is its communication cost. Certainly, the purpose

Manuscript received November 1, 1992; revised December 8, 1993. This work was supported in part by NSC under Grant NSC82-0408-E009-053.

The authors are with Department of Computer and Information Science, National Chiao Tung University, Hsinchu, Taiwan 30050.

IEEE Log Number 9411968.