Toward Optimal Broadcast in a Star Graph Using Multiple Spanning Trees

Yu-Chee Tseng, *Member, IEEE Computer Society,*
and Jang-Ping Sheu, *Member, IEEE Computer Society*

Abstract—In a multicomputer network, sending a packet typically incurs two costs: start-up time and transmission time. This work is motivated by the observation that most broadcast algorithms in the literature for the star graph networks only try to minimize one of the costs. Thus, many algorithms, though claimed to be optimal, are only so when one of the costs is negligible. In this paper, we try to optimize both costs simultaneously for four types of broadcast problems: *one-to-all* or *all-to-all* broadcasting in an *n*-star network with either *one-port* or *all-port* communication capability. As opposed to earlier solutions, the main technique used in this paper is to construct from a source node multiple spanning trees, along each of which one partition of the broadcast message is transmitted.

Index Terms—All-to-all broadcast, collective communication, multicomputer networks, one-to-all broadcast, parallel architecture, routing, star graph.

1 Introduction

The star graph interconnection network, since being proposed in [1], is receiving increasing attention in the literature. A large number of references can be found in studying the star graph regarding its topological properties [6], [7], [15], embedding capability [10], [18], fault-tolerant capability [3], and even the construction of incomplete stars [11]. Among the efforts in studying the star graph, one of the central issues is around the various versions of broadcasting problems, such as one-to-all broadcast [2], [12], [13], [14], [16], [17] and all-to-all broadcast [8], [13].

In this paper, we study the one-to-all and all-to-all broadcast problems in a star network using packet-switching (or store-and-forward) technique. We consider the network with *one-port* or *all-port* communication capability. Following the formulation of many works (e.g., [9]), we assume that there are two kinds of cost, namely *start-up time* and *transmission time*, associated with the communication. Specifically, sending a packet of b bytes along a link takes $T_s + bT_c$ time, where T_s is the time to initialize (or start-up) the communication link and T_c is the latency to transmit a byte. This work is motivated by the observation that most broadcast algorithms in the literature for star graph networks only try to minimize either the start-up cost or the transmission cost, but both. Typically, the start-up time is significant in current machines, while the transmission time should not be ignored when the packet is long. Thus, many broadcasting algorithms, though claimed to be optimal, are only so when one of the costs is negligible.

In this work, we try to optimize both start-up and transmission costs simultaneously. As opposed to earlier solutions, the main technique used in this paper is to construct from a source node multiple spanning trees, along each of which one partition of the broadcast message is transmitted. For one-to-all broadcast, we propose a new spanning tree in an n-star that has the nice property that $n - 1$ copies of such trees can be embedded simultaneously in the network with an edge congestion of at most 2. By concurrently transmitting data along these trees in a pipelined manner, our results improve over the scheme of [16] (under the all-port model) and schemes of [2], [12], [13], [17] (under the one-port model) by orders of $O(n)$ and $O(\log n)$, respectively, in transmission time. Under the one-port model, the recent result by [14] achieves the same time complexity as ours, but the broadcast message will need to be sliced into an impractically large ($\approx n!$) number of segments. Section 6 gives detailed comparisons. For a quick overview and comparison, see Table 1.

For all-to-all broadcast, we propose a general solution which can be developed based on any spanning tree in the network. As long as the spanning tree has an optimal height, our algorithms achieve optimal start-up time and transmission time under the all-port model, and optimal transmission time under the one-port model. One main contribution of this result is its simplicity and generality. Existing algorithms [8], [13] all try to optimize the transmission time only. Our results reduce the high start-up cost of [8] under the all-port model from an order of $O((n - 1)!)$ to $O(n)$, and that of [13] under the one-port model from an order of $O(n!)$ to $O(n^2)$. For a quick overview and comparison, see Table 2.

To the best of our knowledge, this is the first work reporting the possibility of embedding multiple ($O(n)$) spanning trees in an n-star, while at the same time keeping the edge congestion constant. Similar results for hypercubes can be found in [19]. The technique of using multiple spanning trees for broadcasting has been used in [9] for hypercube networks and in [4] for 2D meshes, but no comparable result has been reported for the star graphs yet.

Section 2 gives some preliminary results. Section 3 constructs the spanning trees that will be used throughout the paper. Our one-to-all and all-to-all broadcast algorithms are then presented in Section 4 and Section 5, respectively. We compare our results with other related works in Section 6. Conclusions are drawn in Section 7.

2 Preliminaries

An *n-dimensional star graph*, also referred to as *n-star* or S_n, is an undirected graph consisting of $n!$ nodes (or vertices). Each node is assigned a unique label $x_0 x_1 \cdots x_{n-1}$, which is a permutation of n symbols $\{0, 1, \ldots, n-1\}$. Given any node label $x_0 \cdots x_i \cdots x_{n-1}$, let function g_i, $1 \le i \le n-1$, be such that $g_i(x_0 \cdots x_i \cdots x_{n-1}) = x_i \cdots x_0 \cdots x_{n-1}$ (i.e., swap x_0 and x_i and keep the rest of the symbols unchanged). In S_n, for any node x, there is an edge joining x and node $g_i(x)$, and this edge is said to be *along* dimension i. It is known that S_n is node- and edge-symmetric and has a diameter of $D_n = \left\lfloor \frac{3(n-1)}{2} \right\rfloor$.

Throughout this paper, for any graph G, we denote by V_G and E_G the vertex set and edge set, respectively, of G. If node x has a label of $x_0 \cdots x_i \cdots x_{n-1}$, we may denote symbol x_i by $x[i]$. Given two nodes x and y, $dif(x, y)$ is the smallest $i > 0$ such that $x[i] \ne y[i]$; if $x = y$, then $dif(x, y) = \infty$. For instance, $dif(0123, 0132) = 2$. Given any node x, $\rho^i(x)$ is the node obtained from x by cyclically shifting the label of x to the right by i positions.

One-to-all broadcast refers to the problem of sending a message from one source node to all other nodes in the network, while *all-to-all broadcast* is $n!$ copies of the former problem with every node

• *Y.-C. Tseng is with the Department of Computer Science and Information Engineering, National Central University, Chung-Li, 32054, Taiwan. E-mail: yctseng@csie.ncu.edu.tw*
• *J.-P. Sheu is with the Department of Computer Science and Information Engineering, National Central University, Chung-Li, 32054, Taiwan. E-mail: sheujp@mbox.ee.ncu.edu.tw*

Manuscript received 27 July, 1995; revised 30 April, 1996.
For information on obtaining reprints of this article, please send e-mail to: transcom@computer.org, and reference IEEECS Log Number C96102.

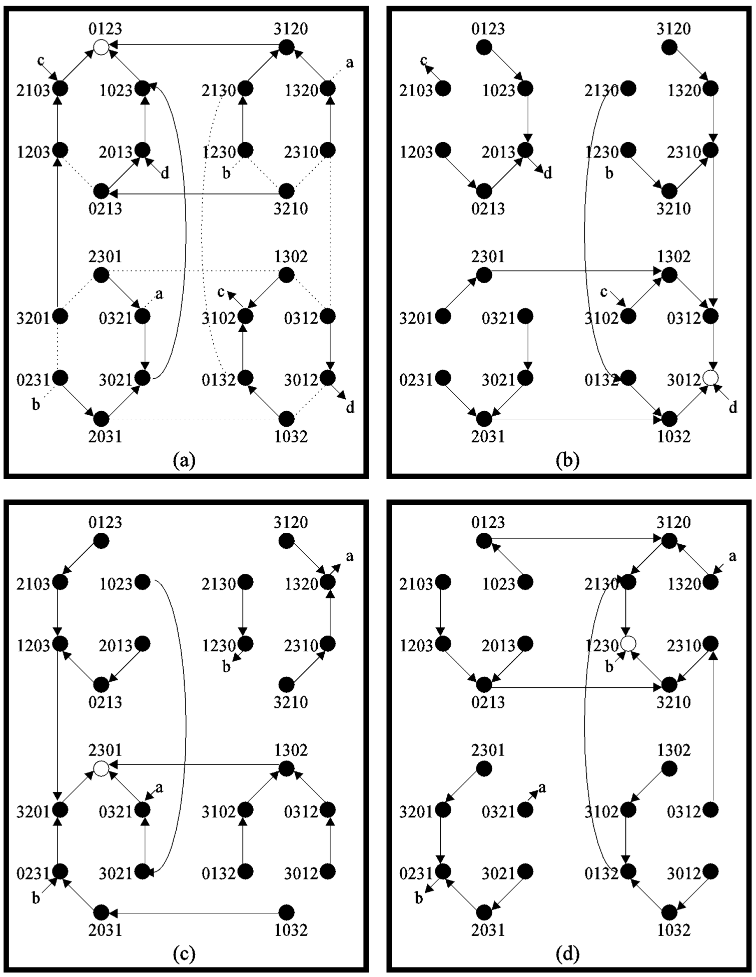

Fig. 1. Spanning trees in S_4: (a) $\mathcal{L}(0123)$, (b) $\mathcal{L}(3012)$, (c) $\mathcal{L}(2301)$, and (d) $\mathcal{L}(1230)$, where the circled nodes are roots. In (a), the dotted lines are the edges in S_4 that are not used by $\mathcal{L}(0123)$.

acting as the source node. In this paper, we assume the packet-switching (or store-and-forward) model, and thus the latency to transmit a packet of b bytes along a link is $T_s + bT_c$, where T_s is the time to initialize the communication link and T_c is the time to transmit a byte. Each (undirected) edge is regarded as two independent directed links (or channels) pointing in opposite directions. Two communication models will be considered. In the *one-port* model, a node can send and receive at most one packet at a time, while in the *all-port* model, a node can simultaneously send and receive packets along all $n - 1$ ports.

LEMMA 1. *A lower bound for one-to-all broadcasting in a store-and-forward S_n is* $\max\left\{D_nT_s, \frac{m}{n-1}T_c\right\}$ *under the all-port model, and* $\max\left\{D_nT_s, \lceil \log n! \rceil T_s, mT_c\right\}$ *under the one-port model, where m is the size of the broadcast message.*[1]

LEMMA 2. *A lower bound for all-to-all broadcasting in a store-and-forward S_n is* $\max\left\{D_nT_s, \frac{(n!-1)m}{n-1}T_c\right\}$ *under the all-port model, and* $\max\left\{D_nT_s, \lceil \log n! \rceil T_s, (n!-1)mT_c\right\}$ *under the one-port model, where m is the size of the broadcast message.*

1. All logarithm is based on 2 in this paper.

3 Construction of Multiple Spanning Trees

DEFINITION 1. *Given any node r in S_n, the directed graph $\mathcal{L}(r)$ is defined such that $V_{\mathcal{L}(r)} = V_{S_n}$ and $E_{\mathcal{L}(r)}$ contains the directed edge $\langle v, g_\alpha(v) \rangle$ for all vertices $v \in V_{\mathcal{L}(r)} - \{r\}$, where*

$$\alpha = \begin{cases} i & \text{if } v[0] = r[i] \wedge i \neq 0 \\ dif(v, r) & \text{if } v[0] = r[0] \end{cases}. \tag{1}$$

See Fig. 1 for examples. The following lemma can be proved easily using the routing rules in [7].

LEMMA 3. *The graph $\mathcal{L}(r)$ is a greedy spanning tree rooted from node r in S_n. From each node $v \neq r$, the edge $\langle v, g_\alpha(v) \rangle$ leads to a node closer to r.*

The next lemma shows the possibility of simultaneously embedding in S_n $n - 1$ copies of the above tree, which only incur an edge congestion of at most 2, where the *congestion* of a set of directed trees is defined to be the maximum number of times the links of these trees overlapping on same edges. (Note: Here two links of opposite directions are not considered overlapping.)

LEMMA 4. *Given any $r \in V_{S_n}$, the $n - 1$ directed trees $\mathcal{L}(\rho^1(r))$, $\mathcal{L}(\rho^2(r)), \ldots, \mathcal{L}(\rho^{n-1}(r))$ have an edge congestion of at most 2 in S_n.*

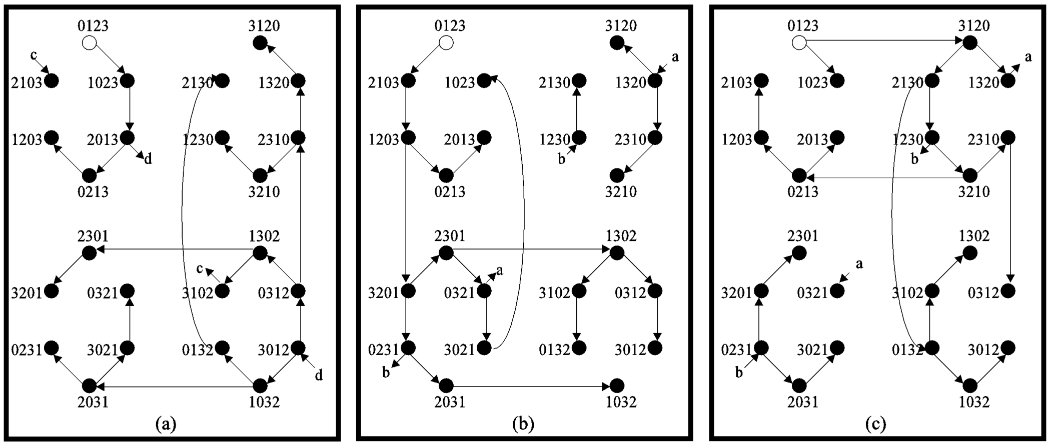

Fig. 2. The three congestion-2 spanning trees rooted at 0123 in S_4: (a) $\hat{\mathcal{L}}(3012)$, (b) $\hat{\mathcal{L}}(2301)$, and (c) $\hat{\mathcal{L}}(1230)$.

PROOF. By definition, $r[i] = \rho^j(r)[(i + j) \bmod n]$ for any i and j. (All arithmetics in the rest of the proof are based on "mod n," so we will omit saying so.) Consider the outgoing edges from any node x. Let $x[0] = r[i]$ for some i, $0 \le i \le n - 1$. We consider the value of i in two cases.

If $i = 0$, then for each $j = 1..n - 1$, $x[0] = \rho^j(r)[j]$. Since $j \ne 0$, in $\mathcal{L}(\rho^j(r))$ node x satisfies the first condition in (1). So there is an edge from x to $g_j(x)$ along dimension j in $\mathcal{L}(\rho^j(r))$. It follows that the $n - 1$ outgoing edges from x are all along different dimensions and there is no congestion among them.

If $i \ne 0$, then for each $j = 1..n - 1$, $x[0] = \rho^j(r)[i + j]$. Consider the $n - 2$ trees $\mathcal{L}(\rho^j(r))$ such that $j \ne 0$ or $n - i$. Since $i + j \ne 0$, in $\mathcal{L}(\rho^j(r))$ node x satisfies the first condition in (1) and there is an outgoing edge from x to $g_{i+j}(x)$ along dimension $i + j$. So there is no congestion for the $n - 2$ outgoing edges from x in the above $n - 2$ trees. It remains one more tree, $\mathcal{L}(\rho^{n-i}(r))$, yet to be considered, and, obviously, the congestion is at most 2. □

An example of the above lemma is shown in Figs. 1b, 1c, and 1d, where the three trees $\mathcal{L}(\rho^1(r))$, $\mathcal{L}(\rho^2(r))$, and $\mathcal{L}(\rho^3(r))$ together have only congestion of 2 in S_4.

4 ONE-TO-ALL BROADCAST

In this section, we consider the problem of a source node r broadcasting a message M of size m to the rest nodes in S_n. As M may be sliced into submessages, we assume for ease of presentation that m is infinitely divisible.

DEFINITION 2. *Given any node r, let p_i be the (unique) path in the tree $\mathcal{L}(\rho^i(r))$ leading from r to $\rho^i(r)$. Define $\hat{\mathcal{L}}(\rho^i(r))$, $i = 1..n - 1$, to be the directed graph obtained from $\mathcal{L}(\rho^i(r))$ by reversing the direction of all edges of $\mathcal{L}(\rho^i(r))$ except those edges that are along the path p_i.*

Now each $\hat{\mathcal{L}}(\rho^i(r))$ is a tree, which spans, following the directions of the edges, from r to the rest of the network. Fig. 2 illustrates the trees in Figs. 1b, 1c, and 1d after the above transformation.

LEMMA 5. *The height of $\hat{\mathcal{L}}(\rho^i(r))$ is $D_n + n + gcd(n, i) - 2$.*

PROOF. The height is D_n plus the length of p_i. Observe that r is a permutation of $\rho^i(r)$. A permutation can be viewed as a set of cycles [5], [7], such that each cycle is a sequence of dimensions, say, $(d_0, d_1, d_2, \ldots)$ in which the desired position of the symbol in dimension d_i is dimension d_{i+1}. (For instance, consider r = 01234567, which is a permutation of $\rho^6(r)$ = 23456701. Node r can be viewed as two cycles $C_0 = (0, 6, 4, 2)$ and $C_1 = (1, 7, 5, 3)$. See Fig. 3.) Also observe that $\rho^i(r)$ is obtained by cyclically shifting the label of r. It is a simple fact from number theory that there are $gcd(n,i)$ cycles, namely

$$C_k = (k, (k + i) \bmod n, (k + 2i) \bmod n, (k + 3i) \bmod n, \ldots), \quad (2)$$

where $k = 0..\gcd(n, i) - 1$. Because $\mathcal{L}(\rho^i(r))$ is greedy, the length of p_i is the minimum distance from r to $\rho^i(r)$. By [1], this distance equals the number of misplaced symbols plus the number of cycles minus 2. As the number of misplaced symbols is always n, the height then follows. □

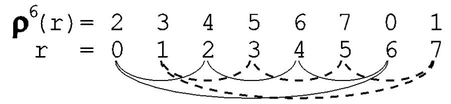

Fig. 3. The node r represented as a permutation of $\rho^6(r)$ using two cycles.

LEMMA 6. *Given any $r \in V_{S_n}$, the n – 1 directed trees $\hat{\mathcal{L}}(\rho^1(r))$, $\hat{\mathcal{L}}(\rho^2(r))$, …, $\hat{\mathcal{L}}(\rho^{n-1}(r))$ have a congestion of at most 2 in S_n.*

PROOF. Since the edges in p_i are not reversed in the translation from $\mathcal{L}(\rho^i(r))$ to $\hat{\mathcal{L}}(\rho^i(r))$, it is easy to see that excluding those edges in p_is, the trees $\hat{\mathcal{L}}(\rho^i(r))$, $i = 1..n - 1$, still have a congestion of at most 2 in S_n. So it remains to calculate the congestion for the edges in p_is. Observe that any two p_is are edge disjoint (this can be proved by showing that each node $v \ne r$ in p_i has $v[i] = r[0]$). So the proof will be complete if we can show that for any edge $e = \langle x, y \rangle \in p_i$, there exists at most one edge $e' = \langle y, x \rangle$ in the trees $\mathcal{L}(\rho^1(r)), \mathcal{L}(\rho^2(r)), \ldots, \mathcal{L}(\rho^{n-1}(r))$. Note that e' (if any) will be reversed by Definition 2 and cause congestion with e.

In order to have an edge $e' = \langle y, x \rangle$, there must exist one tree such that y satisfies one of the conditions in (1). Furthermore, we could obtain two such edges only if there are two trees in one of which y satisfies the first condition in (1), and in another of which y satisfies the second condition. This is impossible as shown below.

Now suppose $x = g_d(y)$. We view r as a permutation of $\rho^i(r)$ by representing r as a set of cycles C_k, $k = 0..\gcd(n, i) - 1$, where C_k is as defined in (2). From (1), observe that path p_i will pass the sequence of dimensions:

$$seq = trail(C_0), C_1, first(C_1), C_2, first(C_2), C_3, first(C_3), \ldots, C_{\gcd(n,i)-1}, first(C_{\gcd(n,i)-1}),$$

where the function *first*(s) returns the first element of a sequence s, and function *trail*(s) is sequence s with *first*(s) removed. In the following, we consider the location of d in *seq* in two cases.

Case 1: d located in *first*(C_k) for some k. We first observe two facts:

F1: After finishing routing the dimensions up to *trail*(C_0), the end node has $\rho^i(r)[0]$ as its first symbol.

F2: After finishing the routing up to *first*(C_k) for any k, the end node still has $\rho^i(r)[0]$ as its first symbol. (This can be proved by simple induction.)

So $y[0] = \rho^i(r)[0]$. This implies that the second condition of (1) cannot be satisfied for any tree. Otherwise, suppose there is a tree $\mathcal{L}(r')$ having the edge e' by satisfying this condition. Then $r'[0] = y[0] = \rho^i(r)[0]$, implying that $r' = \rho^i(r)$ (because r' is cyclically shifted from r). So e' is an edge in $\mathcal{L}(\rho^i(r))$ and a cycle consisting of e and e' is formed in $\mathcal{L}(\rho^i(r))$.

Case 2: d not located in any *first*(C_k). Observe that this d is the first such value appearing in *seq*. Therefore, $x[d] = r[d] = y[0]$. This implies that the first condition of (1) cannot be satisfied by any tree. Otherwise, suppose tree $\mathcal{L}(r')$ having the edge e' by satisfying this condition. This means $r'[d] = y[0] = r[d]$, implying $r' = r$, which is impossible. So there is at most one edge of type e'. □

4.1 All-Port Model

In the proposed algorithm, time will be slotted by fixed length and all nodes in the network are assumed to perform broadcast synchronously. In each time slot each node will transmit a packet of a fixed size $\frac{2m}{p(n-1)}$, where m is the size of the message M and p is an integer to be determined later. So each time slot is of length $T_s + \frac{2m}{p(n-1)} T_c$.

```
Algorithm 1: /* One-to-all-broadcast, all-port */
1) Slice the message M evenly into $p(n-1)$ parts, each called a "message segment" and of size $\frac{m}{p(n-1)}$.
2) In each time slot, node r issues $n-1$ message segments to the network, each along one of the trees $\hat{\mathcal{L}}(\rho^i(r))$, $i = 1..n-1$.
   A message segment is then propagated along the tree it is issued. In each time slot, each node helps propagating all message segments it received in the previous time slot to the subsequent nodes in the corresponding trees.
```

Note that a packet can hold *two* message segments generated by step 1. However, in each time slot, node r only issues *one* message segment along each tree. This is because the maximum edge congestion of the trees $\hat{\mathcal{L}}(\rho^i(r))$, $i = 1..n-1$, is two. So in each time slot, every node will be able to propagate all message segments it received in the previous time slot without any delay.

Next, we analyze the communication latency of the algorithm. The computational time (such as making routing decision or packing/unpacking packets) will be ignored in the analysis. Let h be the maximal height of $\hat{\mathcal{L}}(\rho^i(r))$, $i = 1..n-1$. Note that by Lemma 5, $D_n + n - 1 \le h \le 2D_n$. The broadcast algorithm will take

$$T = h\left(T_s + \frac{2m}{p(n-1)}T_c\right) + (p-1)\left(T_s + \frac{2m}{p(n-1)}T_c\right) \tag{3}$$

time to finish, where the former term is the time for the first packet to arrive at the bottom of the tallest tree and the latter term is due to the pipelined effect. To minimize (3), let the derivative of T with respect to p equal to 0,

$$\frac{\partial T}{\partial p} = T_s - \frac{2m(h-1)T_c}{(n-1)p^2} = 0. \tag{4}$$

So we obtain

$$p = \sqrt{\frac{2m(h-1)T_c}{(n-1)T_s}} = O\left(\sqrt{\frac{mT_c}{T_s}}\right). \tag{5}$$

THEOREM 1. *Under the all-port model, one-to-all broadcast can be performed in S_n within time*

$$\left(h + \sqrt{\frac{2m(h-1)T_c}{(n-1)T_s}} - 1\right)\left(T_s + \sqrt{\frac{2mT_sT_c}{(n-1)(h-1)}}\right) = O\left(nT_s + \sqrt{mT_sT_c} + \frac{mT_c}{n}\right).$$

By Lemma 1, a lower bound for this problem is $\Omega(nT_s + (m/n)T_c)$. When $n^2T_s \gg mT_c$, the above order reduces to $O(nT_s)$. When $n^2T_s \ll mT_c$, the above order reduces to $O((m/n)T_c)$. So our algorithm is asymptotically optimal.

4.2 One-Port Model

A node with one-port communication capability can simulate the communication activity of an all-port node in one time slot using $n-1$ time slots. The simulation can be done as follows: In the first time slot, the one-port node simulates the all-port node's activity along dimension 1; in the second time slot, the one-port node simulates the all-port node's activity along dimension 2; etc. Clearly, the communication follows the one-port model. By simulating **Algorithm 1** at every one-port node in S_n, the following theorem is readily seen.

THEOREM 2. *Under the one-port model, one-to-all broadcast can be performed in S_n within time*

$$(n-1)\left(h + \sqrt{\frac{2m(h-1)T_c}{(n-1)T_s}} - 1\right)\left(T_{s+}\sqrt{\frac{2mT_sT_c}{(n-1)(h-1)}}\right) = O\left(n^2T_s + n\sqrt{mT_sT_c} + mT_c\right).$$

By Lemma 1, a lower bound for this problem is $\Omega((n\log n)T_s + mT_c)$. When $n^2T_s \gg mT_c$, the above order reduces to $O(n^2T_s)$, which is an order of $O(n/\log n)$ higher than optimum. When $n^2T_s \ll mT_c$, the above order reduces to $O(mT_c)$, so our algorithm is asymptotically optimal when the broadcast message is sufficiently large. Algorithms which use optimal start-up time do exist [2], [12], [17], but the transmission time is not optimal. Hence, the algorithm presented here provides an alternative when a large message needs to be broadcast.

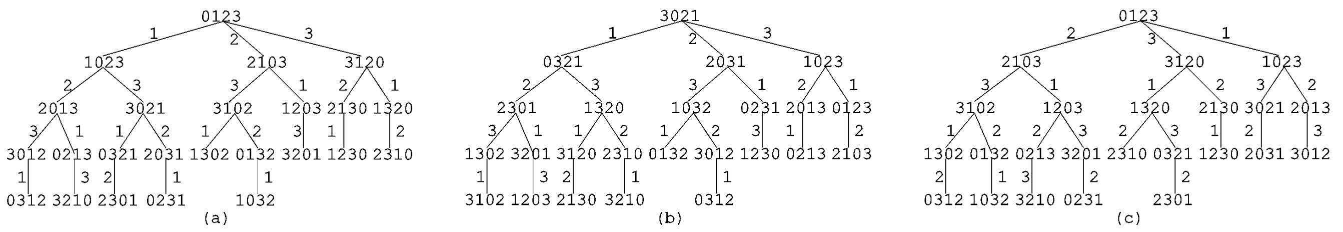

Fig. 4. (a) The spanning tree $\mathcal{L}(0123)$ in S_4, (b) the tree $LC(3021, \mathcal{L}(0123))$ obtained from label change, and (c) the tree $DC(1, \mathcal{L}(0123))$ obtained from dimension change.

5 All-to-All Broadcast

To perform all-to-all broadcast, each node will use $n-1$ spanning trees. These trees are obtained from two special operations called *label change (LC)* and *dimension change (DC)* as defined below.

Definition 3. *Let $x = x_0x_1\ldots x_{n-1}$ be any node in S_n. Given any spanning tree T of S_n, define LC(x,T) to be the tree obtained from T by translating each node $y_0y_1\ldots y_{n-1}$ in T into node $x = x_{y_0}x_{y_1}\ldots x_{y_{n-1}}$*

Definition 4. *Given any spanning tree T rooted at r and any integer i, we define DC(i,T) to be the tree obtained from T by performing the following two steps:*

1) *Translate each link, say, along dimension* d, *into one along dimension* $d' = [(d+i-1) \bmod (n-1)] + 1$ *(i.e., cyclically increase* d *by* i *to obtain* d').
2) *Let the root of* DC(i,T) *be labeled* r *and modify the labels of all other nodes according to the dimensions defined in step (1).*

The LC operation will modify the labels of nodes in T to generate a new tree. The DC operation will modify the dimensions of links in T to generate a new tree (note that in doing so, the node labels may also be modified). For instance, Fig. 4a shows the undirected version of tree $T = \mathcal{L}(0123)$. Fig. 4b shows *LC(3021, T)*, where symbols 0, 1, 2, 3 in T are changed to 3, 0, 2, 1, respectively. Fig. 4c shows $DC(1, T)$, where dimensions 1, 2, 3 are modified to 2, 3, 1, respectively. Since star graphs are node- and edge-symmetric, it is easy to see that $LC(x, T)$ and $DC(i, T)$ are both spanning trees.

Next, we need to identify a link's location in a tree. Intuitively, the distance of a link from the root node indicates when the link needs to help propagating a message. The following definition is intended for this purpose. Note that in a tree we number links as level 1, level 2, …, etc, starting from the root.

Definition 5. *Let T be any spanning tree and t be any integer. Given any link e, define*

$$f(e,t,T) = \begin{cases} 1 & \text{if } e \text{ is located in } T \text{ at level } t \\ 0 & \text{otherwise} \end{cases}. \qquad (6)$$

Lemma 7. *Let T be any spanning tree and t be any integer. Suppose that integer i is ranging from* 0 *to* $n-2$ *and labels x is ranging from the n! possible node labels in S_n. Then for every link $e \in E_{S_n}$, the value of the following expression is always the same*:

$$\sum_{\forall i, \forall x} f(e,t,LC(x,DC(i,T))). \qquad (7)$$

Proof. There are $(n-1)n!$ trees in (7). First, consider the $n!$ trees $LC(x,T)$ for all permutations of x. Let e be any link along dimension d such that $f(e, t, T) = 1$. Link e will be translated into $n!$ distinct links in these $n!$ trees. As all these $n!$ links are also along dimension d, they actually contain all the links in S_n along dimension d. We thus conclude that every link along the same dimension will have the same value of

$$\sum_{\forall x} f(e,t,LC(x,T)).$$

Second, for any fixed i, consider the $n!$ trees $LC(x, DC(i,T))$. It is nature to extend the above argument. So links along the same dimension will have the same value

$$\sum_{\forall i \forall x} f(e,t,LC(x,DC(i,T))).$$

It remains to prove the equivalence for links along different dimensions. This follows from the observation that the links located at level t in the $n-1$ trees, $DC(i,T), i = 0..n-2$, are evenly distributed into every dimension. Hence, the lemma. ☐

5.1 All-Port Model

The main idea is as follows. Let T be any spanning tree rooted at the identity node $I = 01\ldots(n-1)$. Each node x will use $n-1$ spanning trees $LC(x, DC(i, T))$, $i = 0..n-2$, to broadcast its message. These trees are all rooted at x. At time step t, all communication links located at the tth level of these trees will help propagating x's broadcast message. Concurrently and synchronously, all nodes will follow such scheduling. If so, by Lemma 7 all communication links in S_n will be equally loaded.

The algorithm is spelled out below for node x, where T is assumed to be $\mathcal{L}(I)$. All nodes perform this algorithm concurrently.

```
Algorithm 2: /* All-to-all-broadcast, all-port */
1) Slice the message evenly into $(n-1)$ message segments. We
   associate each message segment to one of the spanning trees
   $LC(x, DC(i, T))$, $i = 0..n-2$, along which the message seg-
   ment will be propagated.
2) for $t = 1$ to $D_n$ do
   At time step $t$, every node in $S_n$ helps propagating message
   segments along their associated spanning trees. Along each
   link $e$, a node needs to transmit a packet containing
   $\sum_{\forall i, \forall x} f(e,t,LC(x,DC(i,T)))$
   message segments.
```

Now we analyze the time complexity of the algorithm. We assume that in each iteration of step 2, all message segments can be combined into one packet and sent at one time. So the start-up overhead is D_nT_s. To calculate the transmission time, let m be the size of broadcast messages. To propagate a message segment (of size $\frac{m}{n-1}$) along a spanning tree (of $n!-1$ links), network bandwidth of $\frac{m(n!-1)}{n-1}T_c$ is required. There are totally $(n-1)n!$ message segments being transmitted. So the total network bandwidth required

TABLE 1
COMPARISON OF ONE-TO-ALL BROADCAST ALGORITHMS ON START-UP COST, TRANSMISSION COST, AND OVERALL TIME COMPLEXITY

Model	Algorithm	Start-up comp.	Trans. comp.	Overall complexity
all-port	Sheu [16]	$O(nT_s)$	$O(mT_c)$	$O(nT_s + \sqrt{nmT_sT_c} + mT_c)$
	Day [7]	$O(nT_s)$	$O(mT_c)$	$O(nT_s + \sqrt{nmT_sT_c} + mT_c)$
	Ours	$O(nT_s)$	$O(mT_c / n)$	$O(nT_s + \sqrt{mT_sT_c} + mT_c / n)$
one-port	Misic [13]	$O(n^2T_s)$	$O(n^2mT_c)$	$O(n^2T_s + n^2mT_c)$
	Akl [2]	$O(n\log nT_s)$	$O(nm\log nT_c)$	$O(n\log n(T_s + mT_c))$
	Mendia [12]	$O(n\log nT_s)$	$O(nm\log nT_c)$	$O(n\log n(T_s + mT_c))$
	Sheu [17]	$O(n\log nT_s)$	$O(m\log nT_c)$	$O(\log n(nT_s + \sqrt{nmT_sT_c} + mT_c))$
	Qiu [14]	$O(n^2T_s)$	$O((n\log n + m)T_c)$	$O(n^2T_s + (n\log n + m)T_c)$
	Ours	$O(n^2T_s)$	$O(mT_c)$	$O(n^2T_s + n\sqrt{mT_sT_c} + mT_c)$

TABLE 2
COMPARISON OF ALL-TO-ALL BROADCAST ALGORITHMS ON START-UP COMPLEXITY, TRANSMISSION COMPLEXITY, AND OVERALL LATENCY

Model	Algorithm	Start-up comp.	Trans. comp.	Overall time
all-port	Frago. HAM [8]	$O((n-1)!T_s)$	$O(\frac{(n!-1)m}{n-1}T_c)$	$(n-1)!T_s + \frac{(n!-1)m}{n-1}T_c$
	Frago. NL [8]	$O(nT_s)$	$O(\frac{(n!-1)m}{n-1}T_c)$	$D_nT_s + \frac{(n!-1)m}{n-1}T_c$
	Ours	$O(nT_s)$	$O(\frac{(n!-1)m}{n-1}T_c)$	$D_nT_s + \frac{(n!-1)m}{n-1}T_c$
one-port	Misic [13]	$O(n!T_s)$	$O(n!mT_c)$	$(n!-1)(T_s + mT_c)$
	Ours	$O(n^2T_s)$	$O(n!mT_c)$	$(n-1)D_nT_s + mT_c(n!-1)$

is $m(n!-1)n!T_c$. Since the network is equally loaded at every time step, the bandwidth will be evenly distributed to all $(n-1)n!$ links in the network. So the transmission time is

$$\frac{m(n!-1)n!T_c}{(n-1)n!} = \frac{m(n!-1)}{(n-1)}T_c. \qquad (8)$$

THEOREM 3. *Under the all-port model, all-to-all broadcast can be performed in S_n with time*

$$D_nT_s + \frac{m(n!-1)}{(n-1)}T_c.$$

By Lemma 2, the above time is optimal in both start-up time and transmission time. Also, note that the assumption $T = \mathcal{L}(I)$ is not a necessary condition. Any spanning tree will work for our algorithm. As long as T has an optimal height of D_n (which does not necessarily imply that T is a greedy tree), the communication latency remains optimal.

5.2 One-Port Model

A node with one-port communication capability can simulate an all-port node by a delay factor of $n-1$. By simulating **Algorithm 2**, all-to-all broadcast can be performed in S_n within $(n-1)D_nT_s + m(n!-1)T_c$ under the one-port model.

6 COMPARISON WITH RELATED WORKS

6.1 One-to-All Broadcast

Under the all-port model, it is common to use one spanning tree to solve this problem. Sheu et al. [16] suggests a spanning tree of height $2n-3$. A better tree, which is greedy and has height of D_n, is proposed by Day and Tripathi [7]. Without using the pipelining technique, broadcasting along these trees needs $h'(T_s + mT_c) = O(nT_s + mnT_c)$ time, where h' is the height of the tree. This has used the optimal start-up time, but the transmission time is an order of $O(n^2)$ higher than optimum. By applying a pipelining technique as in Section 4.1 (by slicing the broadcast message), the time complexity can be reduced to $O(nT_s + \sqrt{nmT_sT_c} + mT_c)$(we leave the detailed derivation to the reader, as the approach is similar), which is asymptotically optimal in start-up time, but is still an order of $O(n)$ higher than the optimal transmission time. In this paper, by using the pipelining technique along multiple spanning trees, both start-up time and transmission time are minimized. See Table 1 for a summary.

Under the one-port model, many broadcast solutions have been proposed [2], [12], [13], [14], [17]. The algorithm by Misic and Jovanovic [13] requires $n(n-1)/2$ phases to complete, where in each phase the whole message is transmitted. So the latency is $n(n-1)(T_s + mT_c)/2$. The algorithms by Akl et al. [2] and Mendia and Sarkar [12] further reduce the number of phases required to $\sum_{i=2}^{n}\lceil\log i\rceil = O(n\log n)$, resulting in a time complexity of $O(n\log n(T_s + mT_c))$. The start-up time is asymptotically optimal, but the transmission is an order of $O(n \log n)$ higher then optimum.

Sheu et al. [17] observe that in the above algorithms [2], [12], [13], a node may receive the broadcast message more than once. This would make applying the pipelining technique difficult (if not impossible), as the flow of the pipeline may be impeded. Sheu et al. [17] shows how to broadcast a packet in $\sum_{i=2}^{n}\lceil\log i\rceil = O(n\log n)$ phases. Furthermore, pipelining is possible because the next packet can be issued in $\lceil\log n\rceil$ phases after the previous one was issued. Suppose we slice the message into p segments (or packets) each of size m/p. Then the time it takes is $T = ((p-1)\log n + n\log n)(T_s + \frac{m}{p}T_c)$. T is minimized when $p = O(\sqrt{nmT_c / T_s})$(following similar formulation as in Section 4.2). So the time complexity becomes $O(\log n(nT_s + \sqrt{nmT_sT_c} + mT_c))$. The start-up time is asymptotically optimal, but the transmission time is still an order of $O(\log n)$ higher than optimum. Consequently, under the one-port model, our result improves over [17] in transmission time by an order of $O(\log n)$, but has a start-up cost of $O(n/\log n)$ higher than optimum. Our algorithm provides an alternative to [17] when a large message (i.e., $n^2T_s \ll mT_c$) needs to be broadcast.

Recently, Qiu [14] proposed a new algorithm that uses $O((n \log n + m)T_c)$ transmission time (the work did not try to optimize the start-up cost). The algorithm is based on a divide-and-conquer approach. First, S_n is partitioned into n substars S_{n-1} and the message is sliced into n segments. Each segment is then sent to one of the S_{n-1}, in which the broadcast will proceed recursively. Finally, the message segments are combined in each node. This uses $O(mT_c)$ transmission time. Note that the above statement has assumed that a message segment can always be sliced into smaller ones. This may not be possible when m is too small. Thus, Qiu [14] suggests to apply the algorithm of [2], [12] in the recursion when the size of message segments reaches some constant. This incurs $O(n \log nT_c)$ transmission time. A serious problem in this approach is that to achieve the previous order $O(mT_c)$, the broadcast message needs to be sliced into n! segments (e.g., when n = 8, $n! \approx$ 40,000). This number of message segments is much larger than ours of $O\left(\sqrt{\frac{mT_c}{T_s}}\right)$(see (5)). Also, the computational overhead to pack and unpack message segments will be significantly higher than ours. We summarize the above discussion in Table 1.

6.2 All-to-All Broadcast

Under the all–port model, Fragopoulou and Akl [8] propose to use n! isomorphic trees. Each node uses one tree. Altogether these n! trees incur equal communication load on all links. There are two ways to construct such trees. The first way is to partition the S_n into n S_{n-1}s and find a Hamiltonian path in each S_{n-1}. So the height of the trees is $\approx (n - 1)!$. The second way is based on grouping nodes in S_n into a number of *necklaces*. The height is reduced to D_n. Both ways achieve the optimal transmission cost of $\frac{(n!-1)m}{n-1}T_c$. But the first tree will have a high start-up cost of $(n-1)!T_s$, while the second will only take D_nT_s start-up time. So the second tree can give the same performance as ours. In our algorithm, each node uses $n - 1$ trees and there are totally $(n - 1)n!$ trees. Our result is more general—as commented in Section 5, any tree of height D_n can be used to achieve the same performance.

Under the one-port model, Misic and Jovanovic [13] develops a scheduling that can perform all-to-all broadcast in optimal transmission time of $(n! - 1)mT_m$, but high start-up time of $(n!-1)T_s$. Fragopoulou and Akl [8] suggest using a Hamiltonian cycle in S_n and every node simply propagates its message along the cycle. This has the same problem of high start-up time as in [13]. It would be more efficient to simulate an all-port algorithm, such as the necklace scheme of [8] or our all-port algorithm, as suggested in this paper.

The above discussions are summarized in Table 2. The two trees used in [8] under the all-port model are denoted as HAM (Hamiltonian path) and NL (necklace).

7 Conclusions

We have shown how to solve various versions of broadcast problems in a star graph using multiple spanning trees to simultaneously optimize both start-up and transmission costs. For one-to-all broadcast under the all-port model, our algorithm is optimal in both start-up time and transmission time, while existing results only achieve optimal start-up time. For one-to-all broadcast under the one-port model, our algorithm is optimal in transmission time, while existing results either only achieve optimal start-up time or achieve the same performance as ours but having a much higher computational overhead. For all-to-all broadcast under the all-port model, our algorithm is optimal in both start-up time and transmission time. For all-to-all broadcast under the one-port model, our algorithm is optimal in transmission time.

Acknowledgments

Dr. Yu-Chee Tseng's research is supported by the National Science Council of the Republic of China under Grant # NSC85-2213-E-216-021. Dr. Jang-Ping Sheu's research is supported by the National Science Council of the Republic of China under Grant # NSC 85-2213-E-008-032.

References

[1] S.B. Akers, D. Harel, and B. Krishnameurthy, "The Star Graph: An Attractive Alternative to the *n*-Cube," *Proc. Int'l Conf. Parallel Processing*, pp. 393-400, 1987.
[2] 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.
[3] N. Bagherzadeh, N. Nassif, and S. Latifi, "A Routing and Broadcasting Scheme on Faulty Star Graphs," *IEEE Trans. Computers*, vol. 42, no. 11, pp. 1,398-1,403, Nov. 1993.
[4] J.-C. Bermond, P. Michallon, and D. Trystram, "Broadcasting in Wraparound Meshes with Parallel Monodirectional Links," *Parallel Computing*, vol. 18, pp. 639-648, 1992.
[5] J.A. Bondy and U.S.R. Murthy, *Graph Theory with Applications*. Amsterdam: North Holland, 1979.
[6] T.-S. Chen, Y.-C. Tseng, and J.-P. Sheu, "Balanced Spanning Trees in Complete and Incomplete Star Graphs," *IEEE Trans. Parallel and Distributed Systems*, vol. 7, no. 7, pp. 717-723, July 1996.
[7] K. Day and A. Tripathi, "A Comparative Study of Topological Properties of Hypercubes and Star Graphs," *IEEE Trans. Parallel and Distributed Systems*, vol. 5, no. 1, pp. 31-38, Jan. 1994.
[8] P. Fragopoulou and S.G. Akl, "Optimal Communication Algorithms on Star Graphs Using Spanning Tree Constructions," *J. Parallel and Distributed Computing*, vol. 24, pp. 55-71, 1995.
[9] S.L. Johnsson and C.T. Ho, "Optimal Broadcasting and Personalized Communication in Hypercubes," *IEEE Trans. Computers*, vol. 38, no. 9, pp. 1,249-1,268, Sept. 1989.
[10] J.-S. Jwo, S. Lakshmivarahan, and S.K. Khall, "Embeddings of Cycles and Grids in Star Graphs," *Proc. Symp. on Parallel and Distributed Processing*, pp. 540-547, 1990.
[11] S. Latifi and N. Bagherzadeh, "Incomplete Star: An Incrementally Scalable Network Based on the Star Graph," *IEEE Trans. Parallel and Distributed Systems*, vol. 5, no. 1, pp. 97-102, Jan. 1994.
[12] V.E. Mendia and D. Sarkar, "Optimal Broadcasting on the Star Graph," *IEEE Trans. Parallel and Distributed Systems*, vol. 3, no. 4, pp. 389-396, July 1992.
[13] J. Misic and Z. Jovanovic, "Communication Aspects of the Star Graph Interconnection Network," *IEEE Trans. Parallel and Distributed Systems*, vol. 5, no. 7, pp. 678-687, July 1994.
[14] K. Qiu, "Broadcasting on the Star and Pancake Interconnection Networks," *Proc. Int'l Parallel Processing Symp.*, pp. 660-665, 1995.
[15] K. Qiu, S.G. Akl, and H. Meijer, "On Some Properties and Algorithms for the Star and Pancake Interconnection Networks," *J. Parallel and Distributed Computing*, vol. 22, pp. 16-25, 1994.
[16] J.-P. Sheu, W.-H. Liaw, and T.-S. Chen, "A Broadcasting Algorithm in Star Graph Interconnection Networks," *Information Processing Letters*, vol. 48, pp. 237-241, 1993.
[17] J.-P. Sheu, C.-T. Wu, and T.-S. Chen, "An Optimal Broadcasting Algorithm without Message Redundancy in Star Graphs," *IEEE Trans. Parallel and Distributed Systems*, vol. 6, no. 6, pp. 653-658, June 1995.
[18] Y.-C. Tseng, S.-H. Chang, and J.-P. Sheu, "Fault-Tolerant Ring Embedding in Star Graphs," *Proc. Int'l Parallel Processing Symp.*, pp. 660-665, 1996.
[19] Y.-C. Tseng, T.-H. Lai, and L.-F. Wu, "Matrix Representation of Graph Embedding in a Hypercube," *J. Parallel and Distributed Computing*, vol. 23, pp. 215-223, 1994.