Multicast Algorithms on the Star Graph Network

Chang-Biau Yang and Chiu-Chu Liu
Department of Applied Mathematics
National Sun Yat-sen University
Kaohsiung, Taiwan 80424
cbyang@math.nsysu.edu.tw

Abstract

The multicast communication problem has been formulated as four different versions of graph theoretical problems: Steiner tree, multicast tree, multicast path and multicast cycle. Our efforts in this paper are to reduce the communication traffic of multicast in the star graph interconnection network. We propose two heuristic algorithms for the first two of the four problem models. Our Steiner tree algorithm is centralized, and our multicast tree algorithm is hybrid, which is partially centralized and partially distributed. By experiment results, the performance of our algorithms is superior to that of two previous algorithms.

Key words. star graph, multicast, interconnection network.

1 Introduction

In a multiprocessor system, the three types of interprocessor communications are *one-to-one (unicast)*, *one-to-many (multicast)* and *one-to-all (broadcast)*. Since unicast and broadcast communication have been studied intensively, recently researchers focus their attention on the multicast communication. Multicast is highly demanded in many applications, such as parallel game-tree search algorithms [6], parallel graph algorithms [9], speech analysis and image processing problems [4]. Such kind of algorithms usually need a source node to send messages to k destination nodes repeatedly. The main problem here is to determine which paths should be used to deliver the message to the k destination nodes. Our work in this paper is to study how to select the transmission paths with minimum or near minimum length in the star graph interconnection network.

An *n-dimensional star graph* [1,2,8,12,14], denoted as n-star or S_n, is represented by $S_n = (V_n, E_n)$, where V_n consists of $n!$ nodes in which each node is identified by one of the permutations of $\{1, 2, \cdots, n\}$. For any $u, v \in V_n$, $(u, v) \in E_n$ if and only if there exists some i, $2 \leq i \leq n$, such that u and v are identified as $(p_1\ p_2\ \cdots\ p_{i-1}\ p_i\ p_{i+1}\ \cdots\ p_n)$ and $(p_i\ p_2\ \cdots\ p_{i-1}\ p_1\ p_{i+1}\ \cdots\ p_n)$ respectively. Here we say that u and v are connected by link g_i, and we write $g_i(u) = v$. For $u, v \in S_n$, we have that $g_i(u) = v$ if and only if $g_i(v) = u$. For example, S_4 is illustrated in Figure 1. The *distance* between nodes u and v on S_n, denoted as $H(u, v)$, is defined as the length of the shortest path between u and v, where each edge is counted as length 1.

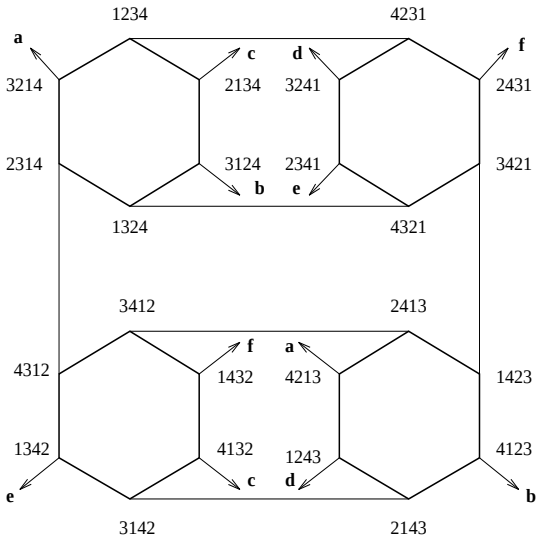


Figure 1: A 4-dimensional star graph, S_4.

The star graph is comparable to the hypercube in many aspects. For example, both of them are edge symmetric, node symmetric, strongly hierarchical, bipartite and optimally fault tolerant. However, star graphs offer a lower degree and a smaller diameter than hypercubes. So the star graph structure has been considered as an attractive alternative to the hyper-

cube structure [1,2].

The message transmission time is highly dependent on the underlying switching technology [11]. Early direct networks used the *store-and-forward* switching. In this approach, the *network latency* is proportional to the distance, which is the number of hops (links) in the network, between the source and the destination nodes. Recently, more switching technologies have been proposed, such as *circuit switching*, *virtual cut-through* and *wormhole routing* [11]. In these cases, the network latency or message transmission time is almost independent of the number of hop(s) between the source and destination nodes. Depending on the underlying switching technology which a machine adopts, Lin and Ni formulated the multicast communication problem in multicomputers as four different graph problems [11]: the Steiner tree (ST) problem, the multicast tree (MT) problem, the multicast path (MP) problem and the multicast cycle (MC) problem. We give the formal definitions of the ST problem and the MT problem as follows.

Definition 1 *Given a graph $G = (V, E)$ and a multicast set $M \subseteq V$, a Steiner Tree is a subtree $G_S = (V_S, E_S)$ of G, such that $M \subseteq V_S$. A multicast tree is a subtree $G_T = (V_T, E_T)$ of G such that $M \subseteq V_T$ and for each $u \in M$, the path length from u_0 to u on T is equal to $H(u_0, u)$. The Steiner tree and multicast tree problems are to find the Steiner tree and the multicast tree with minimum total length respectively.*

The optimization of the above two problems on a general graph have been proved to be NP-complete [7,10,11]. In this paper, we shall propose two heuristic algorithms to solve the ST and MT problems, and compare each of them with the previous results [11,14] by some simulation experiments. Our algorithms usually need less amount of communication traffic than the previous algorithms do. When implementing a multicast routing scheme, another issue is whether *centralized* routing or *distributed* routing should be adopted [11,13,15]. In this paper, our ST algorithm is centralized, and our MT algorithm is *hybrid* (The preprocessing is centralized and other processing is distributed.).

The rest of this paper is organized as follows. In Section 2, we shall first present some properties of star graphs. Then, we shall briefly review the heuristic ST algorithm on the hypercube, proposed by Lin and Ni [11], and the heuristic MT algorithm on the star graph, proposed by Tsay [14]. In Sections 3 and 4, we shall propose our heuristic algorithms for the ST problem and the MT problem on the star graph respectively. In Section 5, we shall give the simulation experiment results and compare the performances of our multicast algorithms with the previous results. Finally, some concluding remarks will be given in Section 6.

2 Previous Work

2.1 Properties of the Star Graph

We shall review some properties of the star graph as follows.

Property 1 *[1]An n-star graph is a regular graph of degree $n-1$.*

Property 2 *[1] The diameter of an n-star graph is $\lfloor \frac{3(n-1)}{2} \rfloor$.*

Property 3 *[1]The number of $(n-p)$-stars in an S_n, counting all possible sub-stars, is $\binom{n-1}{p} \frac{n!}{(n-p)!}$.*

Property 4 *[1, 2] The star graph is both edge symmetric and node symmetric.*

Property 5 *[1, 2, 12] There are optimal routing and broadcasting algorithms on the star graph.*

Property 6 *[5] There are $n-1$ node disjoint paths between any two nodes of S_n.*

Property 7 *[1, 3] The star graph is optimally fault tolerant.*

Since the star graph is both node symmetric and edge symmetric, routing between any pair of nodes can be reduced to the routing from a node to the identity node $(123\cdots n)$. As well as, routing between two nodes is equivalent to sort a permutation [1, 2]. The routing method (sorting a permutation) on a star graph is as follows.

1. If 1 is in the first position, move it to any position not occupied by the correct symbol.
2. If i, not equal to 1, is in the first position, then move it to position i.

For example, in S_8, suppose (23145687) is the source node, which can be rewritten as permutation cycles multiplication $(231) \circ (78) \circ (4) \circ (5) \circ (6)$. And suppose the identify node (12345678) is the destination node. One of the shortest routing paths is along $g_2 g_3 g_8 g_7 g_8$ as follows.
(23145687) $\stackrel{g_2}{\longleftrightarrow}$ (32145687) $\stackrel{g_3}{\longleftrightarrow}$ (12345687) $\stackrel{g_8}{\longleftrightarrow}$ (72345681) $\stackrel{g_7}{\longleftrightarrow}$ (82345671) $\stackrel{g_8}{\longleftrightarrow}$ (12345678)

Property 8 *[1] The minimum distance $d(\pi)$, from π to the identity permutation, is given by*

$$d(\pi) = c+m-\begin{cases} 0 & : \ if\ 1\ is\ in\ the\ first\ position. \\ 2 & : \ if\ 1\ is\ not\ in\ the\ first\ position. \end{cases}$$

where c denotes the number of cycles of length at least 2 and m is the total number of symbols not in their correct positions.

For example, the distance from $(23145687) = (231) \circ (78)$ to identity is $2 + 5 - 2 = 5$.

2.2 The Previous ST Algorithm on the Hypercube

There is a heuristic centralized multicast algorithm for constructing a Steiner tree on the hypercube, proposed by Lin and Ni [11]. The algorithm consists of two phases: (1) Phase 1: the message preparation phase, and (2) Phase 2: the message routing phase. In phase 1, the distance from the source to each destination node is calculated and then the destination nodes are sorted in ascending order according to the distances to the source node.

In phase 2, an important idea is that, for any three nodes s, t and u in the hypercube, a node v can be located in a constant time such that v is the nearest node to u among the nodes in all shortest paths from s to t. Let $u = u_{n-1}u_{n-2}...u_0$, $s = s_{n-1}s_{n-2}...s_0$, $t = t_{n-1}t_{n-2}...t_0$ and $v = v_{n-1}v_{n-2}...v_0$. Then v_j, $0 \leq j \leq (n-1)$, is set as $v_j = u_j$ if $s_j \neq t_j$ and $v_j = s_j$ if $s_j = t_j$. In phase 2, initially, the source node and one nearest node are linked as an initial Steiner tree. Then other destination nodes are sequentially added into the Steiner tree in the ascending order. That is, the Steiner tree is constructed in the nearest first order. Note that each node in the Steiner tree is a node on the hypercube and each edge in the Steiner tree corresponds to one path on the hypercube. Let T denote the current Steiner tree. When a node u, which is not in T now, is going to be added into T, we will find the nearest node to u on each edge in T. After all the edges in T are examined, an edge (x, y) is chosen such that the distance between u and (x, y) is the minimum. Suppose that v is the node which is the nearest to u on (x, y). Then (x, y) is deleted and three new edges (x, v), (v, y) and (u, v) are added into T. A new partial Steiner tree is constructed. After all destination nodes are added, the final Steiner tree is obtained.

2.3 The Previous MT Algorithm on the Star Graph

Tsay is the first one to propose a heuristic distributed MT algorithm on the star graph [14]. Since his algorithm is distributed, it is performed on each intermediate node. His algorithm is as follows. When an intermediate node w receives the message and the multicast set M, it has to check if itself is a destination node. If it is, it accepts the message locally and deletes itself from M. For each destination node u in M, the routing algorithm [1] is used to find all possible paths from w to u.

For example, in S_4, the possible paths from node 1234 to node 3412 are $g_3g_2g_4g_2$, $g_3g_4g_2g_4$, $g_2g_4g_2g_3$ and $g_4g_2g_4g_3$, which can be represented as the cycle multiplication (31)o(24). Thus, the first links possibly used for the path from 1234 to 3412 are g_3, g_2 and g_4. In an intermediate node w, all possible first links for each destination node can be found. And the occurrence of each link in the set of possible first links can be calculated. The most frequently possible first link is the preferred link, which is used to send the source message to the next node, because the maximum number of destination nodes will be included. All destination nodes which do not follow the preferred link still remain in node w. Then, the same procedure is repeatedly performed until all destination nodes in w are sent out.

Figure 2 shows an example of Tsay's algorithm. The source node is (12345) and there are nine destination nodes. From the set of possible first links, g_3 is selected as the preferred link, because g_3 appears 6 times. Then the source message is sent to node (32145).

3 A Heuristic ST Algorithm

In this section, we shall propose a heuristic centralized algorithm for the Steiner tree (ST) problem on star graphs. LinNi's ST algorithm on the hypercube consists of two phases: (1) Phase 1 (message preparation phase): sort the destination nodes in ascending order according to their distances to the source node. (2) Phase 2 (message routing phase): the destination nodes are added to the Steiner tree one by one in the ascending order (the nearest first order).

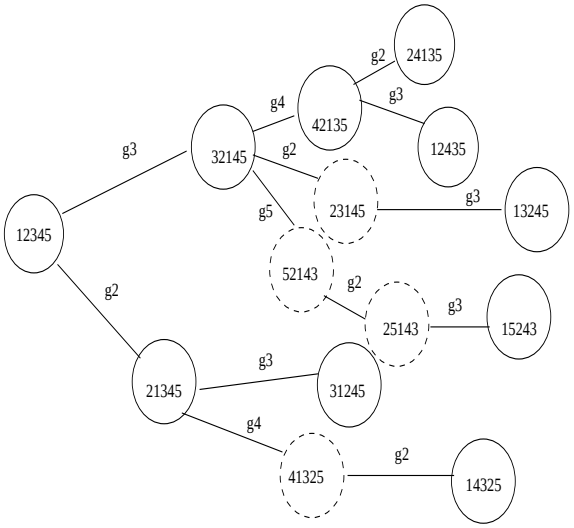
Basically, we shall follow these two phases to design our ST algorithm on the star graph. We find that the final Steiner tree is deeply dependent on the order of adding destinations into the Steiner tree. However, in LinNi's algorithm, the nearest first order may be

multicast set: { 32145,21345,42135,31245,24135,12435,13245,14325,15243 }

source node : 12345

Destination	Path	Possible first links
32145	(31)	$\{g_3\}$
21345	(21)	$\{g_2\}$
42135	(341)	$\{g_3\}$
31245	(231)	$\{g_2\}$
24135	(3421)	$\{g_3\}$
12435	(34)	$\{g_3, g_4\}$
13245	(32)	$\{g_3, g_2\}$
14325	(24)	$\{g_2, g_4\}$
15243	(235)	$\{g_3, g_5, g_2\}$

(a)



(b)

Figure 2: An example of the multicast tree obtained by Tsay's MT algorithm.

helpless to minimize the total amount of traffic. So, we will find a better order to add destinations to the Steiner tree. Given a multicast set M (including the source node), we first construct a complete graph $G = (V, E)$, where $V = M$ and $E = \{(u, v) | u, v \in V\}$. For each edge (u, v) in E, the weight $W(u, v)$ is defined as $H(u, v)$. We find a minimum spanning tree T on G. And we arrange the nodes into the breadth first order. In other words, in phase 1, we find the breadth first order on T instead of the nearest first order. Then, in phase 2, the destination nodes are added into the Steiner tree one by one according to the breadth first order.

The second problem for us is when we want to add one destination node to the Steiner tree, ho do we add it by connecting it to the nearest node in the current Steiner tree? There are two types of nodes in a Steiner tree: (1) *tree node*: a destination node in the multicast set; and (2) *relay node*: an intermediate node and a joint of some tree nodes. Each edge (u, v) in the Steiner tree corresponds to a set of shortest paths between u and v in the star graph, which is called the *virtual path*, denoted as $vir(u, v)$, between u and v. $|vir(u, v)|$ is used to denote the length of one of the shortest paths connecting u and v.

In phase 2, when we add a destination node to the current Steiner tree, the most important is to find a node, a relay node or a tree node, to which the newly added node has the minimum distance. The relay node (or a tree node) can be found by the following way.

Step 1: To decide the virtual paths. Let u be the node to be added into the current Steiner tree $T = (V_T, E_T)$. For each edge (x, y) in T, construct the virtual path from x to u and from y to u, denoted as $vir(x, u)$ and $vir(y, u)$, respectively. An example is shown in Figure 3 (a).

Step 2: To find the matching path. For each edge (x, y) in T, find the matching path between $vir(x, y)$ and $vir(x, u)$, and the matching path between $vir(y, x)$ and $vir(y, u)$. Then the relay node on (x, y), denoted as $relay(x, y)$, to which u have the minimum distance among all nodes in $vir(x, y)$, can be obtained by calculating $min\{|vir(x, u)| - $matching length of $vir(x, y)$ and $vir(x, u)$, $|vir(y, u)| - $matching length of $vir(y, x)$ and $vir(y, u)\}$.

Step 3: To determine the relay node. For each edge (x, y) in T, now we have the distance between $relay(x, y)$ and u. The relay node for u on the current Steiner tree is determined by the minimum distance.

The matching path of two virtual paths can be found by the following method.

Algorithm Matching Path

Input: The permutation representations $(a_1 a_2 ... a_n)$ and $(b_1 b_2 ... b_n)$ of two virtual paths $vir(x, y)$ and $vir(x, u)$ respectively.

Output: The matching path of $vir(x, y)$ and $vir(x, u)$.

Notations: FM: the full matching set
PM: the partial matching set
ME: the matching element

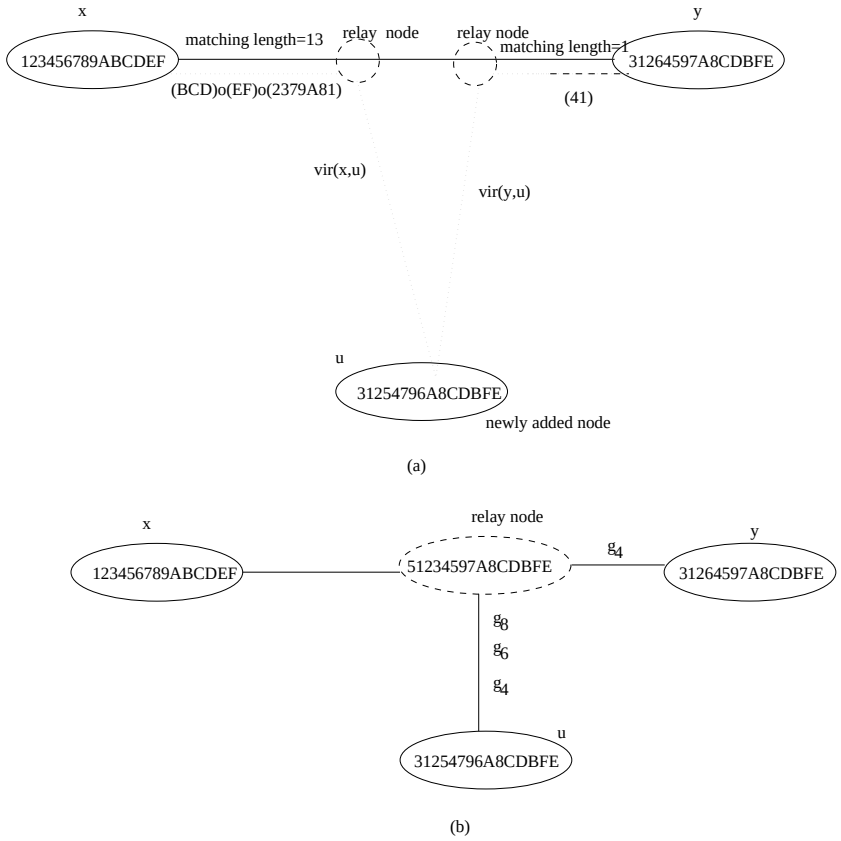


Figure 3: Finding the relay node on an edge.

Step 1: $FM = \emptyset$, $PM = \emptyset$.

Step 2: If all positions of $(a_1 a_2 .... a_n)$ and $(b_1 b_2 ... b_n)$ are checked, go to step 6. Let j be the first unchecked position. Set ME to a null string.

Step 3: If $(a_j = j)$ or $(b_j = j)$ or $(j = 1 \ and \ a_j \neq b_j)$
go to step 2
else
append j to ME

Step 4: $i \leftarrow j$

Step 5: If $a_i = b_i$
append a_i to ME
$i \leftarrow a_i$
If $i = j$
add ME to FM
go to step 2
else
go to step 5
else
add ME to PM
go to step 2

Step 6: The matching path consists of all elements in FM and the element in PM with the maximum length. Calculate the matching length.

The matching path is divided into two parts. One is the full matching set, the other is the partial matching set. Each element in the full matching set is of a permutation cycle form, but the elements in the partial matching set are not of cycle form. We shall first select all elements in the full matching set, and then append one longest element in the partial matching set to the matching set. The matching set forms the matching path.

As an example, consider Figure 3. The matching path of $vir(x, y)$ and $vir(x, u)$ can be found as follows.

$$\begin{aligned} vir(x,y) &= \begin{pmatrix} 3\,1\,2\,6\,4\,5\,9\,7\,A\ 8\ C\ D\ B\ F\ E \\ 1\,2\,3\,4\,5\,6\,7\,8\,9\ A\ B\ C\ D\ E\ F \end{pmatrix} \\ &= (2\ 3\ 1\ 5\ 6\ 4\ 8\ A\ 7\ 9\ D\ B\ C\ F\ E) \\ &= (2\ 3\ 1) \circ (4\ 5\ 6) \circ (7\ 8\ A\ 9) \circ (B\ D\ C) \\ &\quad \circ (E\ F) \end{aligned}$$

$$
\begin{aligned}
vir(x,u) &= \begin{pmatrix} 3\,1\,2\,5\,4\,7\,9\,6\,A\,8\,C\,D\,B\,F\,E \\ 1\,2\,3\,4\,5\,6\,7\,8\,9\,A\,B\,C\,D\,E\,F \end{pmatrix} \\
&= (2\ 3\ 1\ 5\ 4\ 8\ 6\ A\ 7\ 9\ D\ B\ C\ F\ E) \\
&= (2\ 3\ 1) \circ (4\ 5) \circ (6\ 8\ A\ 9\ 7) \circ (B\ D\ C) \\
&\quad \circ (E\ F)
\end{aligned}
$$

We get the full matching set {(123), (BDC), (EF)}= {(BDC), (EF), (231)}, whose length is 4+3+2=9. And the partial matching set is {45, 6, 7, 8A97}, in which 8A97 has the maximum length 4. Hence, the final matching set is $\{(BDC), (EF), (231), 8A97\}$, which corresponds to the virtual path $(BDC) \circ (EF) \circ (238A971)$. Its length is 4+3+6=13. We can determine the relay node from x to u by the matching path. Thus, the relay node is $(4325678A9BCDFE)$, and the length from it to u is equal to $18 - 13 = 5$, where $18 = |vir(x,u)|$.

For another virtual path $vir(y,u)$, we can find the matching path of $vir(y,x)$ and $vir(y,u)$.

$$
\begin{aligned}
vir(y,x) &= \begin{pmatrix} 1\,2\,3\,4\,5\,6\,7\,8\,9\,A\,B\,C\,D\,E\,F \\ 3\,1\,2\,6\,4\,5\,9\,7\,A\,8\,C\,D\,B\,F\,E \end{pmatrix} \\
&= (3\ 1\ 2\ 6\ 4\ 5\ 9\ 7\ A\ 8\ C\ D\ B\ F\ E) \\
&= (3\ 2\ 1) \circ (4\ 6\ 5) \circ (7\ 9\ A\ 8) \circ (B\ C\ D) \\
&\quad \circ (E\ F) \\
vir(y,u) &= \begin{pmatrix} 3\,1\,2\,5\,4\,7\,9\,6\,A\,8\,C\,D\,B\,F\,E \\ 3\,1\,2\,6\,4\,5\,9\,7\,A\,8\,C\,D\,B\,F\,E \end{pmatrix} \\
&= (1\ 2\ 3\ 8\ 5\ 4\ 7\ 6\ 9\ A\ B\ C\ D\ E\ F) \\
&= (4\ 8\ 6)
\end{aligned}
$$

The full matching set is empty and the partial matching set is $\{4, 6, 8\}$, in which 4 has the maximum length is 1. Hence, the final matching set is {4}, which corresponds to the virtual path (41). Its length is 1. The relay node from y to u is $(61234597A8CDBFE)$, and the length from it to u is equal to $4 - 1 = 3$, where $4 = |vir(y,u)|$. To u, the relay node $(61234597A8CDBFE)$ has the minimum distance $3 = min\{5, 3\}$, thus, it is the relay node for u on edge (x,y), i.e. it is $relay(x,y)$.

The complete algorithm for solving the Steiner tree problem is as follows.

Algorithm Steiner Tree (ST)

Phase 1: message preparation phase

Input: Source node u_0 and multicast set M.

Output: A destination list in the breadth first order BF.

Step 1.1: Construct a complete graph $G = (V, E)$, where $V = \{u_0\} \cup M$, $E = \{(u,v) \mid u, v \in V\}$, and the weight of edge (u,v) is equal to $H(u,v)$.

Step 1.2: Find a minimum spanning tree T of G.

Step 1.3: Perform the breadth first search on T, rooted at u_0. Then, the breadth first order BF is obtained.

Phase 2: message routing phase

Input: The destination list in the breadth first order $BF = \{u_0, v_1, v_2, \cdots, v_k\}$

Output: A Steiner tree (ST) on the star graph.

Step 2.1: Initially, the ST contains only one edge (u_0,v_1). Each destination node v_i in $BF - \{u_0, v_1\}$ is sequentially added to the Steiner tree with method in Step 2.2:

Step 2.2: **(a)** For each edge (x,y) in ST, find $relay(x,y)$ on it for v_i. Calculate $|vir\ (relay\ (x,y),\ v_i)|$.

(b) Find $min_{(x,y) \in ST}\ \{|vir(relay(x,y),\ v_i)|\}$. Let (x_i, y_i) be such an edge achieving the minimum value and its relay node is r.

(c) If r is equal to neither x_i nor y_i, add two nodes v_i and r, and add three edges (x_i, r), (y_i, r) and (v_i, r) into the current ST, and then delete edge (x_i, y_i). If r is equal to x_i, then add node v_i and add edge (x_i, v_i). If r is equal to y_i, then add node v_i and add edge (y_i, v_i).

Let the number of nodes in the multicast set be k. In phase 1, $O(nk^2)$ time is required since there are $O(k^2)$ edges in the complete graph G and the weight of each edge can be obtained in $O(n)$ time. In phase 2, there are at most k relay nodes since at most one relay node is added when one destination node is added into the Steiner tree. Thus, there are $O(k)$ nodes in the Steiner tree. The time required for adding one destination node to the Steiner tree is $O(nk)$ since $O(k)$ edges are checked and $O(n)$ time is required for finding the relay node on an edge. Hence, phase 2 requires $O(nk^2)$ time. Therefore, the total time required for our ST algorithm is $O(nk^2)$. Note that if we implement LinNi's algorithm on the star graph, it also needs $O(nk^2)$ time.

Let's show an example to illustrate how Algorithm ST works. Suppose, in S_4, node 1234 wants to send a message to each destination node in the multicast set { 3412, 4312, 3142, 1243, 2341, 3421 }. First, an assisting complete graph G is constructed and a minimum spanning tree T on G is found, as shown in Figure 4. After a breadth first search is performed on T, we obtain the breadth first order $BF = \{1234,$

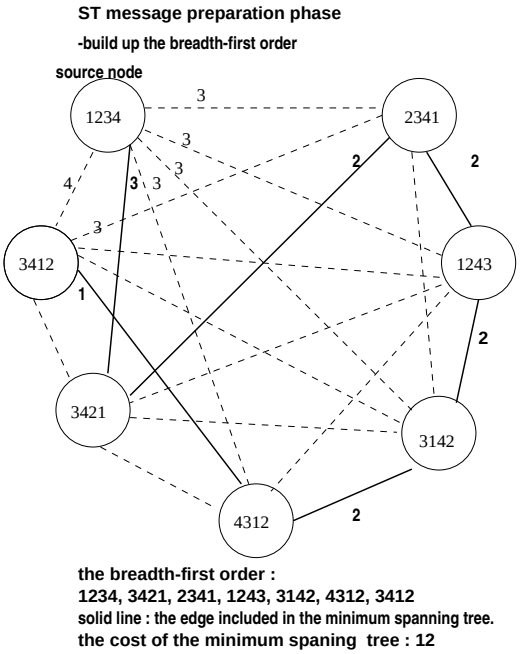


Figure 4: An example of the minimum spanning tree obtained by our ST algorithm.

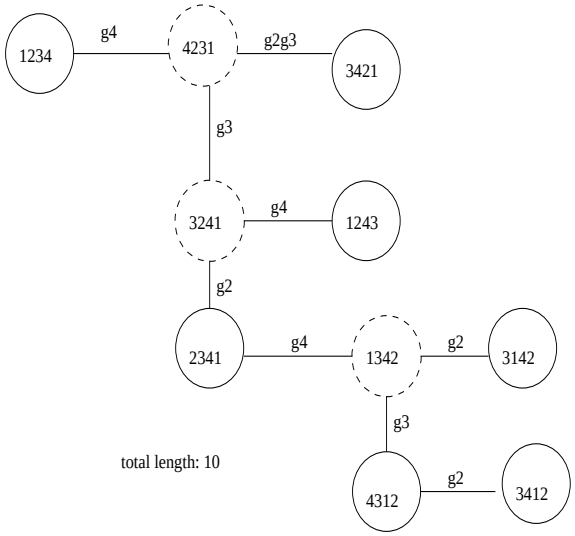


Figure 5: An example of a Steiner tree constructed by our ST algorithm.

3421, 2341, 1243, 3142, 4312, 3412 }. Finally, in phase 2, the Steiner tree is obtained, as shown in Figure 5, and the total cost is 10.

Figure 6 illustrates the implementation of LinNi's ST algorithm on the star graph with the same example. The nearest first order is NF = {1234, 1243, 3421, 4312, 3142, 2341, 3412 }, and the total cost is 12.

4 A Heuristic MT Algorithm

Tsay proposed a simple heuristic distributed multicast tree (MT) algorithm for the star graph [14]. In his algorithm, each intermediate node chooses the preferred link to send the source message to the next node.

We will propose an MT algorithm to improve Tsay's algorithm. In the multicast tree problem, for a 1-to-k multicast, at least k nodes are needed to construct a multicast tree. Our basic idea is when u and v are adjacent, putting these two nodes on the same path of the multicast tree will be never worse than putting them on different paths. Our heuristic MT algorithm consists of two phases: (1) phase 1: neighbors linking phase and (2) phase 2: message routing phase. Phase 1, executed only on the source node, links the neighboring destination nodes to reduce the size of the

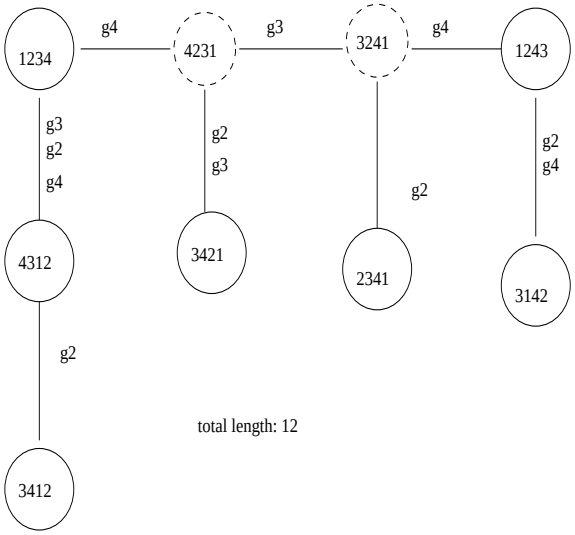


Figure 6: An example of a Steiner tree constructed by LinNi's ST algorithm.

multicast set. After this phase, the multicast set becomes a neighboring forest T and each element in T is a root of a neighboring tree. Phase 2, executed on each intermediate node in the multicast tree, is similar to Tsay's algorithm but a little different. If the local node w is an element in T, then w is deleted from T and all sons of w in the neighboring tree are added to T as new elements. Then the preferred link g_j which has the maximum count value is selected. Note that we only count the possible first links of the elements on the current T. Then the message is sent along link g_j from the local node w to its neighbor. Here, if some of the destinations can not be reached through the neighbor via a shortest path, they temporarily stay in w. Since each element v in T is a root of a neighboring tree, all elements in the neighboring tree rooted by v are also sent with v. Then, all these elements sent to the neighbor are deleted from T on w. This procedure repeats until T on w is empty. Finally, a multicast tree is constructed. Our heuristic ST algorithm is formally described as follows.

Algorithm Multicast Tree (MT)
Phase 1: neighbors linking phase

Input: Source node u_0, multicast set M =$\{u_1, u_2, \cdots, u_k\}$ and the diameter m of S_n.

Output: A neighboring forest T.

Step 1: Let $M_0 = \{u_0\}$. Partition M into m disjoint sets M_1,M_2,...M_m, such that for each node v in M_i, $H(v, u_0) = i$. Note that M_i may be empty.

Step 2: For $i = m, m-1, \cdots, 1$ do
For each element $u \in M_i$ do
Check if u has an adjacent node in M_{i-1}
If so, say v, let v be the father of u and delete u from M_i.

Step 3: Let $T = M_1 \cup M_2 \cup ... \cup M_m$.

Phase 2: message routing phase

Input: Local node w and a neighboring forest T of S_n.

Goal: To build the multicast tree.

Step 1: If $w \in T$ then
Accept the message locally.
Let $T = T - \{w\} \cup \{v \mid v$ is the son of w in the neighboring tree $\}$.

Step 2: While $T \neq \emptyset$ do

(a) For $2 \leq i \leq n$ do
Let V_i =$\{v | v \in T$, the link $(w, g_i(w))$ can be included in a shortest path from w to $v\}$.
Let $count[i] = |V_i|$. Note that $V_i \bigcap V_j$, $i \neq j$, may not be empty.

(b) Let $count[j]$=$max_{2 \leq i \leq n}\{count[i]\}$.

(c) Transmit V_j and the message to node $g_j(w)$.

(d) $T = T - V_j$.

endwhile

Our MT algorithm is hybrid since phase 1 is executed only in the source node, and phase 2 should be executed in each intermediate node. In our main idea, if a destination node has an ancestor on the neighboring forest, it is hidden behind a common ancestor until this ancestor is reached. Clearly, every destination node receives the source message via the shortest path from the source node. In our MT algorithm, that the source node links neighboring nodes to form a smaller multicast set can help the intermediate nodes to make good routing decisions.

In phase 1, step 1 needs $O(nk)$ time since the distance from a destination node to the source can be calculated in $O(n)$ time, and step 2 needs $O(k^2)$ time. Thus, $O(nk + k^2)$ time is required for this phase. In phase 2, $O(nk)$ time is required for each intermediate node since step 2(a) takes $O(nk)$ time.

Figure 7 shows an example to illustrate our algorithm working in S_5. Initially, in the source node, we partition the multicast set M and obtain $M_1 = \{32145, 21345\}$, $M_2 = \{42135, 31245\}$, $M_3 = \{24135, 12435, 13245, 14325\}$, and $M_4 = \{15243\}$. The result of the neighbors linking phase, as shown in Figure 7(a), is $M_1 = \{32145, 21345\}$, $M_3 = \{14325\}$, $M_4 = \{15243\}$, and $M_2 = \emptyset$. Each element of M_i represents the root of a neighboring tree. Thus, the neighboring forest $T = M_1 \cup M_2 \cup M_3 \cup M_4$, which is equal to $\{32145, 21345, 15243, 14325\}$. Then, phase 2 is executed with T as input. As shown in Figure 7(b), the count values of possible first links g_2, g_3 ,g_4 ,and g_5 are 3, 2, 1, and 1 respectively. Thus, $V_j = V_2 =$ { 21345, 14325, 15243 } with the source message is transmitted to $g_2(12345)$. Repeating this procedure in all intermediate nodes, the final multicast tree, as shown in figure 7(c), can be obtained. The communication traffic of our multicast tree is 11. While with the same source and multicast set, the cost of the multicast tree obtained by Tsay's MT algorithm is 13, as shown in Figure 2.

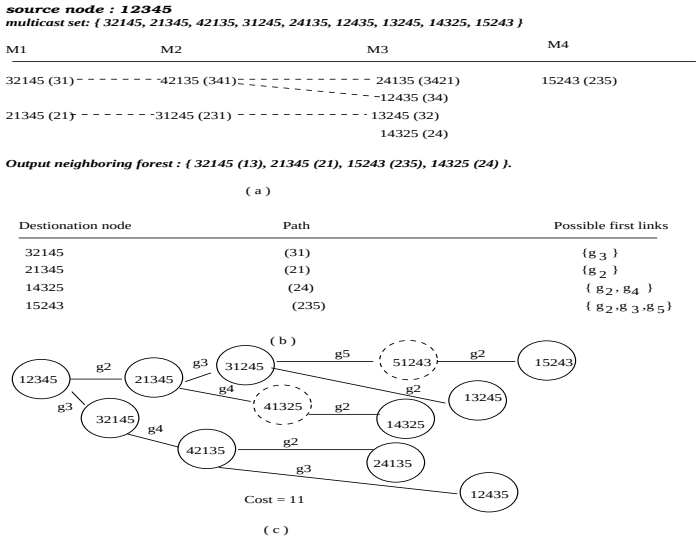


Figure 7: An example of the multicast tree obtained by our MT algorithm. (a) The neighbors linking phase. (b) The paths and possible first links from the source to the elements in the neighboring forest. (c) The final multicast tree.

5 Performance Analysis

In this section, we shall present the the performance of our algorithms by some simulation experiments. A large number of randomly generated multicast sets with different number of destinations are tested to measure the traffic generated by the algorithms. For a 1-*to*-k multicast, it requires at least k units of traffic. The *additional traffic* is defined as the total amount of traffic minus k, where a unit of traffic is measured as one message traversed over one link. As in the previous studies [11, 13, 15], we use *average additional traffic* to evaluate the performance of a multicast algorithm. We shall compare the performance of our heuristic algorithms with two other alternative approaches for multicasting, namely, LinNi's ST algorithm and Tsay's MT algorithm.

Figures 8 and 9 show the average additional traffic obtained by simulating ST and MT algorithms, respectively. In Figure 8, the performance of our ST algorithm is compared with LinNi's ST algorithm on S_5. The number of destination nodes is chosen from 10 to 110 step by 10. Clearly, by the experiment results, the performance of our ST algorithm is better than that of LinNi's ST algorithm. In Figure 9, the performance of our MT algorithm is compared with Tsay's MT algorithm on S_6. The number of destination nodes is chosen from 100 to 700 step by 50. Our MT algorithm is also superior to Tsay's MT algorithm. Thus, our heuristic algorithms usually generates less

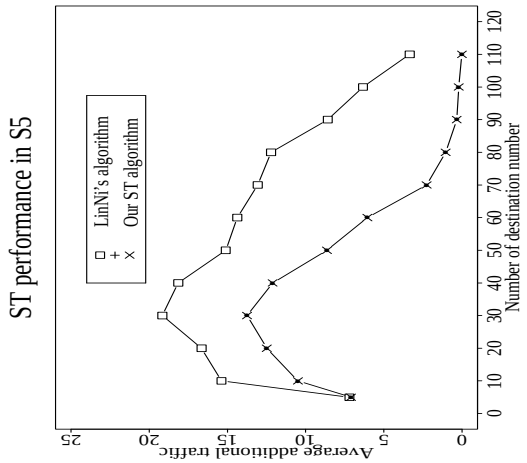


Figure 8: Performance of ST algorithms on S_5.

amount of traffic compared with these two previous approaches.

6 Conclusion

The star graph, like the hypercube, has rich structure and symmetry as well as fault-tolerant capabilities. In addition, the star graph has a smaller diameter and degree than the hypercube. Thus, it has

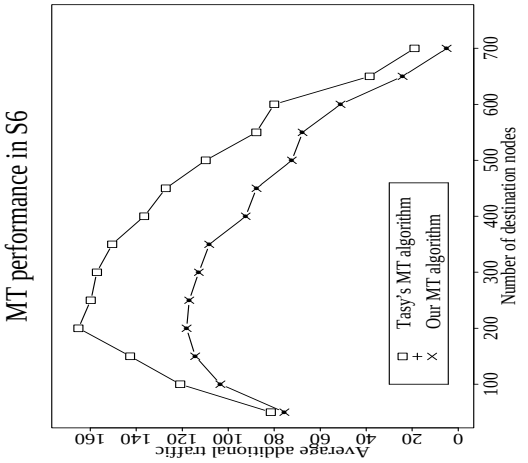


Figure 9: Performance of MT algorithms on S_6.

been shown to be an attractive alternative to the hypercube. Lin and Ni formulated the multicast communication problem in multicomputers as four different graph theoretical problems [11]. In this paper, we study the multicast problem on star graphs and propose two heuristic algorithms for solving the Steiner tree (ST) and multicast tree (MT) problems.

We also implement LinNi's ST algorithm, originally designed for the hypercube, on the star graph. By the experiment results, our algorithms are superior to LinNi's ST algorithm and Tsay's MT algorithm.

References

[1] S. B. Akers, D. Horel, and B. Krishnamurthy, "The star graph:an attrative alternative to the n-cube," *Proceedings of the International Conference on Conference on Parallel Processing*, pp. 393–400, 1987.

[2] S. B. Akers and B. Krishnamurthy, "A group-theoretical model for symmetric interconnection networks," *IEEE Transactions on Computers*, Vol. 38, No. 4, pp. 555–566, 1989.

[3] B. Alspach, "Cayley graphs with optimal fault tolerance," *IEEE Transactions on Computers*, Vol. 41, No. 10, pp. 1337–1339, 1992.

[4] H. D. Cheng, Y. Y. Tang, and C. Y. Suen, "Parallel image transformation and its VLSI implementation," *Pattern Recognition*, Vol. 23, No. 10, pp. 1113–1129, 1990.

[5] K. Day and A. Tripathi, "A comparative study of topological properties of hypercubes and star graphs," *IEEE Transactions on Parallel and Distributed Systems*, Vol. 5, No. 1, pp. 31–38, Jan. 1994.

[6] R. F. DeMara and D. I. Moldovan, "Performance indices for parallel marker-propagation," *Proc. 1991 Int. Conf. Parallel Processing*, Vol. 1, pp. 658–659, Aug. 1991.

[7] R. L. Graham and L. R. Foulds, "Unlikelihood that minimal phylogenies for realistic biological study can be constructed in reasonable computational time," *Mathematical Biosci*, Vol. 60, pp. 133–142, 1982.

[8] S. C. Hu and C. B. Yang, "Fault tolerance on star graphs," *accepted by Int. Journal of Foundations of Computer Sciences*, 1997.

[9] V. Kumar and V. Singh, "Scalability of parallel algorithms for the all-pairs shortest-path problem," *Journal of Parallel and Distributed Computing*, Vol. 10, pp. 124–138, Oct. 1991.

[10] Y. Lan, A. H. Esfahanian, and L. M. Ni, "Multicast in hypercube multiprocessors," *Journal of Parallel and Distributed Computing*, Vol. 8, pp. 30–41, 1990.

[11] X. Lin and L. M. Ni, "Multicast communication in multicomputer networks," *IEEE Transactions on Parallel and Distributed Systems*, Vol. 4, No. 10, pp. 1105–1117, 1993.

[12] V. E. Mendia and D. Sarkar, "Optimal broadcasting on the srar graph," *IEEE Transactions on Parallel and Distributed Systems*, Vol. 3, No. 4, pp. 389–396, July 1992.

[13] J. P. Sheu and M. Y. Su, "A multicast algorithm for hypercube multiprocessors," *Parallel Algorithms and Applications*, Vol. 2, pp. 277–290, 1994.

[14] W. S. Tsay, *Multicasting and fault-tolerant design on the star graph.* Master Thesis, Institute of Information Engineering, National Sun Yat-sen University, Kaohsiung, Taiwan, June 1993.

[15] C. B. Yang and S. H. Sheu, "Multicast algorithms on hypercube multiprocessors," *Proceedings of 1995 Workshop on Distributed System Technologies & Applications*, Tainan, Taiwan, pp. 163–170, July 1995.