Congestion-Free, Dilation-2 Embedding of Complete Binary Trees into Star Graphs*

Yu-Chee Tseng†, Yuh-Shyan Chen‡, Tong-Ying Juang‡, and Chiou-Jyu Chang‡

†Department of Computer Science and Information Engineering
National Central University
Chung-Li 32054, Taiwan
Tel: 886-3-4227151 ext. 4512, Fax: 886-3-4222681
yctseng@csie.ncu.edu.tw

‡Department of Computer Science and Information Engineering
Chung-Hua University
Hsin-Chu, 30067, Taiwan
Tel: 886-3-5374281 ext. 8313, Fax: 886-3-5373771
chenys@csie.chu.edu.tw

July 27, 1998

Abstract

Trees are a common structure to represent the inter-task communication pattern of a parallel algorithm. In this paper, we consider the embedding a complete binary tree in a star graph with the objective of minimizing congestion and dilation. We develop two embeddings: i) a congestion-free, dilation-2, load-1 embedding of a level-p binary tree, and ii) a congestion-free, dilation-2, load-2^k embedding of a level-$(p+k)$ binary tree, into an n-dimensional star graph, where $p = \Sigma_{i=2}^{n} \lfloor \log i \rfloor = \Omega(n \log n)$ and k is any positive integer. The first result offers a tree of size comparable or superior to existing results, but with less congestion and dilation. The second result provides more flexibility in the embeddable tree sizes compared to existing results.

Keywords: Graph embedding, interconnection network, complete binary tree, star graph, parallel processing.

*This research was supported in part by the National Science Council, R.O.C., under grant numbers NSC87-2213-E-008-012, NSC87-2213-E-008-016, and NSC87-2213-E-216-004.

1. Introduction

One new interconnection network that has attracted lots of attention recently is the star graph [1, 2]. The star graph, being a member of the Cayley graphs, has been shown to possess several appealing features such as its diameter-to-node-degree ratio, scalability, partitionabilty, symmetry, and high degree of fault tolerance. Indeed, a lot of research force has been put forward in studying the star graph's topological properties [9, 23], embedding capability [3, 5, 6, 11, 12, 14, 16, 21, 25], communication capability [10, 18, 19, 22, 23, 24], and fault-tolerant capability [4, 7, 8, 13, 15, 25].

One of the central issues in evaluating a network is to study the *graph embedding problem* [17, 20]. Given a *guest graph* $G = (V, E)$ and a *host graph* $H = (V', E')$, the problem is to find a mapping from each node $\in V$ to one of V', and a mapping from each edge $\in E$ to one path in H. The problem has long been used to model the problem of arranging a parallel algorithm in a parallel architecture. There are several measurements to estimate an embedding. The *expansion* of an embedding is the ratio of the number of vertices of H to that of G. The *load* of an embedding is the maximum of the number of times a node of H mapped by those of G, over all nodes of H. The *vertex congestion* is the maximum number of edges of the guest graph routed through a single node over all nodes of the host graph [5, 11]. The *dilation* is the maximum of the lengths of paths mapped by all edges of E. The *congestion* is the maximum of the number of times that an edge of H is used in the mapping, over all the edges of H. When the embedding congestion is 1, we may interchangeably adopt the term *congestion-free* to reflect the fact that no two edges of E share a same link in E'.

This paper considers the problem of embedding a complete binary tree into an n-dimensional star graph S_n. Binary trees are a common structure of parallel algorithms (such as those using divide-and-conquer strategies). In the literature, several results have been proposed to this problem, with different values of congestion, dilation, and expansion incurred [3, 5, 6, 11, 12, 16]. Table 1 summarizes these results and compares them to our results under some embedding measurements[1]. As to congestion, all results are congestion free except [3], which incurs high congestion (the exact congestion is not analyzed in [3]). As to dilation, the schemes proposed by [5, 6] and [16] offer a dilation-1 embedding; however, the expansion of these two schemes is pretty large and thus a lot of nodes are left un-embedded. Our dilation of 2 is equal to that of [3], and outperforms

[1]Throughout this paper, log means $\log_2$, unless otherwise stated.

Table 1: Comparison of some binary tree embedding schemes.

	congestion	dilation	level of tree
Lee and Chang [16]	1	1	$n-1$
Hsu [12] (load> 1)	1	3	$\Sigma_{i=2}^{n}\lfloor \log i \rfloor$
Bouabdallah *et al.* [5, 6]	1	1	$(1/2+\varepsilon(n))n\log n+1$, $\varepsilon(n)\to 0$ if $n\to\infty$
Heydemann *et al.* [11]	1	4	$(n+1)\lfloor \log n \rfloor - 2^{\lfloor \log n \rfloor+1}+2$
Bagherzadeh *et al.* [3]	high	$O(1)$	$\lfloor \log(\lfloor e(n-1)! \rfloor) \rfloor$, $e\approx 2.718282$
Ours (load-1)	1	2	$\Sigma_{i=2}^{n}\lfloor \log i \rfloor$
(load-2^k)	1	2	$\Sigma_{i=2}^{n}\lfloor \log i \rfloor + k$

the rest two schemes [11, 12].

Another measurement is how large the embeddable tree is, which is reflected in the last column of Table 1. As it is hard to judge from merely formulas, we demonstrate this through Fig. 1, which shows the levels of trees embeddable under various dimensions (n). As can be seen, ours is only slightly smaller than that of [3] (which has the weakness of high congestion), but is comparable to or larger than all other schemes. As a result, our embedding can offer fairly large binary trees while at the same time improving over existing results in both congestion and dilation. It should also be noted that $\Sigma_{i=2}^{n}\lfloor \log i \rfloor = \Omega(n\log n)$ levels is the best possible (asymptotically) result one can achieve if a load of one is a prerequisite.

The way our embedding is achieved is also interesting by itself. The main technique is to split a binary tree into a number of small *threaded trees*, where a threaded tree is a binary tree attached with a linear path at its root node. Threaded trees are in fact the basic components in our embedding. Each threaded tree can be embedded in a separate substar, thus making easy extending our embedding to star graphs of any dimension. The details are in Section 3..

In addition, we also develop an alternative which can embed a binary tree of any level (refer to the bottom row in Table 1, where k is any positive integer). By adjusting the value of k, a binary tree of any size can be embedded. This will somehow increase the load, but it is desirable to have such a result for flexibility reason. The main technique is based on "folding" some nodes near the leaves of the binary tree to their ancestor nodes (refer to Section 4 for details).

The rest of this paper is organized as follows. Preliminaries are in Section 2. Our main embedding scheme is presented in Section 3. Section 4 shows how to embed

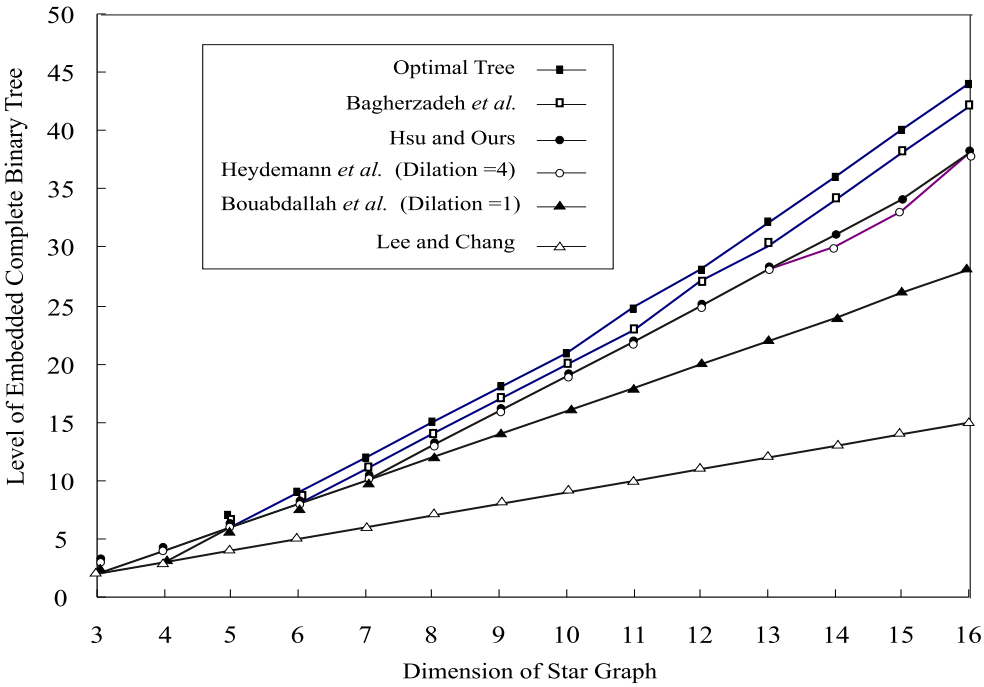

Figure 1: Comparison of levels of trees embeddable by some schemes.

larger complete binary trees, if higher load is allowed. Finally, conclusions are drawn in Section 5.

2. Preliminaries

An n-dimensional star graph, also referred to as n-star or S_n, is an undirected graph consisting of $n!$ nodes (vertices) and $(n-1)n!/2$ edges (links). Each node is uniquely assigned a label $x_1x_2\cdots x_n$, which is a permutation of any n distinct symbols. Without loss of generality, let these n symbols be $\{1, 2, \ldots, n\}$. Given any node $x = x_1\cdots x_i\cdots x_n$, define the permutation function $g_i, 2 \leq i \leq n$, to be $g_i(x) = x_i\cdots x_1\cdots x_n$ (i.e., swap x_1 and x_i and keep the rest 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. Each node x has $n-1$ edges, each connecting to $g_i(x)$ along dimension $i, i = 2..n$.

The star graphs have a nice recursive property [25]. Let $\Gamma = \{1, 2, \ldots, n, *\}$, where $*$ means "*don't care.*" A *k-dimensional substar*, or *k-substar*, is denoted by a string $G = x_1x_2\cdots x_n$ such that $x_1 = *$, $x_i \in \Gamma, 2 \leq i \leq n$, and there are exactly k $*$'s in G. G is a subgraph of S_n consisting of all legal vertices obtained from G by replacing each $*$ by a non-$*$ symbol, and all edges induced by these vertices in S_n. For instance, an

S_n can be partitioned into n disjoint $(n-1)$-substars $*^{n-1}i$, $i = 1..n$, where $*^{n-1}$ is a sequence of $n-1$ $*$'s.

A complete binary tree of p levels (with $2^p - 1$ nodes) is denoted as T_p. The leaf nodes are said to be *at* level 1, and the *levels* of other nodes are defined recursively and incrementally from the leaves. We will use the standard terminology in graph theory, such as parent, child, left, and right to relate nodes in a binary tree.

Definition 1 A *threaded tree* $TT_{i,p}$ is a tree obtained from a binary tree T_p by attaching a linear path (called *thread*) of i links at the root of T_p. As a convention, the end node of the thread of degree 1 (after the attachment) is regarded as the root of a threaded tree.

For instance, Fig. 3 is a threaded tree $TT_{3,6}$, which consists of a thread of 3 links at the top and a binary tree of 6 levels at the bottom. Node 23451 is the root of the threaded tree.

3. The Embedding Algorithm

Let $\rho(k) = \Sigma_{i=2}^{k} \lfloor \log i \rfloor$. We will develop a congestion-free, dilation-2, load-1 embedding of $T_{\rho(n)}$ into S_n, $n \geq 3$. When $n = 3$, the embedding is trivial. When $n = 4$, one possible embedding is already shown in Fig. 2. Each tree edge is labeled by one or two numbers, which indicate the dimensions of the links connecting the two end nodes of the tree edge. For instance, the tree edge connecting nodes 4231 and 2341 is labeled by "3, 2", meaning that 4231 goes first along dimension 3, followed by dimension 2, to node 2341.

When $n \geq 5$, the embedding is established by induction. We will show that there always exists a congestion-free, dilation-2 embedding of a threaded tree $TT_{\alpha(n),\rho(n)}$ into S_n such that

1. $\alpha(n) = \lfloor \log(n+1) \rfloor + 1$, and
2. all $\alpha(n)$ links on the thread of $TT_{\alpha(n),\rho(n)}$ are along distinct dimensions.

Why such a tree is needed will become clear during the derivation. Clearly, this implies a correct embedding from $T_{\rho(n)}$ to S_n.

The induction basis is at $n = 5$, for which case we have shown in Fig. 3 an embedding of $TT_{\alpha(5),\rho(5)} = TT_{3,6}$ into S_5. The embedding is obtained by constructing in each of

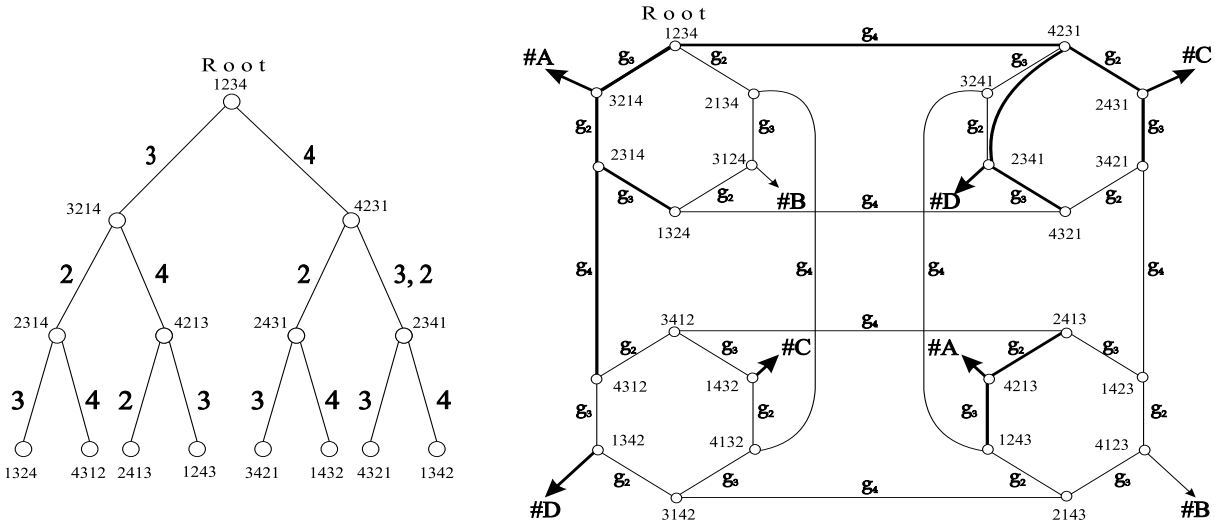

Figure 2: An embedding of a complete binary tree $T_{\rho(4)} = T_4$ in S_4.

the four substars, $****5, ****3, ****4$, and $****2$, a T_4, and connecting these four T_4's together with a threaded tree $TT_{3,2}$ in substar $****1$. The embedding has a congestion of 1 and dilation of 2 (refer to Fig. 3). However, congestion and dilation will not be increased as the induction is being established.

Now for the induction hypothesis suppose the existence of an embedding from $TT_{\alpha(n-1),\rho(n-1)}$ to S_{n-1}. We will establish an embedding from $TT_{\alpha(n),\rho(n)}$ to S_n. Observe that $T_{\rho(n)}$ has $\lfloor \log n \rfloor$ more levels than $T_{\rho(n-1)}$ does. The main idea is to regard $T_{\rho(n)}$ as a $T_{\lfloor \log n \rfloor + 1}$ in which each leaf represents a tree $T_{\rho(n-1)}$. We outline the construction steps as follows:

Step 1: Construct a threaded tree $\hat{T} = TT_{\alpha(n),\lfloor \log n \rfloor + 1}$ such that each of its leaves is located in a distinct $(n-1)$-substar.

Step 2: Embed, based on the induction hypothesis, in each of the aforementioned $(n-1)$-substars a $T_{\rho(n-1)}$ rooted at each leaf of $\hat{T}$.

Intuitively, the threaded tree $TT_{\alpha(n),\rho(n)}$ can be partitioned into a number of smaller components (each can be a linear path or a smaller threaded tree) by cutting some edges of $TT_{\alpha(n),\rho(n)}$ near the root of the binary tree. Through distributing these components into different S_{n-1}'s, the congestion, load, and dilation issues will be taken care of. Depending on whether n is a power of 2 or not, the thread of $TT_{\alpha(n),\rho(n)}$ will be either an individual component or as a part of some threaded tree. The latter happens when

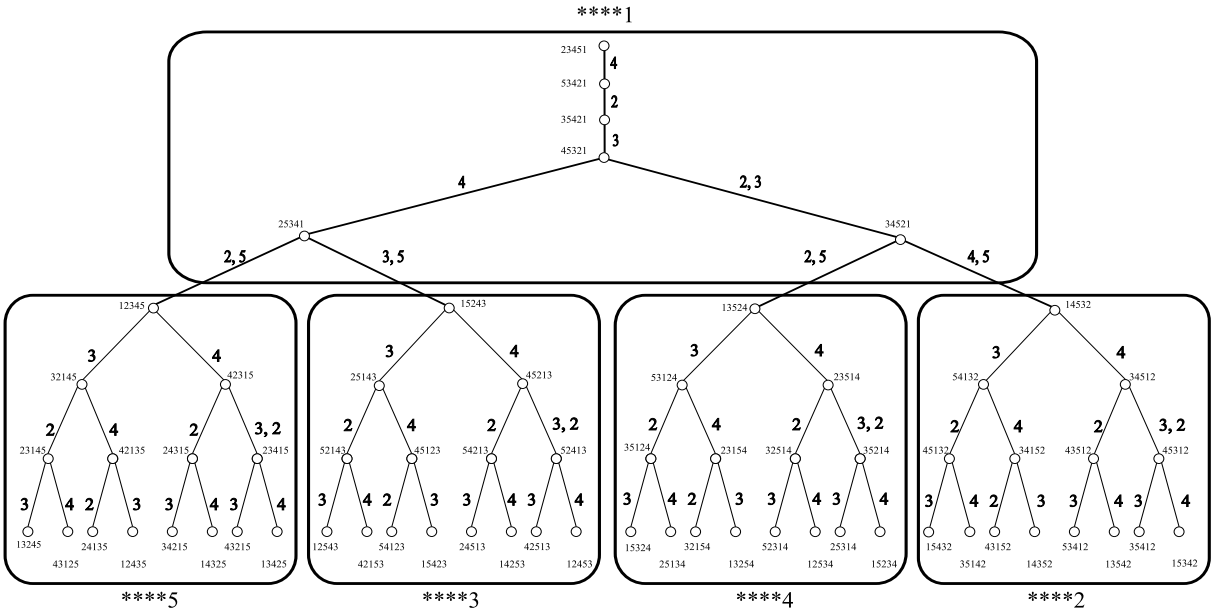

Figure 3: An embedding of a threaded tree $TT_{\alpha(5),\rho(5)} = T_{3,6}$ in S_5.

n is a power of 2, in which case all S_{n-1}'s will be used out by components that are threaded trees; this case is more difficult as we are "squeezing" more links and nodes into one S_{n-1}.

Below, we develop Step 1 and Step 2. We separate the discussion into two cases, depending on the value of n.

3.1 When n Is Not a Power of 2

We will first construct a threaded tree whose thread is of length 1. Then we will extend the length of the thread and attach binary trees to its leaves. Intuitively, the thread of the threaded tree will occupy one S_{n-1}, and so does each of the attached binary trees.

3.1.1 Step 1: Construct $\hat{T}$

When n is non-power-of-2, we will actually establish a stronger result: an embedding of $\hat{T} = TT_{n-1,\lfloor \log n \rfloor + 1}$ (the thread is longer). This turns out to be helpful for the embedding when n is a power of 2.

We first establish an embedding of $\tilde{T} = TT_{1,\lfloor \log n \rfloor + 1}$. Later we will extend the length of the thread. This is done by constructing some logical trees as shown below in which each edge is labeled by one or two dimensions. This will imply an embedding of $\tilde{T}$ as desired.

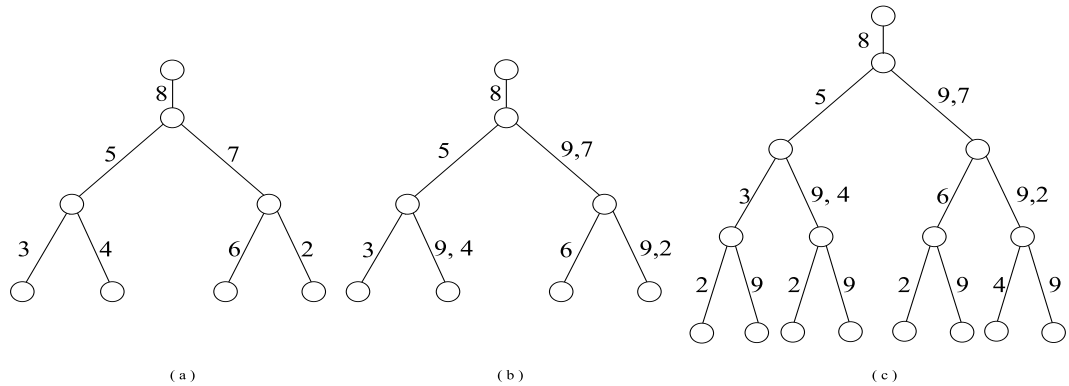

Figure 4: An example of constructing logical trees in S_9: (a) L_1, (b) L_2, and (c) L_3.

A1: Construct a logical tree $L_1 = TT_{1,\lfloor \log n \rfloor}$. On L_1, we arbitrarily label each edge by a distinct integer from 2 to $n-1$. (Note that this is always possible because when n is non-power-of-2, there are $2^{\lfloor \log n \rfloor} - 1 + 1 = 2^{\lfloor \log n \rfloor} \leq n - 1$ nodes, and thus $\leq n - 2$ edges, in L_1.) For instance, when $n = 9$, $L_1 = TT_{1,3}$; a labeling is shown in Fig. 4(a).

A2: Let logical tree L_2 be L_1 with the following modification. For each non-thread link in L_2 which is a *right link* (i.e., a link to a right child), suppose the label of this link is i; insert in front of i the number n. (See the example in Fig. 4(b).)

A3: Let logical tree L_3 be L_2 with the following modification. For each leaf node x of L_3, attach two child nodes to x. Label the link to x's right child by n. Label the link to x's left child by any integer k such that (i) $2 \leq k \leq n-1$, and (ii) on the path from x upward the tree until a link of a label n is met, no link in-between should have a label of k. Note that (ii) is possible as there are always more choices than the length of the path. (See the example in Fig. 4(c).)

We then let the logical tree L_3 represent an embedding of $\tilde{T}$ in S_n, by assigning any node in S_n as the root and traversing the labels (dimensions) indicated in L_3 to find all other tree nodes. Also, from Fig. 4, we see that the embedding dilation still does not exceed 2. Below we prove some important properties provided by such an embedding.

Lemma 1 *If we remove all edges on dimension n from $\tilde{T}$, then $\tilde{T}$ will be partitioned into $2^{\lfloor \log n \rfloor}$ linear paths, each located in a distinct $(n-1)$-substar $*^{n-1}i$, $1 \leq i \leq n$.*

Proof. Links along dimension n must connect two $(n-1)$-substars $*^{n-1}i$ and $*^{n-1}j$. Observe that in L_3, each right link has a label n, so there are totally $2^{\lfloor \log n \rfloor} - 1$

links along dimension n in $\tilde{T}$. Consequently, removing these edges will partition $\tilde{T}$ into $2^{\lfloor \log n \rfloor}$ segments. Furthermore, these segments are all linear paths as the cutting points consistently occur at every right link on L_3.

It should also be clear that each linear path must fall in an $(n-1)$-substar $*^{n-1}i$. It remains to prove that these $(n-1)$-substars are all distinct. Without loss of generality, let $x = x_1 x_2 \cdots x_n$ be the parent node of the leftmost leaf in $\tilde{T}$. Then the linear path where x resides falls in substar $*^{n-1}x_n$. The linear path where x's right child resides falls in substar $*^{n-1}x_1$.

The location of any other linear path, say, P_1 can be reasoned as follows. Let y be the uppermost node (i.e., at the highest level) in P_1. Consider the path from y to node x in $\tilde{T}$, denoted as $P_2 = y \rightarrow y' \rightarrow y'' \rightarrow \cdots \rightarrow x$. The first link $y \rightarrow y'$ must be along dimension n. Suppose the second link $y' \rightarrow y''$ is along dimension k (note that $k \neq 1$ and $k \neq n$). According to the construction of L_1, no other link in P_2 will be along dimension k. Thus, the first symbol of y' must be x_k, implying that node y (and thus P_1) will fall in substar $*^{n-1}x_k$. Furthermore, by **A1**, each linear path P_1 will correspond to a different value of k, and thus fall in a distinct substar $*^{n-1}x_k$. Hence the lemma. □

An example of the above lemma is shown in Fig. 5(a), which is obtained from the example in Fig. 4 by extending L_3 into a real embedding of $\tilde{T}$. After removing all links along dimension 9, $\tilde{T}$ is partitioned into $2^{\lfloor \log 9 \rfloor} = 8$ linear paths. The node x identified in the above proof is also shown in the figure. As can be seen, the path where $x = 123456789$ resides falls in substar $*^8 9$. The right child of x falls in substar $*^8 1$. Consider the linear path $P_1 = 521436987 \xrightarrow{2} 251436987 \xrightarrow{4} 451236987$, where the numbers on top of arrows indicate the link dimensions. The corresponding P_2 will be $521436987 \xrightarrow{9} 721436985 \xrightarrow{7} 921436785 \rightarrow \cdots \rightarrow x$. Indeed, P_1 falls in substar $*^8 7$, where 7 comes from the dimension of the second link on P_2.

Lemma 2 *On each linear path obtained from Lemma 1, no two links are along the same dimension.*

Proof. This is guaranteed by the assignment rules **A1** and **A3**. □

Lemma 3 *The embedding of $\tilde{T}$ in S_n is congestion-free, dilation-2, and load-1.*

Proof. According to Lemma 1, all linear paths fall in distinct $(n-1)$-substars, so it suffices to prove that each linear path incurs no congestion, dilation 2, and load 1. This

directly follows from Lemma 2. □

Finally, we will extend the length of the thread of $\tilde{T}$ to construct $\hat{T}$.

A4: Let k be the dimension of the only thread link on $\tilde{T}$ (according to **A1**, $2 \leq k \leq n-1$). We traverse a linear path of length $n-2$ as follows to construct $\hat{T}$: (root of $\tilde{T}$) $\stackrel{n}{\rightarrow} t_1 \stackrel{2}{\rightarrow} t_2 \stackrel{3}{\rightarrow} \cdots \stackrel{k-1}{\rightarrow} t_{k-1} \stackrel{k+1}{\rightarrow} t_k \stackrel{k+2}{\rightarrow} \cdots \stackrel{n-1}{\rightarrow} t_{n-2}$.

Lemma 4 *$\hat{T}$ is a congestion-free, dilation-2 embedding of threaded tree $TT_{n-1,\lfloor \log n \rfloor+1}$ in an S_n such that all thread links are along distinct dimensions, where n is not a power of 2.*

Proof. Clearly, the thread follows distinct dimensions and thus does form a linear path. With the result in Lemma 3, it suffices to prove that the newly added thread $t_1 \rightarrow \cdots \rightarrow t_{n-2}$ incurs no extra congestion, dilation, and load.

We will use the same proof technique as in Lemma 1 here. As no link in $t_1 \rightarrow \cdots \rightarrow t_{n-2}$ is along dimension n, the path falls in an $(n-1)$-substar. Consider the thread link at the lowest level of $\hat{T}$ (called e_1, which is along dimension k), and that at the second lowest (called e_2, which is along dimension n). Recall the node $x = x_1 x_2 \cdots x_n$ in the proof of Lemma 1 (i.e., the parent of the leftmost child in $\hat{T}$). By **A1**, along the path from the root of $\tilde{T}$ to x, e_1 is the only link along dimension k. So the first symbol of the root of $\tilde{T}$ is x_k. Since e_2 is along dimension n, it will lead its end node t_1 (and thus the whole newly added thread $t_1, t_2, \ldots, t_{n-2}$) to substar $*^{n-1}x_k$, which is distinct from the $(n-1)$-substars resident by other linear paths. So the lemma is proved. □

For instance, we show in Fig. 5(b) the $\hat{T}$ extended from the $\tilde{T}$ in Fig. 5(a). The newly added thread $921436758 \rightarrow \cdots \rightarrow 792143658$ falls in substar $*^8 8$, where the label 8 comes from the dimension of the lowest thread link. This substar is distinct from those resident by other linear paths.

3.1.2 Step 2: Attach a $T_{\rho(n-1)}$ to Each Leaf of $\hat{T}$

We already establish an embedding of $\hat{T}$, which can be partitioned into $2^{\lfloor \log n \rfloor}$ linear paths, each falling in a distinct $(n-1)$-substar and having an end node as a leaf. However, arbitrarily attaching a binary tree $T_{\rho(n-1)}$ at the end node inside the corresponding $(n-1)$-substar may not work as the binary tree and the linear path may conflict and thus collide on common nodes or links. In the following, we will use the induction hypothesis to resolve this problem.

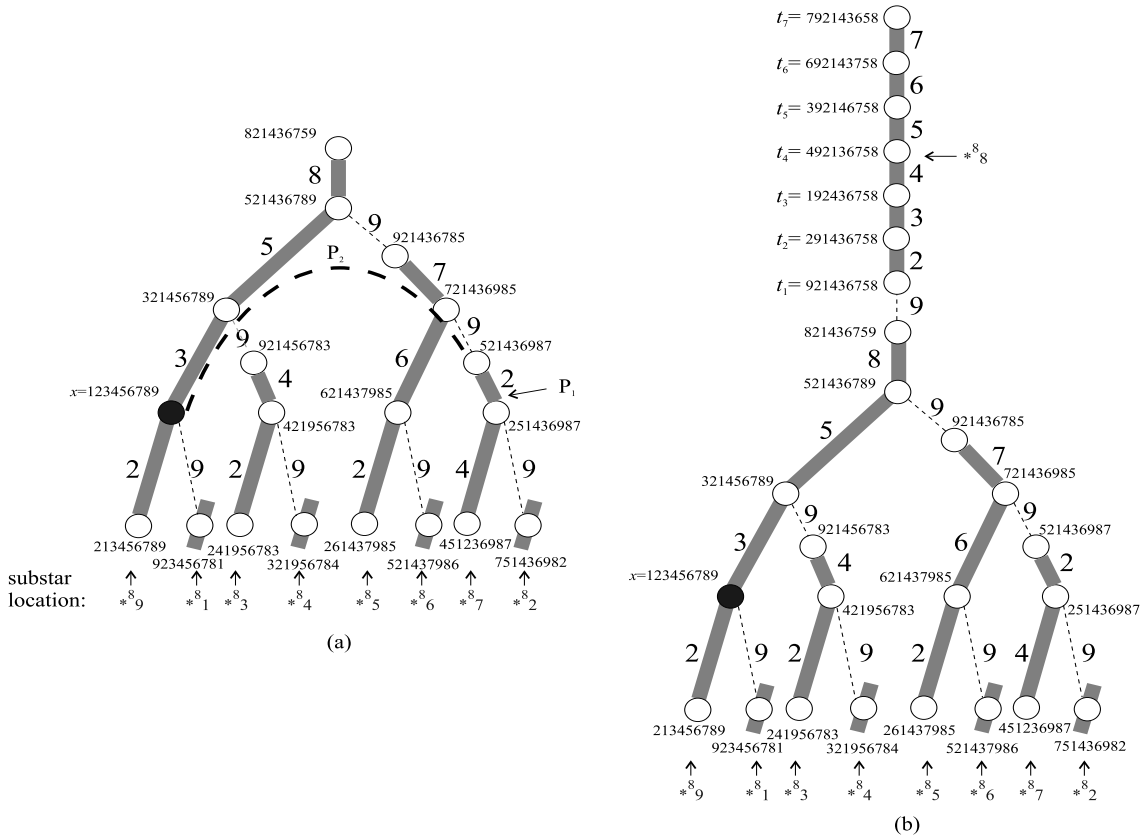

Figure 5: Illustration of (a) Lemma 1, the construction of tree $\tilde{T}$, and (b) Lemma 4, the extension of $\tilde{T}$ to $\hat{T}$. Note that n is non-power-of-2.

Consider any linear path (say, of length l) obtained by Lemma 1. Attaching a $T_{\rho(n-1)}$ at the bottom of the linear path in fact forms a threaded tree $TT_{l,\rho(n-1)}$. Lemma 2 already states that dimensions of the thread links are all distinct. The path length satisfies $l \leq \lfloor \log n \rfloor + 1 = \alpha(n-1)$ (i.e., height of $\tilde{T}$). Hence the induction hypothesis is applicable.

Lemma 5 *When n is not a power of 2, there always exists a congestion-free, dilation-2, load-1 embedding of $TT_{n-1,\rho(n)}$ in S_n such that all thread links are along distinct dimensions.*

3.2 When n Is a Power of 2

Recall the above embedding. We mainly partition S_n into $2^{\lfloor \log n \rfloor}$ $(n-1)$-substars and embed a tree $T_{\rho(n-1)}$ in each of these substars. The difficulty in the case when n is a power of 2 is: all $(n-1)$-substars are used out and thus no free $(n-1)$-substar remains for use to extend $\tilde{T}$ to $\hat{T}$. Thus, special treatment is necessary. The extension from $\tilde{T}$ to $\hat{T}$ will be done in the S_{n-1} where the leftmost $T_{\rho(n-1)}$ is located. In the following, we discuss how to modify Steps 1 and 2 to accommodate to this change. We first develop the case of $n = 2^d, d \geq 4$. The special case of $n = 2^3$ will be discussed later.

3.2.1 When $n = 2^d$, $d \geq 4$

In Step 1, we use the following revised **A1 − A3** to construct a $\hat{T} = TT_{\alpha(n),\log n+1}$. The step **A4** is unnecessary.

A1': Construct a logical threaded tree $L_1 = TT_{\alpha(n),\log n}$. On L_1, label each edge with the following rules:

- **i)** Label each edge belonging to the binary tree by a distinct number from 2 to $n-1$. (This is possible as there are totally $2^{\log n} - 2 = n - 2$ links. Thus each dimension will be used once.)
- **ii)** Label each thread link by a distinct number from 2 to $n-1$. Further, no thread link should use a dimension equal to that of any link along the path from the root of the binary tree to the leftmost leaf. (Note that this is always possible as the thread and the path contain totally

$$\alpha(n) + \log n - 1 = \lfloor \log(n+1) \rfloor + 1 + \log n - 1 = 2d \leq n - 2 \qquad (1)$$

links.)

A2': Same as **A2**, insert n to labels of all right links to construct L_2.

A3': Same as **A3**, attach two children to each leaf of L_2 to construct L_3. (Note that this requires one more distinct label for the newly added left links. There are $n-2$ dimensions available and the height of L_3 satisfies

$$\alpha(n) + \log n - 1 + 1 = 2d + 1 \leq n - 2 \tag{2}$$

when $d \geq 4$.)

An example when $n = 2^4$ is shown in Fig. 6(a) to illustrate this process. As compared to the case of n being non-power-of-2 (such as Fig. 5(b)), the main difference is that the dimension of the second thread link from the bottom is not n any more. Now let's use L_3 as an embedding of $\hat{T} = TT_{\alpha(n),\log n+1}$. A property similar to Lemma 1 will hold: if we remove all links along dimension n in $\hat{T}$, $\hat{T}$ will be partitioned into $2^{\log n} = n$ linear paths, each resident in a distinct $(n-1)$-substar. However, the linear path containing the thread now becomes longer.

In Step 2, we will still attach a $T_{\rho(n-1)}$ on each leaf of $\hat{T}$. Since $\hat{T}$ can be partitioned into disjoint linear paths, the attachment is an embedding of a threaded tree $TT_{l,\rho(n-1)}$ in an S_{n-1}, where l is the length of the linear path. By **A1'** and **A3'**, links on the thread are all along distinct dimensions. As $n-1$ is not a power of 2, by Lemma 4, an embedding of a threaded tree $TT_{n-2,\rho(n-1)}$ in which all thread links are along distinct dimensions exists in an S_{n-1}. By Eq. (2), $l \leq n-2$, so the attachment is feasible.

3.2.2 When $n = 2^d, d = 3$

Step 1 is the same as the case of $d \geq 4$, except **A3'**. The difficulty is that Eq. (2) does not hold true any more when $d = 3$, making impossible labeling the newly added left links of the *leftmost* leaf. So we revise **A3'** as follows:

A3": Same as **A3'**, but label the newly added left link of the leftmost leaf node by the label of the topmost thread link of L_3.

For instance, Fig. 6(b) shows an example L_3 when $n = 8$; the leftmost link at the bottom is labeled 4, which is the same as the label of the topmost thread link. Now suppose we embed $\hat{T}$ using L_3. It should be clear that the similar partitioning of $\hat{T}$ along dimension n into linear paths is still possible. So Step 2 is feasible, following the induction hypothesis, except the linear path containing the thread (see the linear

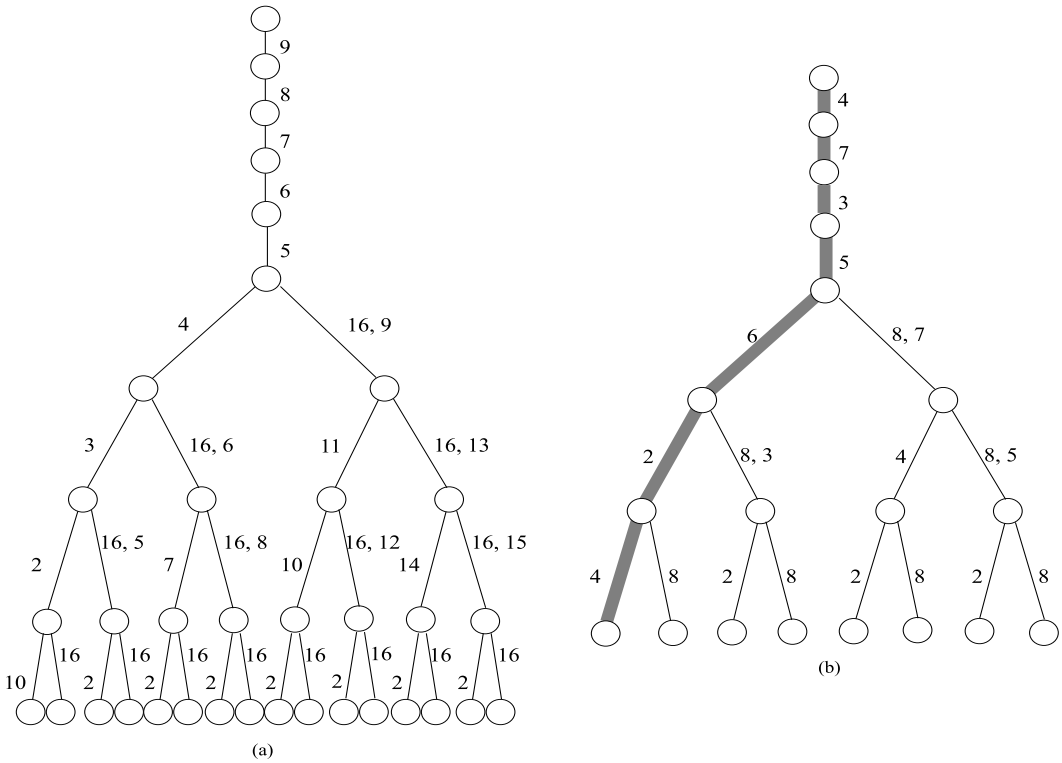

Figure 6: An example of constructing logical tree L_3 when n is a power of 2: (a) $n = 2^4$ and (b) $n = 2^3$.

path shown in bold in Fig. 6(b), which has the same dimension at the first and last links; recall that Lemma 4 does not support such an embedding). For the linear path containing the thread, we need an embedding of $TT_{7,\rho(7)}$ in an S_7 such that both the first and last links on the thread are along the same dimension (the length of the thread is from Eq. (2)). By Lemma 4, we already have an embedding of $TT_{6,\rho(7)}$ in which thread links are all along distinct dimensions. To achieve our goal, simply add one more link along dimension i on top of the thread of $TT_{6,\rho(7)}$ offered by Lemma 4, where i is the dimension of thread link at the lowest level in $TT_{6,\rho(7)}$. Such an extension does not incur extra congestion or load (instead of giving a formal proof, we sketch such an embedding in an S_7 in Fig. 7 to support this argument; the newly added link, together with some other thread links, are within a separate substar S_6). So Step 2 is solved.

Lemma 6 *When n is a power of 2, there exists a congestion-free, dilation-2, load-1 embedding of $TT_{\alpha(n),\rho(n)}$ in S_n such that all thread links are along distinct dimensions.*

Lemma 5 and Lemma 6 together conclude our induction step.

Theorem 1 *A complete binary tree $T_{\rho(n)}$ can be embedded in $S_n, n \geq 3$, with no congestion, dilation 2, and load 1.*

4. Embedding Larger Complete Binary Trees

We have shown how to embed a $T_{\rho(n)}$ into S_n. To embed a larger binary tree $T_{\rho(n)+k}$ into S_n for any integer k, one straight-forward approach is to regard $T_{\rho(n)+k}$ as a $T_{\rho(n)}$ in which each leaf stands for a tree T_{k+1}. This gives an embedding without increasing congestion and dilation, but with higher load of $2^{k+1} - 1$. In the following, we present an approach which reduces the load by about a half. Intuitively, we will "fold" the nodes near the bottom of the tree to their ancestor nodes.

First, we will show how to embed a $T_{\rho(n)+1}$ into S_n. In Fig. 8, we show a congestion-free, dilation-2, load-2 embedding of a $T_{\rho(4)+1} = T_5$ in S_4. The tree nodes from level 2 to level 5 are embedded similarly[2] as in Fig. 2. Such an embedding possesses two properties:

P1. For each node at level 2, its left child is embedded in the same node as itself, and its right child in a new, unused node.

[2]The only modification is that the left child of the leftmost node on level 2 (i.e., node 2314) is changed to 3124.

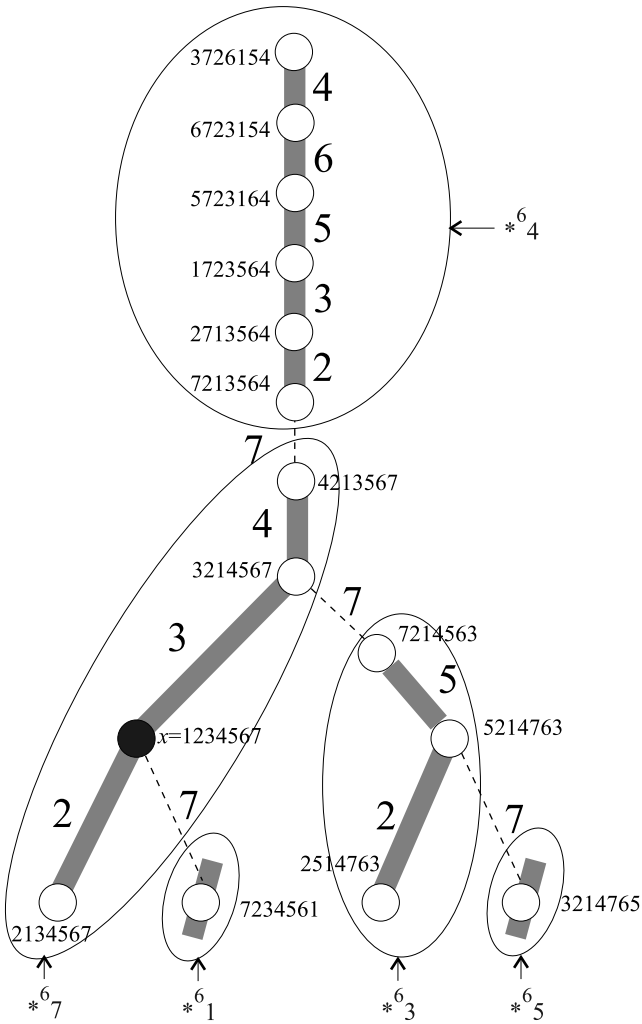

Figure 7: An embedding of a $TT_{7,\rho(7)}$ in an S_7 such that the first and last links on the thread are along the same dimension. Excluding the first thread link, the embedding is supported by Lemma 4.

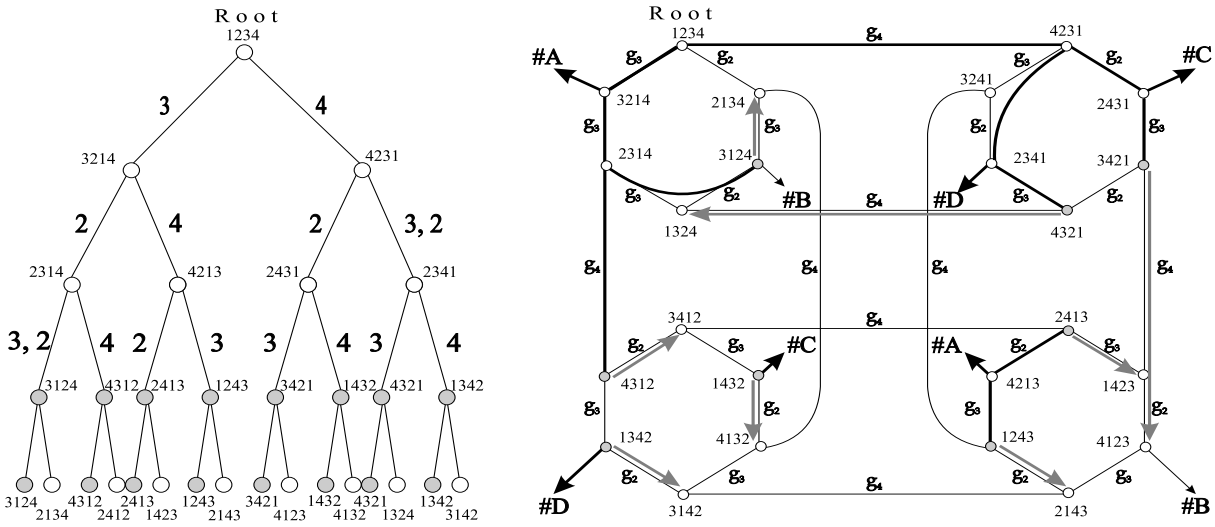

Figure 8: A congestion-free, dilation-2, load-2 embedding of a $T_{\rho(4)+1} = T_5$ in S_4.

P2. All leaf nodes are embedded in distinct nodes.

These properties can be proved by examination. **P2** is especially important for our result to extend to embedding larger binary trees.

A congestion-free, dilation-2, load-2 embedding of a threaded tree $TT_{\alpha(5),\rho(5)+1} = TT_{3,7}$ in an S_5 can be easily obtained as follows. Observe that in the embedding in Fig. 3, each of four 4-substars $*^4 5$, $*^4 3$, $*^4 4$, and $*^4 2$ is embedded with a T_4. If we replace the embedding by a T_5 inside each 4-substar (using Fig. 8), our goal is achieved. It is not hard to see that in the development of Section 3, if we replace the embedding of each $TT_{\alpha(5),\rho(5)}$ in an S_5 by $TT_{\alpha(5),\rho(5)+1}$, then the level of the binary tree embeddable is increased by one.

Lemma 7 *A complete binary tree $T_{\rho(n)+1}$ can be embedded in an S_n with no congestion, dilation-2, and load-2. Furthermore, each leaf node is embedded in a distinct node.*

This leads to the following result.

Theorem 2 *A complete binary tree $T_{\rho(n)+k}$ can be embedded in an S_n with no congestion, dilation-2, and load-2^k, where k is any positive integer.*

Proof. When $k = 1$, this reduces to Lemma 7. When $k \geq 2$, we first use the embedding in Lemma 7 to embed a $T_{\rho(n)+1}$, and regard each leaf of such embedding as a tree T_k.

This forms an embedding of $T_{\rho(n)+k}$. The load is the original load, 2, plus the number of newly added nodes, $2^k - 2$. □

5. Conclusions

We have presented two schemes for embedding a complete binary tree into a star graph. The first scheme successfully constructs a complete binary tree of size comparable or superior to existing results, while at the same time using less edge congestion and dilation; a detailed comparison is in Table 1. The second scheme allows embedding a complete binary tree of any size into a star graph of a fixed dimension; this would provide great flexibility in designing parallel algorithms. Open problems include how to reduce the dilation of our first scheme, how to increase the embeddable tree size of our first scheme, and how to reduce the load of our second scheme. It will also be interesting to see whether our inductive and splitting approaches can be applied to the embedding of binary trees on other Cayley graph families.

Acknowledgments

The authors would like to thank the referees for their helpful comments to the earlier version of this paper.

References

[1] S. B. Akers, D. Harel, and B. Krishnameurthy. The star graph: An attractive alternative to the n-cube. In *Proceedings of International Conference on Parallel Processing*, pages 393–400, Aug. 1987.

[2] S. B. Akers and B. Krishnameurthy. Group-theoretic model for symmetric interconnection networks. *IEEE Transactions on Computers*, 38(4):555–565, Apr. 1989.

[3] N. Bagherzadeh, M. Dowd, and N. Nassif. Embedding an arbitrary binary tree into star graph. *IEEE Transactions on Computers*, 45(4):475–481, Apr. 1996.

[4] N. Bagherzadeh, N. Nassif, and S. Latifi. A routing and broadcasting scheme on faulty star graphs. *IEEE Transactions on Computers*, 42(11):1398–1403, Nov. 1993.

[5] A. Bouabdallah, M. C. Heydemann, J. Opatrny, and D. Sotteau. Embedding complete binary tress into star graphs. In *Proceedings of Conference on the Mathematical Foundation of Computer Science, Lecture Notes in Computer Science, 841*, pages 266–275, 1994.

[6] A. Bouabdallah, M. C. Heydemann, J. Opatrny, and D. Sotteau. Embedding complete binary tress into star and pancake graphs. *Theory of Computing Systems*, 31(3):279–305, May/June 1998.

[7] Y.-S. Chen and J.-P. Sheu. A fault-tolerant reconfiguration scheme in the faulty star graph. In *Proceedings of 1996 Second International Conference on Algorithms & Architectures for Parallel Processing*, pages 241–248, 1996.

[8] Y.-S. Chen, J.-P. Sheu, and Y.-C. Tseng. Toward optimal ring embedding in the faulty star graph. In *Proceedings of 1998 Fourth International Conference on Parallel and Distributed Processing Techniques and Applications*, July 1998.

[9] K. Day and A. Tripathi. A comparative study of topological properties of hypercubes and star graphs. *IEEE Transactions on Parallel and Distributed Systems*, 5(1):31–38, Jan. 1994.

[10] P. Fragopoulou and S. G. Akl. Optimal communication algorithms on star graphs using spanning tree constructions. *Journal of Parallel and Distributed Computing*, 24:55–71, 1995.

[11] M.-C. Heydemann, J. Opatrny, and D. Sotteau. Embeddings of complete binary trees into star graphs with congestion 1. In *Proceedings of the 28th Annual Hawaii International Conference on System Sciences*, pages 546–554, 1995.

[12] C.-C. Hsu. All-fault-tolerant embedding of a complete binary tree in a group of cayley graphs. *IEE Proc.-Comput. Digit. Tech.*, 143(2):156–160, 1996.

[13] Z. Jovanović and J. Mišić. Fault tolerance of the star graph interconnection network. *Information Processing Letters*, 49:145–150, 1994.

[14] J.-S. Jwo, S. Lakshmivarahan, and S. K. Dhall. Embeddings of cycles and grids in star graphs. In *Proceedings of Symposium on Parallel and Distributed Processing*, pages 540–547, 1990.

[15] S. Latifi. On the fault-diameter of the star graph. *Information Processing Letters*, 46:143–150, 1993.

[16] J. Lee and J. Chang. Embedding complete binary tress in star graphs. *J. Korea Inf. Sci. Soc.*, 21(2):407–415, 1994.

[17] F. T. Leighton. *Introduction to Parallel Algorithms and Architectures: Arrays-Trees-Hypercubes*. Morgan Kaufmann Publishers, San Mateo, California, 1992.

[18] V. E. Mendia and D. Sarkar. Optimal broadcasting on the star graph. *IEEE Transactions on Parallel and Distributed Systems*, 3(4):389–396, July 1992.

[19] J. Mišić and Z. Jovanović. Communication aspects of the star graph interconnection network. *IEEE Transactions on Parallel and Distributed Systems*, 5(7):678–687, July 1994.

[20] B. Monien and H. Sudborough. Embedding one interconnection network in another. *Computing Supplement*, 7:257–282, 1990.

[21] M. Nigam, S. Sahni, and B. Krishnamurthy. Embedding hamiltonians and hypercubes in star interconnection network. In *Proceedings of International Conference on Parallel Processing*, volume III, pages 340–343, Aug. 1990.

[22] K. Qiu. Broadcasting on the star and pancake interconnection networks. In *Proceedings of International Parallel Processing Symposium*, pages 660–665, 1995.

[23] K. Qiu, S. G. Akl, and H. Meijer. On some properties and algorithms for the star and pancake interconnection networks. *Journal of Parallel and Distributed Computing*, 22:16–25, 1994.

[24] J.-P. Sheu, C.-T. Wu, and T.-S. Chen. An optimal broadcasting algorithm without message redundancy in star graphs. *IEEE Transactions on Parallel and Distributed Systems*, 6(6):653–658, 1995.

[25] Y.-C. Tseng, S.-H. Chang, and J.-P. Sheu. Fault tolerant ring embedding in star graphs. *IEEE Transactions on Parallel and Distributed Systems*, 8(12):1185–1195, Dec. 1997.