PAPER

Efficient Broadcasting in an Arrangement Graph Using Multiple Spanning Trees*

Yuh-Shyan CHEN†, Tong-Ying JUANG††, *and* En-Huai TSENG†, *Nonmembers*

SUMMARY The arrangement graph $A_{n,k}$ is not only a generalization of star graph ($n-k=1$), but also more flexible. Designing efficient broadcasting algorithm on a regular interconnection network is a fundamental issue for the parallel processing techniques. Two contributions are proposed in this paper. Initially, we elucidate a first result to construct $n-k$ edge-disjoint spanning trees in an $A_{n,k}$. Second, we present efficient (one/all)-to-all broadcasting algorithms by using constructed $n-k$ spanning trees, where height of each spanning tree is $2k-1$. The arrangement graph is assumed to use one-port and all-port communication models and packet-switching (or store-and-forward) technique. Using $n-k$ spanning trees allows us to present efficient broadcasting algorithm in the arrangement graphs and outperforms previous results. This is justified by our performance analysis.

key words: *arrangement graph, broadcast, interconnection network, parallel processing, routing*

1. Introduction

Designing large multi-processor systems frequently involves organizing into various configurations. One of the widely studied interconnection network topologies is the star graph [9], [11], [12]. As a member of the Cayley graphs, the star graph possesses several attractive features such as its diameter-to-node-degree ratio, scalability, partitionability, symmetry, and high degree of fault tolerance [1], [3]. However, the star graph is limited with respect to its number of nodes: $n!$ for an n-dimensional star graph.

A new interconnection topology, *arrangement graph*, has recently been proposed [4]. As a family of undirected graphs that contains the star graph family, the arrangement graph has desired properties, such as symmetric vertex and symmetric edge, strongly resilience and maximally fault-tolerance. Arrangement graph is more flexible than the star graph in terms of choosing major design parameters, i.e. member of vertices, degree and diameter, while preserving most of the excellent properties of the star graph.

Manuscript received April 19, 1999.

Manuscript revised August 16, 1999.

†The authors are with the Department of Computer Science and Information Engineering, Chung-Hua University, Hsin-Chu, 30067, Taiwan, Republic of China.

††The author is with the Department of Statistics, National Chung Hsing University, Taipei, Taiwan, Republic of China.

*This paper was presented at ICPADS '98 conference, and this work was supported by the National Science Council, R.O.C., under Contract NSC88-2213-E-216-011.

The arrangement graph has received considerable attention in [2], [4]–[8], [10]. Firstly, Day and Tripathi [5] designed a shortest-path routing algorithm for the arrangement graphs. According to their results, the arrangement graph can be embedded cycles whose length ranging from three to the size of the graph [6]. Moreover, the arrangement graph can be decomposed into vertex disjoint cycles in many different ways [6]. Furthermore, multi-dimensional grads, hypercube and one spanning tree can be embedded in arrangement graph [7]. The spanning tree can support broadcasting communication in the arrangement graph. Tsai and Horng proposed an efficient scheme to embed hypercube on arrangement graphs [10]. Heieh and Chen [8] further demonstrated that the arrangement graph remains a ring even if it is faulty. Bat et al. [2] recently proposed a distributed fault-tolerant algorithm for one-to-all broadcasting only in the one-port communication model on the arrangement graph.

In light of above discussion, this work elucidates the broadcast problem on arrangement graphs in *one-port* and *all-port* communication model. In this work, the network adopts the packet-switching or store-and-forward technique. By following the formulation of many works (e.g. [13]), assume the existence of two kinds of cost: *start-up time* and *transmission time*. In particular, a packet of b bytes is sent along a link takes T_s+bT_c time, where T_s denotes the time to initialize (or start-up) the communication link and T_c represents the latency to transmit a byte. An attempt to minimize either the start-up time cost and the transmission cost is the underlying motivation of this work. Typically, the start-up time is significant in current machines, while the transmission time should not be neglect when the packet is long.

Note that Bat et al. [2] proposed broadcast algorithm only optimize the start-up time, but not consider the transmission time. In contrast to earlier solution in arrangement graph [2], [7], this work aims to minimize start-up time and transmission costs simultaneously. The major approach used herein is to construct multiple spanning trees. For the broadcasting algorithm, we propose a new scheme to construct multiple spanning trees in an arrangement graph $A_{n,k}$ that has a desired property that $n-k$ copies of such trees can be embedded simultaneously in the network without edge-congestion. By concurrently transmitting data along

these trees in a pipelined manner, our results are an improvement over the schemes of deriving one spanning tree in [7] by order $O(n-k)$ in the transmission time. In this investigation, we assume that a node consists of a processor with bidirectional communication links to each of its adjacent nodes. Therefore, the term *edge-disjoint spanning trees* can be interchangeably adopted to reflect that no two edges of our spanning trees share a same direct communication link.

To our knowledge, this work reports for the first time on feasibility of embedding multiple $O(n-k)$ spanning trees in an $A_{n,k}$ while, at the same time, keeping the edge-congestion free. Similar results for the star graph can be found in [13]. A broadcast algorithm is proposed herein by the subgraph-partitioning scheme. Under the all-port model, the proposed one-to-all broadcast algorithm can be executed in $A_{n,k}$ in time $(\sqrt{(2k-1)T_s}+\sqrt{\frac{mT_c}{n-k}})^2$, where m denotes the size of the broadcast message. Under the one-port model, the one-to-all broadcast algorithm can be implemented in $A_{n,k}$ in time $O(k(n-k)(2k-1)T_s+kmT_c)$. Under the all-port model, the all-to-all broadcast algorithm can be implemented in $A_{n,k}$ in time $O(2k\times T_s+\frac{mn!}{(n-k)(n-k)!}T_c)$. Under the one-port model, the all-to-all broadcast algorithm can be performed in $A_{n,k}$ in time $O(2k^2(n-k)T_s+\frac{kmn!}{(n-k)!}T_c)$.

The rest of this paper is organized as follows. Section 2 introduces preliminaries. Section 3 presents how to construct multiple spanning trees. Section 4 compares our broadcasting results with those of other related works. Conclusions are finally drawn in Sect. 5.

2. Preliminaries

The arrangement graph is denoted by $A_{n,k}$, where specified by integers n and k and $1 \le k \le n-1$. Denote $\langle n\rangle = \{1,2,\ldots,n\}$. Let $P\binom{n}{k}$ be the set of permutations of k symbols taken from $\langle n\rangle$. These k symbols are denoted as $X = x_1x_2\cdots x_k$. Refer x_i as the i'th element of X. The (n,k)-*arrangement graph*, denoted as $A_{n,k}$, defined in [5] is an undirected graph (V,E) as follows.

$$\begin{cases} V = \{X = x_1x_2\cdots x_k | x_i \text{ in } \langle n\rangle \text{ and } x_i \ne x_j \text{ for } i \ne j\} = P\binom{n}{k}\}, \\ E = \{(x,y) | x \text{ and } y \text{ in } V \text{ and for some } i \text{ in } \langle k\rangle \\ \qquad x_i \ne y_j \text{ and } x_j = y_j \text{ for } j \ne i\}. \end{cases}$$

Figure 1 depicts an example of $A_{4,2}$ and $A_{5,3}$. The edge of $A_{n,k}$ connecting neighboring nodes which differ in exactly one of their k positions. The vertices of $A_{n,k}$ are the arrangements of k elements of $\langle n\rangle$. For example, in $A_{4,2}$, the node $p = 41$ is connected to the nodes = 42, 43, 21 and 31. An edge of $A_{n,k}$ connecting two arrangements p and q which differ only in position i, is called as an i-*edge*. For all values of n and k, $A_{n,k}$ is a regular graph on $\frac{n!}{(n-k)!}$ nodes that is regular of degree $k(n-k)$, and a diameter $\lfloor \frac{3}{2}k \rfloor$ [5]. For an arrangement $X = x_1x_2\ldots x_k$, we define $EXT(X) = \langle n\rangle - \{x_1,x_2,\ldots,x_k\}$ to be the $n-k$ elements of $\langle n\rangle$ not appearing in the arrangement X. Let $INT(X) = \{x_1,x_2,\ldots,x_k\} = \langle n\rangle - EXT(X)$ to be the k elements of $<n>$ appearing in the arrangement X. For example, we consider the node $p = 412$ in the arrangement graph $A_{5,3}$, so $EXT(p) = \{3,5\}$ and $INT(p) = \{1,2,4\}$.

In an $A_{n,k}$, each node p performs an adjacent function $ADJ_{x,y}(p)$ to arrive at adjacent node q, where x is the position of label of node and y is the changed label in $EXT(p)$. Given $X = x_1\cdots x_x\cdots x_k \in INT(X)$, the adjacent function $ADJ_{x,y}(X)$ is the adjacent nodes of X obtained by changing x_x in X to be y, where $1 \le x \le k$ and $y \in EXT(X)$. Consider an $A_{5,3}$, adjacent nodes for node $p = 412$ are 413, 415, 432, 452, 312 and 512, where $EXT(p) = \{3,5\}$, $INT(p) = \{1,2,4\}$, $ADJ_{1,3}(p) = 312$, $ADJ_{1,5}(p) = 512$, $ADJ_{2,3}(p) = 432$, $ADJ_{2,5}(p) = 452$, $ADJ_{3,3}(p) = 413$, and $ADJ_{3,5}(p) = 415$.

The $A_{n,k}$ is with recursive structure [5]. That is, an $A_{n,k}$ can be partitioned into n copies of $A_{n-1,k-1}$ each embedded $A_{n-1,k-1}$ is conveniently denoted by $<*^{k-1}\alpha>_{n,k}$, where $\alpha \in \{1,2,\cdots,n\}$, (where $*$ represents a "don't care" symbol). For example, $<**3>_{4,3}$ represents an embedded $A_{3,2}$ of $A_{4,3}$, contains six nodes: 123, 143, 213, 243, 413, and 423. From other point of view, there are n copies of $<*^{k-1}\alpha>_{n,k}$ which are obtained by performing a splitting operation on $<*^k>_{n,k}$, where $\alpha \in \{1,2,\ldots,n\}$. This operation is called as k-partition. Generally, $A_{n,k}$ can be partitioned into $\frac{n!}{(n-p)!}$ node-disjoint copies of $A_{n-p,k-p}$ in $\frac{n!}{p!(n-p)!}$ different ways and that in total $A_{n.k}$ contains $\binom{k}{p}\frac{n!}{(n-p)!}$ copies of $A_{n-p,k-p}$, for $1 \le p \le k-1$.

In this paper, we adopt the packet-switching (or store-and-forward) model and, thus, the latency to transmit a packet of b byte along a link is $T_s + bT_c$, where T_s denotes the time to initialize the communication link and T_c represents the time to transmit a byte. Each (undirected) edge is regarded as bidirectional communication links. Two communication models are also 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 a packet along all $k(n-k)$ ports.

Lemma 1: A lower bound for one-to-all broadcasting in a store-and-forward $A_{n,k}$ is $\max\left\{\lfloor\frac{3}{2}k\rfloor T_s, \frac{m}{k(n-k)}T_c\right\}$ under the all-port model, and $\max\left\{\lfloor\frac{3}{2}k\rfloor T_s, \lceil\log\frac{n!}{(n-k)!}\rceil T_s, mT_c\right\}$ under the one-port model, where m is the size of the broadcast message.

Lemma 2: A lower bound for all-to-all broadcast-

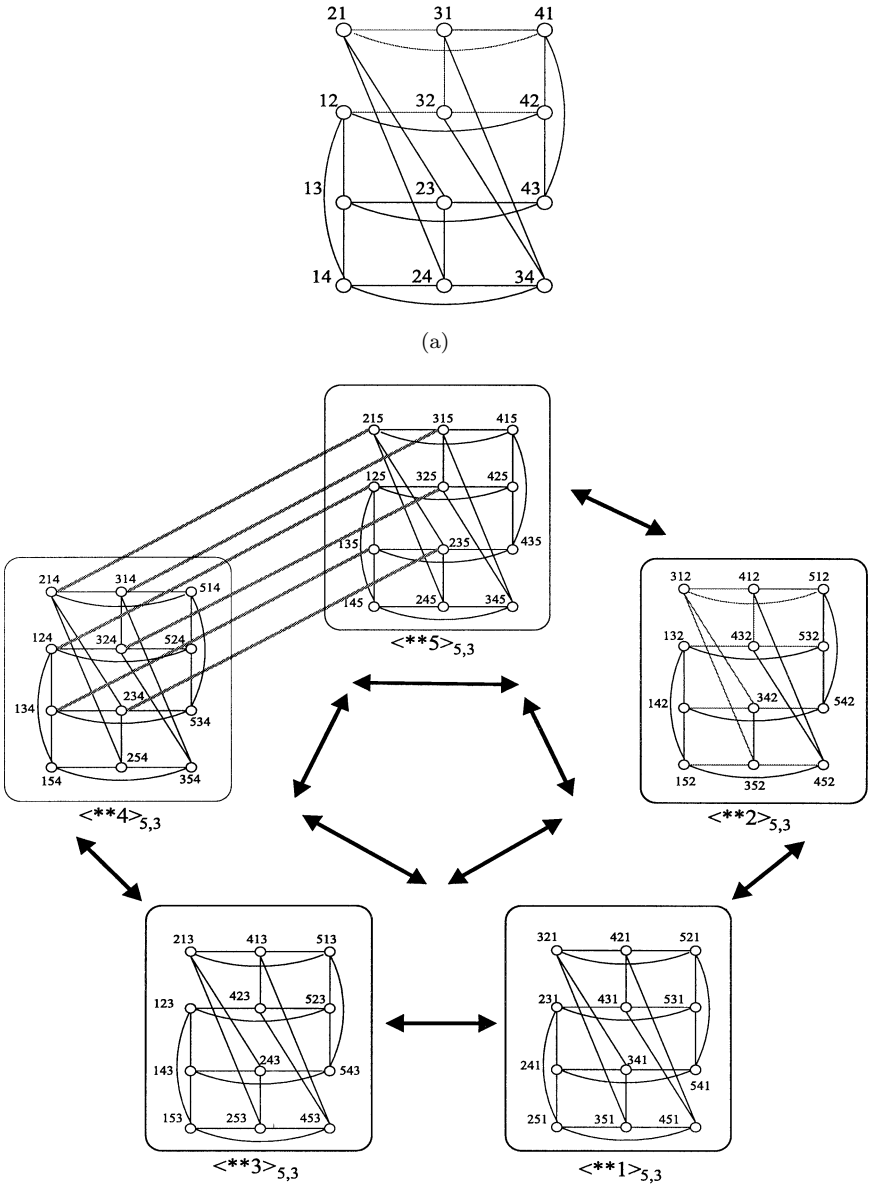


Fig. 1 An example of arrangement graphs (a) $A_{4,2}$ and (b) $A_{5,3}$.

ing in a store-and-forward $A_{n,k}$ is $\max\left\{\lfloor\frac{3}{2}k\rfloor T_s, \frac{n!}{(n-k)!}\cdot\frac{m}{k(n-k)}T_c\right\}$ under the all-port model, and $\max\left\{\lfloor\frac{3}{2}k\rfloor T_s, \lceil\log\frac{n!}{(n-k)!}\rceil T_s, \frac{n!}{(n-k)!}\cdot mT_c\right\}$ under the one-port model, where m denotes the size of the broadcast message.

3. Congestion-Free Embedding of Multiple Spanning Trees

This section presents a scheme to embed $n-k$ edge-disjoint spanning trees in an $A_{n,k}$. Our construction scheme adopts a bottom-up manner. An $A_{n,k}$ can be partitioned into $\frac{n!}{(n-k+2)!}$ copies of $A_{n-k+2,2}$. Each $A_{n-k+2,2}$ initially construct $n-k$ base spanning trees. For each $A_{n-k+3,3}$, there are $n-k+3$ copies of $A_{n-k+2,2}$ and each one containing $n-k$ base spanning trees. We propose a concatenation operation to construct larger $n-k$ spanning trees in an $A_{n-k+3,3}$. Recursively performing the concatenation operations allow us to finally construct $n-k$ spanning trees in an $A_{n,k}$.

Our $n-k$ spanning trees are constructed by two phases:

- Phase 1: Generate $n-k$ base spanning trees in each $A_{n-k+2,2}$.
- Phase 2: Perform a recursive concatenation operation to embed $n-k$ edge-disjoint spanning trees.

These phases are described as follows.

3.1 Phase 1: Generate $n-k$ Base Spanning Trees in an $A_{n-k+2,2}$

After a splitting-operation on $A_{n,k}$, $\frac{n!}{(n-k+2)!}$ copies of $A_{n-k+2,2}$'s are obtained. According to different values of n and k, $n-k$ base spanning trees in each $A_{n-k+2,2}$ can be generated as follows.

- Step 1: Locate $n-k$ roots nodes.
- Step 2: Generate $n-k$ base spanning trees in each $A_{n-k+2,2}$.

3.1.1 Step 1: Locating $n-k$ Roots in an $A_{n-k+2,2}$

We now describe how to locate $n-k$ root nodes in an $A_{n-k+2,2}$. Initially, we let each $A_{n-k+2,2}$ be partitioned into $n-k+2$ copies of $A_{n-k+1,1}$, where each $A_{n-k+1,1}$ is a $(n-k+1)$-node complete graph. Denote these $n-k$ root nodes as $R_1, R_2, \ldots$, and R_{n-k} as follows. Let $R_1 = (y_1x_2x_3\cdots x_k)$ be any node in an $A_{n-k+1,1}$. Other root nodes $R_2 = (y_2x_2x_3\cdots x_k)$, $R_3 = (y_3x_2x_3\cdots x_k), \ldots$, and $R_{n-k+1} = (y_{n-k+1}x_2x_3\cdots x_k)$ by exchanging the first bit of R_1 with α, where $\alpha \in EXT(R_1) = \{y_2, \ldots, y_{n-k+1}\}$. This task can be achieved as follows.

> $R_i = ADJ_{1,\alpha}(R_1)$, where $2 \le i \le n-k+1$ and $\alpha \in EXT(R_1)$.

Each pair of $R_1, R_2, \ldots$, and R_{n-k+1} are adjacent since $R_1, R_2, \ldots$, and R_{n-k+1} belong to an $A_{n-k+1,1}$ (a complete graph $*x_2x_3\cdots x_k$). Note that node R_{n-k+1} is used to as a template node to ensure congestion-free in our embedding.

Intuitively, every R_i is only different in the first bit. Root nodes $R_i, 1 \le i \le n-k$, are selected from $*x_2x_3\cdots x_k$. Clearly, $*x_2x_3\cdots x_k$ is an $A_{n-k+1,1}$ or $(n-k+1)$-node complete graph. Root nodes $R_1, R_2, \ldots$, R_{n-k} satisfy the root-location property. This root-location property is very useful during constructing $n-k$ spanning trees.

> **Root-Location Property:** There are $n-k$ root nodes R_i, $1 \le i \le n-k$, in same subarrangement graph. Root nodes R_i, $i = 1..n-k$, belong to an $A_{n-k+1,1}$.

Root nodes R_i, $i = 1..n-k$, are used to expand $n-k$ base spanning trees in an $A_{n-k+2,2}$. Figure 2 illustrates an example that four root nodes are 21 and 31. Template node is 41.

3.1.2 Step 2: Generate $n-k$ Base Edge-Disjoint Spanning Trees in an $A_{n-k+2,2}$

We now describe how to construct $n-k$ base spanning trees from root R_i, $1 \le i \le n-k$, for every $A_{n-k+2,2}$. The height of base spanning tree is 3. The fact that each $A_{n-k+1,1}$ is a complete graph accounts for why $n-k$ base spanning trees can be constructed if we can connect each root nodes to distinct node of all other $A_{n-k+1,1}$'s. This is because that we have the following result and Fig. 3 gives an example.

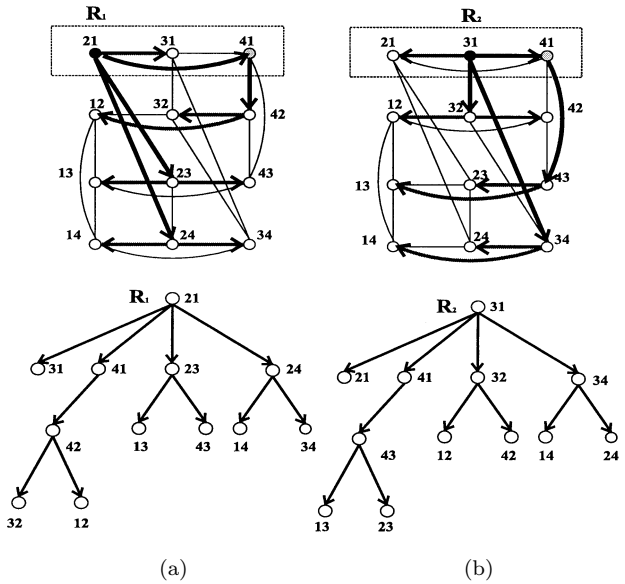


Fig. 2 Two base edge-disjoint spanning trees in an $A_{4,2}$.

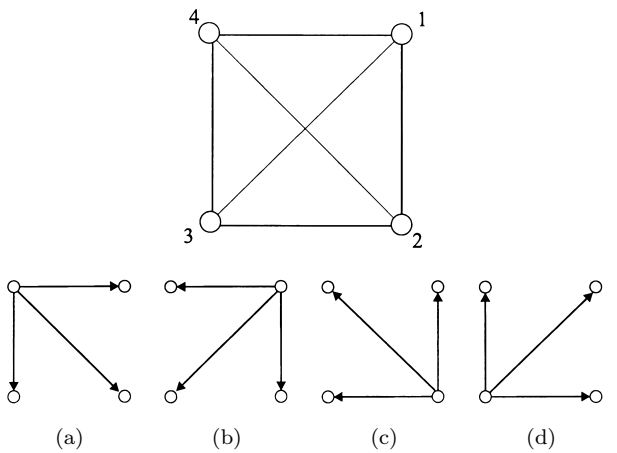


Fig. 3 Four disjoint spanning trees whose tree height = 1 in a 4-node complete graph.

Lemma 3: For a κ-node complete graph G, there exist κ disjoint spanning trees with height one. Notably, each node in G is the root node of each spanning tree.

Now we explain how to construct $n-k$ base spanning trees in each $A_{n-k+2,2}$. As stated in Sect. 2, a n-dimensional arrangement graph contains n subarrangements that we use to derive the desired spanning trees. Given $n-k$ root nodes R_i, $1 \le i \le n-k$, let R be one of these; we must connect R to other $n-1$ subarrangements. For node R, we use a single edge to connect the R to $n-k$ of subarrangements. We use an intermediate node in same subarrangement as the bridge node to

connect R to the remaining $k-1$ subarrangements by two edges. If $k=2$, then there is only one subarrangement connecting by two edges.

Rules A1 and A2 formalize the construction.

A1: (Single edge) Let node $\dot{R}$ denote connected node in every $n-k$ subarrangements.

$$\dot{R} = ADJ_{2,\alpha}(R), \text{ where } \alpha \in EXT(R).$$

A2: (Two edges) Let node $\ddot{R}$ denote connected node in one of remaining $k-1$ subarrangements which connected by two edges (Note that in this case, $k=2$). Recall in phase 1, node R_{n-k+1} is the intermediate nodes for node R.

$$\ddot{R} = ADJ_{2,\beta}(R_{n-k+1}), \text{ where } \beta \text{ is the first bit value in } R.$$

Figure 2 (a) shows that root node 21 uses distinct single edge to connecting nodes 23 and 24, and connecting node 42 by $ADJ_{2,2}(41)$, where node 41 is template node. Figure 2 (b) displays that root node 31 directly connects to 32 and 34 but connects 43 by $ADJ_{2,3}(41)$.

Now we verifies the correctness of $n-k$ base spanning trees in an $A_{n-k+2,2}$. Some notations are defined firstly. Given root nodes R_i, $1 \le i \le n-k$, are constructed by phase 1 of Sect. 3. An $A_{n-k+2,2}$ can be partitioned into $n-k+2$ copies of $A_{n-k+1,1}$ or $A'_{n-k+1,1}$ along dimension two or one. First, let an $A_{n-k+2,2}$ be partitioned into $n-k+2$ copies of $A_{n-k+1,1}$ along second dimension. Root nodes R_i, $1 \le i \le n-k$, located in one of $A_{n-k+1,1}$. Let $IE(R_i)$, $1 \le i \le n-k$, denote a set of all possible internal edges within every $A_{n-k+1,1}$. Let $EE(R_i)$ denote a set of all possible external edges outgoing every $A_{n-k+1,1}$. For example as illustrated in Fig. 4 (a), all bold edges are $IE(R_i)$ and all dash edges are $EE(R_i)$. Similarly, the same $A_{n-k+2,2}$ can be partitioned into $n-k+2$ copies of $A'_{n-k+1,1}$ along first dimension as shown in Fig. 4 (b).

Some important properties are used later as stated herein. For $1 \le i \le n-k$, we have following properties.

P1: Edges in $IE(R_i)$ and edges in $EE(R_i)$ are equal to all edges in the $A_{n-k+2,2}$.

P2: Edges in $IE(R_i)$ and edges in $EE(R_i)$ are distinct.

Lemma 4: There exist $n-k$ edge-disjoint spanning trees $ST_{n-k+2,2}(R_i)$ in an $A_{n-k+2,2}$, for $1 \le i \le n-k$.

Proof. Given root nodes R_i, $1 \le i \le n-k$, are located in one of partitioned $A_{n-k+1,1}$. Every root node needs to connect to other $A_{n-k+1,1}$. Therefore, $n-k$ spanning trees $ST_{n-k+2,2}(R_i)$, for $1 \le i \le n-k$, are mutually disjoint due to the fact that every root node satisfy the following conditions:

1: All edges of $R_1, R_2, \ldots$, and R_{n-k} connecting to the same $A_{n-k+1,1}$ are disjoint.
2: All nodes in the same $A_{n-k+1,1}$ connecting to R_1, $R_2, \ldots$, and R_{n-k} differ from each other.

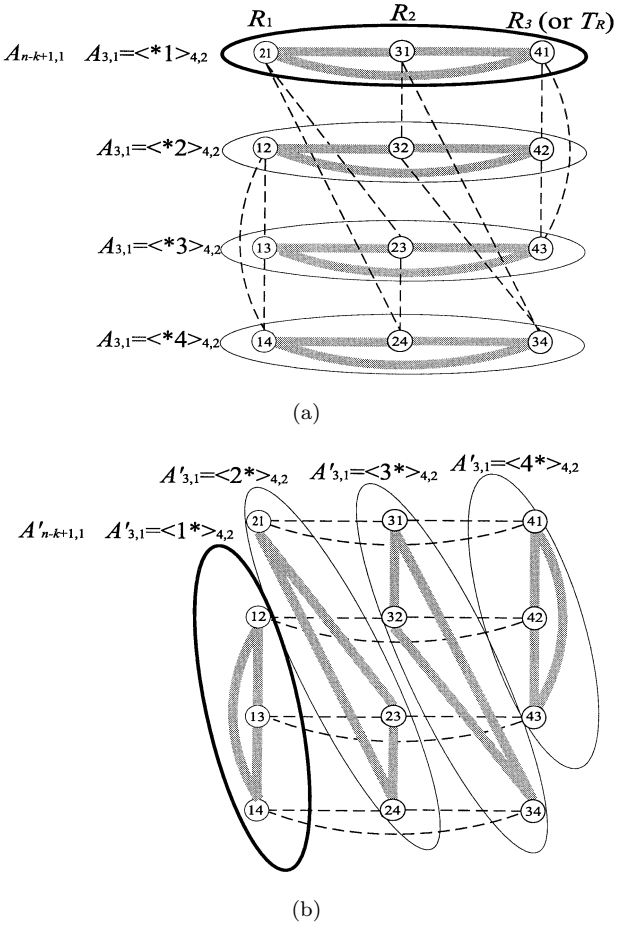
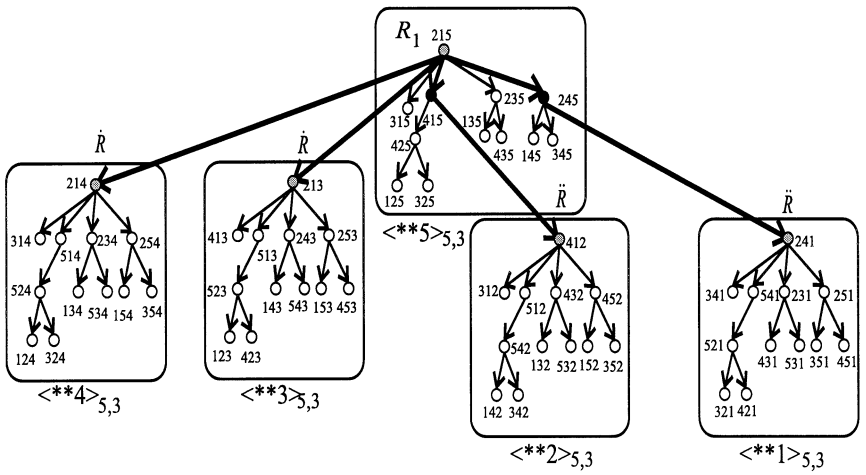


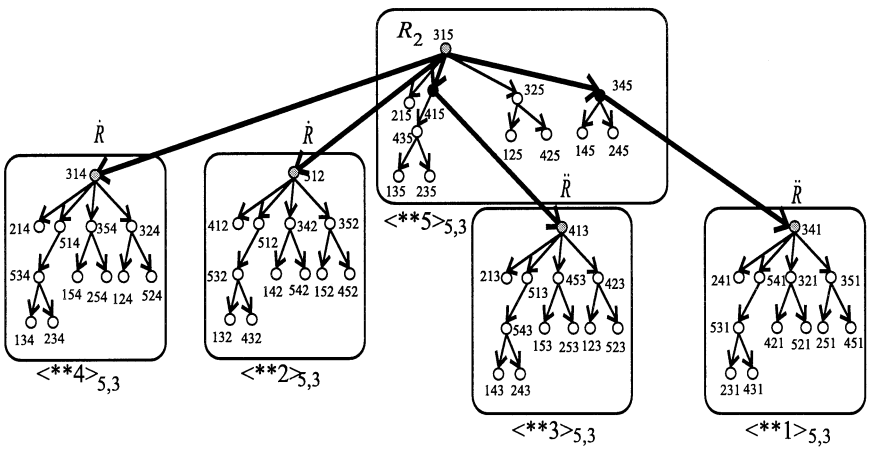
Fig. 4 Edge distribution in an $A_{4,2}$; (a) all internal edges within each $A_{3,1}$ in $A_{4,2}$, (b) all internal edges within each $A'_{3,1}$ in $A_{4,2}$.

The reason is stated as follows. Recalled again, since $k=2$, for all R_i, we use $n-2$ distinct single edges connecting to $n-2$ copies of $A_{n-k+1,1}$ and use two edge to connect with remaining one $A_{n-k+1,1}$. Intuitively, all edges connecting from $R_1, R_2, \ldots$, and R_{n-k} to template node R_{n-k+1} are distinct since all of these nodes located in a $(n-k+1)$-node complete graph (or $A_{n-k+1,1}$). Recall previous notation, edges in all possible $A_{n-k+1,1}$ are denoted as $IE(R_i)$ and all edges in all possible $A'_{n-k+1,1}$ are denoted as $EE(R_i)$. For condition 1, all edges of $R_1, R_2, \ldots$, and R_{n-k} connecting to the same $A_{n-k+1,1}$ are disjoint because every edge belongs to different $A'_{n-k+1,1}$. Remember, these edges are belong to $EE(R_i)$. For condition 2, all nodes in the same $A_{n-k+1,1}$ connecting to $R_1, R_2, \ldots$, and R_{n-k} differ from each other due to the fact that each of these nodes belongs to distinct $A'_{n-k+1,1}$. □

Theorem 5: There exist $n-k$ base edge-disjoint spanning trees $ST_{n-k+2,2}(R_i)$ in an $A_{n-k+2,2}$, where $1 \le i \le n-k$. Each spanning tree's height is 3.



(a)



(b)

Fig. 5 Example of two spanning trees in an $A_{5,3}$; (a) from root node 215, (b) from root node 315.

3.2 Phase 2: Construction of $n-k$ Edge-Disjoint Spanning Trees in an $A_{n,k}$

The above section constructed $n-k$ base spanning trees in $A_{n-k+2,2}$, and each base spanning tree's height is 3. In this section, we describe how to construct $n-k$ spanning trees $ST_{n,k}(R_i)$, for $1 \le i \le n-k$ in an $A_{n,k}$, where the height of each spanning tree is $2k-1$.

For induction, we construct $n-k$ spanning trees in an $A_{n,k}$ by using $n(n-k)$ spanning tress in n copies of $A_{n-1,k-1}$. Our major task is to connect $n(n-k)$ spanning subtrees into $n-k$ spanning trees $ST_{n,k}(R_i),\ 1 \le i \le n-k$.

Consider an $A_{n,k}$ which is partitioned into n copies of $A_{n-1,k-1}$. Assume that $n-k$ spanning trees $ST_{n-1,k-1}(R_i),\ i = 1..n-k$, can be constructed in each $A_{n-1,k-1}$ if their root nodes satisfied the root-location property. A randomly selected $A_{n-1,k-1}$ servers as beginning spanning tree. From this $A_{n-1,k-1}$, there exist $n-k$ spanning trees $ST_{n-1,k-1}(R_i),\ 1 \le i \le n-k$, where R_i are root nodes. Let root node R_i connect to root node R'_i, where R'_i located in one of other $A_{n-1,k-1}$. If R'_i satisfy the root-location property, then we can correctly construct $n-k$ spanning trees $ST_{n,k}(R_i),\ 1 \le i \le n-k$, in an $A_{n,k}$.

As similar work mentioned in phase 1. A n-dimensional arrangement graph contains n subarrangements that we use to derive the desired spanning trees. Given $n-k$ root nodes $R_i,\ 1 \le i \le n-k$, let R are one of these; we must connect R to other $n-1$ subarrangements. For each R, we use a single edge to connect the R to $n-k$ of subarrangements. We also use an intermediate node in same subarrangement as the bridge node to connect R to the remaining $k-1$ subarrangements by two edges.

Rules A1' and A2' formalize our recursively spanning tree construction.

A1': (Single edge, the part is same as A1.) Let node $\dot{R}$ denote connected node in every $n-k$ subarrangements.

$\dot{R} = ADJ_{2,\alpha}(R),$ where $\alpha \in EXT(R)$.

A2': (Two edges, this part is same as A2.) Let node $\ddot{R}$ denote connected node in one of remaining $k-1$ subarrangements which connected by two edges. Nodes R_{n-k+1} is the intermediate node for node R.

$\ddot{R} = ADJ_{2,\beta}(R_{n-k+1})$, where β is the first bit value in R.

Figure 5 illustrates that two spanning trees in an $A_{5,3}$, where root nodes are 215 and 315. Assume that R'_i, $1 \le i \le n-k$, are new connecting root nodes in each of other $(n-1)$-subarrangement. Ensuring that root nodes R'_i satisfied the root-location property would allow us to establish our spanning trees $ST_{n,k}(R_i)$, for $i = 1..n-k$. In lemma 5, we show the correctness that root nodes R'_i satisfied root-location property, for $i = 1..n-k$.

Lemma 6: There are $n-k$ root nodes R'_i, $1 \le i \le n-k$, in same subarrangement graph are satisfied the root-location property.

Proof. There are n subarrangements $<*^{k-1}\beta>_{n,k}$, where $\beta \in <n>$. One of the subarrangements $<*^{k-1}\beta'>_{n,k}$ is selected as base spanning trees. In addition, let root nodes $R_1, R_2, \ldots$, and R_{n-k} belong to a $A_{n-k+1,1}$ denoted as $<\alpha*^{k-2}\beta'>_{n,k}$, where $\alpha \in <n> - \{*^{k-2}\beta'\} - \{t\}$ and let $<t*^{k-2}\beta'>_{n,k}$ be a template node. Our results demonstrate that root nodes $R'_1, R'_2, \ldots$, and R'_{n-k} in each $A_{n-1,k-1}$ also belong to a distinct $A_{n-k+1,1}$. Three cases are discussed as follows.

1: There is a subarrangement $<*^{k-1}\beta''>_{n,k}$ which can directly connect with $<*^{k-1}\beta'>_{n,k}$ by one direct edge. This is owing to that if root nodes R_i are $<\alpha*^{k-2}\beta'>_{n,k}$, R'_i can be obtained by $<\alpha*^{k-2}\beta''>_{n,k}$, where $\beta'' = t$ and $1 \le i \le n-k$. Intuitively, R'_i are $<\underline{\alpha}*^{k-2}t>_{n,k}$, which differ in one bit, so satisfy the root-location property. Figure 6 (a) provides an example, root node 215 $\leftrightarrow$ $\underline{2}14$ and root node 315 $\leftrightarrow$ $\underline{3}14$, where $\underline{2}14$ and $\underline{3}14$ $\in <*14>_{5,3}$.

2: If root nodes R_i are $<\alpha*^{k-2}\beta'>_{n,k}$, there are $n-k$ subarrangement $<*^{k-1}\beta''>_{n,k}$ connecting to $<*^{k-1}\beta'>_{n,k}$ by one edge or two edges, where $\beta'' = \alpha' \in \alpha$. If root nodes R_i are $<\alpha*^{k-2}\beta'>_{n,k}, \alpha \ne \alpha'$, one direct edge is $<\alpha*^{k-2}\beta'>_{n,k} \leftrightarrow <\alpha*^{k-2}\alpha'>_{n,k}$. If other root nodes R_j are $<\alpha*^{k-2}\beta'>_{n,k}$ and $\alpha = \alpha'$, two edges are $<\alpha*^{k-2}\beta'>_{n,k} \leftrightarrow <t*^{k-2}\beta'>_{n,k} \leftrightarrow <t*^{k-2}\alpha'>_{n,k}$. Clearly, $<\underline{\alpha}*^{k-2}\alpha'>_{n,k}$ and $<\underline{t}*^{k-2}\alpha'>_{n,k}$, which differ in one bit, satisfy the root-location property. Figure 6 (b) provides an example, root node 215 $\leftrightarrow$ $\underline{2}13$ and root node 315 $\leftrightarrow$ 415 $\leftrightarrow$ $\underline{4}13$, where $\underline{2}13$ and $\underline{4}13 \in <*13>_{5,3}$.

3: There are $k-2$ subarrangement $<*^{k-1}\beta''>_{n,k}$ which can directly connected to $<*^{k-1}\beta'>_{n,k}$ by two edges. This is owing to that if root nodes R_i are $<\alpha*^{k-2}\beta'>_{n,k}$, root nodes R'_i are obtained by $<\alpha \underbrace{*\cdots*\underbrace{x}_{j}*\cdots*}_{k-2} \beta'>_{n,k} \leftrightarrow$ $<\alpha \underbrace{*\cdots*\underbrace{t}_{j}*\cdots*}_{k-2} \beta'>_{n,k} \leftrightarrow$ $<\alpha \underbrace{*\cdots*\underbrace{t}_{j}*\cdots*}_{k-2} x>_{n,k}$, where $1 \le i \le n-k, 1 \le j \le k-2$. Intuitively, R'_i are $<\underline{\alpha}\underbrace{*\cdots*\underbrace{t}_{j}*\cdots*}_{k-2} x>_{n,k}$, which differ in one bit, so satisfy the root-location property. Figure 6 (c) provides an example, root node 215 $\leftrightarrow$ 245 $\leftrightarrow$ $\underline{2}41$ and root node 315 $\leftrightarrow$ 345 $\leftrightarrow$ $\underline{3}41$, where $\underline{2}41$ and $\underline{3}41 \in <*41>_{5,3}$. □

Lemma 7: The height of $n-k$ edge-disjoint spanning trees $ST_{n,k}(R_i)$ in an $A_{n,k}$ is $2k-1$.

Proof. There are k steps to construct $ST_{n,k}(R_i)$, $1 \le i \le n-k$. Each step needs at most two edges except that the final step in $A_{n-k+1,1}$ needs one single edge since $A_{n-k+1,1}$ is a complete graph. The height is $2(k-1)+1 = 2k-1$. □

Theorem 8: There exist $n-k$ edge-disjoint spanning trees $ST_{n,k}(R_i)$ with height $2k-1$ in an $A_{n,k}$, where $i = 1..n-k$.

Proof. See proof in lemmas 6 and 7. □

The fact that there existing $n-k$ edge-disjoint spanning trees $ST_{n,k}(R_1), ST_{n,k}(R_2), \ldots$, and $ST_{n,k}(R_{n-k})$ in an $A_{n,k}$ with height $2k-1$. The fact that each pair of root nodes $R_1, R_2, \ldots$, and R_{n-k} are adjacent allows us to logical construct a broadcast tree with height $2k$. Note that we use the template node R_{n-k+1} as the new root node of the logical broadcast tree. Consequently, a broadcasting algorithm using $n-k$ edge-disjoint spanning trees can be obtained. The detailed algorithm is given in the next section.

4. Broadcast Algorithm and Performance Analysis

One-to-all broadcast refers to the problem of sending a message from a source node to all other nodes in the network. All-to-all broadcast refers to the problem of sending a message from all source nodes to all other nodes in the network. Two communication models are considered as follows.

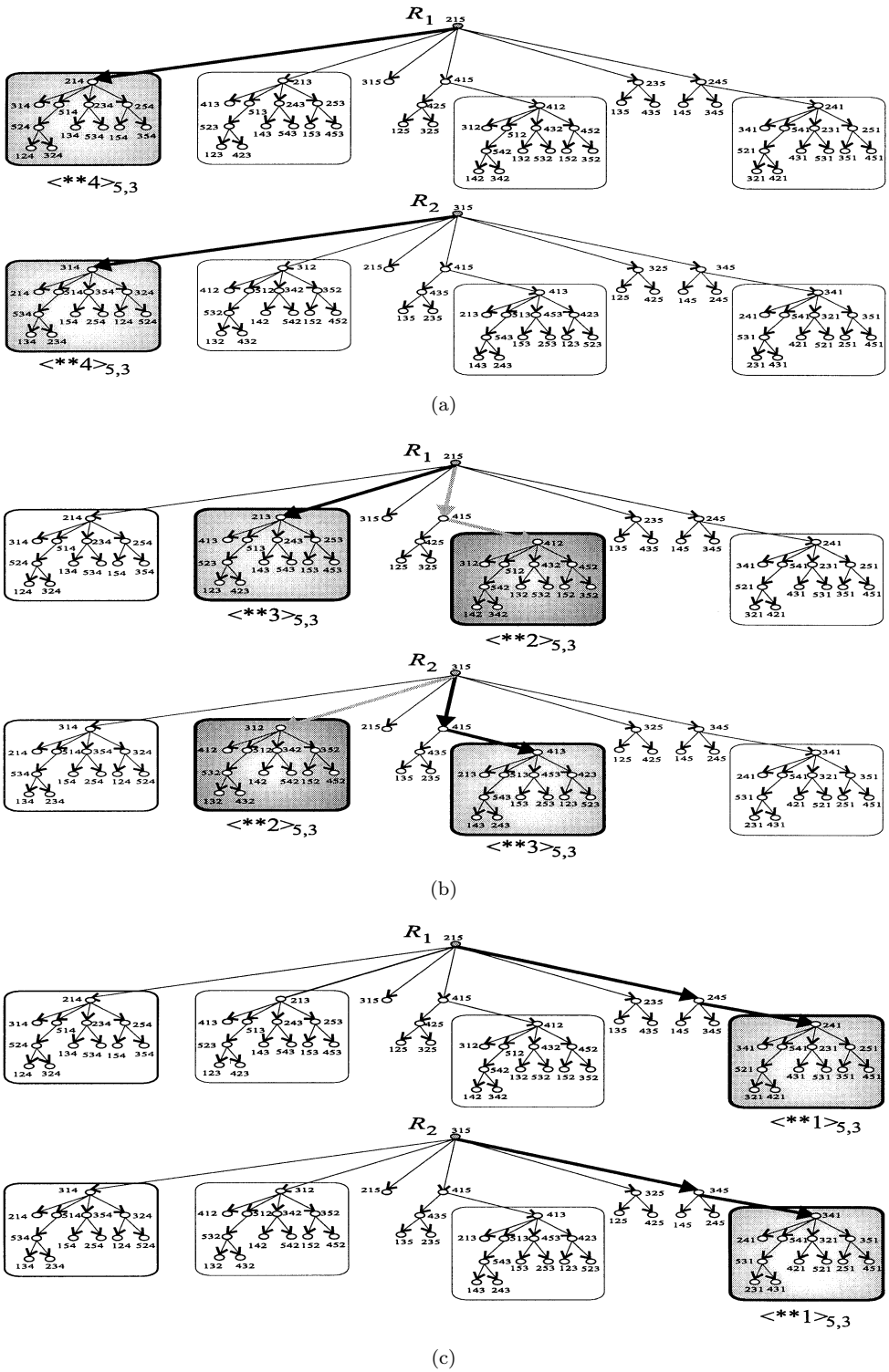


Fig. 6 Example of proof in Lemma 6.

4.1 Under All-Port Model

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

Algorithm 1: /* One-to-all-broadcast, all-port */

1) Divide the message M evenly into $p(n-k)$ parts, each called a "*message segment*" and of size $\frac{m}{p(n-k)}$.
2) In each time slot, node r issues $n-k$ message segments to the network, each along one of the trees $ST_{n,k}(R_i)$, $i = 1..n-k$. A message segment is then propagated along the tree it is issued. Each time slot helps propagate all message segments it received in the previous time slot to the subsequent nodes in the corresponding trees.

Next, we analyze the communication latency of the algorithm. Notably, our analysis neglects the computational time (such as making routing decision or packing/unpacking packets). Let h be the maximal height of $ST_{n,k}(R_i)$, $i = 1..n-k$. The broadcast algorithm run in time

$$T = h\left(T_s + \frac{m}{p(n-k)}T_c\right) + (p-1)\left(T_s + \frac{m}{p(n-k)}T_c\right), \quad \text{where } h = 2k \tag{1}$$

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 pipeline effect. To minimize Eq. (1), let the derivative of T with respect to p equal 0,

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

Therefore, we obtain

$$p = \sqrt{\frac{m(h-1)T_c}{(n-k)T_s}},$$

and we have knowledge that $h = 2k$. Therefore,

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

Theorem 9: Under the all-port model, one-to-all broadcast can be executed in $A_{n,k}$ within time $O((2k-1)T_s + \frac{m}{n-k}T_c)$.

According to Theorem 9, we obtain an order of $(\sqrt{2(k-1)T_s} + \sqrt{\frac{mT_c}{n-k}})^2 = O((2k-1)T_s + (4k-2)\sqrt{\frac{m}{(2k-1)(n-k)}T_sT_c} + \frac{m}{n-k}T_c) \approx O((2k-1)T_s + \frac{m}{n-k}T_c)$, where $h = 2k$ and message size m is sufficiently large. Day and Tripathi's spanning tree [7] obtained an order of $O((\lceil \frac{3}{2}k \rceil - 1)T_s + mT_c)$. Our broadcast algorithm is faster than Day and Tripathi's method [7], when $m > \frac{k}{2} \times \frac{T_s}{T_c}$. However, our algorithm is an order of $O(k)$ higher than optimum as shown in Table 1.

More Specially, *label change* (*LC*) and *element change* (*EC*) schemes proposed by Tseng et al. [13] are used to design our all-to-all broadcasting such that each node in $A_{n,k}$ can use $n-k$ spanning trees. Furthermore, the network can be equally loaded at every time step using Tseng's broadcasting scheme [13]. Therefore, we can have the following result.

Theorem 10: Under to all-port model, all-to-all broadcast can be implemented in $A_{n,k}$ with time

$$O\left(2k \times T_s + \frac{mn!}{(n-k)(n-k)!}T_c\right).$$

Proof. Assume that in each iteration of time slot, all message segments can be combined into one packet and send at one time. Therefore, the start-up overhead is $2k \times T_s$. To calculate the transmission time, let m be the size of broadcast messages. To propagate a message segment (of size $\frac{m}{n-k}$) along a spanning tree (of $\frac{n!}{(n-k)!}$ links), network bandwidth of $\frac{m}{n-k} \times \frac{n!}{(n-k)!} \times T_c$ required. Cumulatively, $(n-k)\frac{n!}{(n-k)!}$ message segments are being transmitted. Therefore, the total network bandwidth required is $m(\frac{n!}{(n-k)!})^2T_c$. Since the network is equally loaded at every time step, the bandwidth is evenly distributed to all $(n-k)\frac{n!}{(n-k)!}$ links in the network. Therefore, the transmission time is $\frac{m(\frac{n!}{(n-k)!})^2T_c}{(n-k)(\frac{n!}{(n-k)!})} = \frac{mn!}{(n-k)(n-k)!}T_c$. □

4.2 Under 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 number of $ST_{n,k}(R_i)$ degree time slots. The activity can be simulated as follows: in the first time slot, the one-port node simulates the all-port node's activity along 1-*edge*; in the second time slot, the one-port node simulates the all-port node's activity along 2-*edge*; etc. Clearly, the communication follows the one-port model.

Algorithm 2: /* One-to-all-broadcast, one-port */

1) Divide the message M evenly into $p(n-k)$ parts, each called a "*message segment*" and of size $\frac{m}{p(n-k)}$.
2) In each time slot, node r issues $k(n-k)$ message segments to the network, each along one of the trees $ST_{n,k}(R_i)$, $i = 1..n-k$. A message segment is then propagated along the tree it is issued. Each time slot helps propagate one message segments it receives in the previous time slot to the subsequent nodes in the corresponding trees.
3) Repeatedly perform step 2 until all *message segments* have be completed.

In fact, each root of our spanning trees does not have full degree to connection leaf node accounting for why some nodes do not process a message in each time slot. In the one-port model, out analysis neglects the computational time (such as making routing decision

Table 1 Comparison of one-to-all broadcast algorithms on start-up cost, transmission cost, and overall time complexity.

Model	Algorithm	Start-up cost	Trans. cost	Overall complexity
All-port	Day [7]	$O((\lceil \frac{3}{2}k \rceil - 1)T_s)$	$O(mT_c)$	$O((\lceil \frac{3}{2}k \rceil - 1)T_s + mT_c)$
	Our	$O((2k-1)T_s)$	$O(\frac{m}{n-k}T_c)$	$O((2k-1)T_s + \frac{m}{n-k}T_c)$
	Optimal	$\lfloor \frac{3}{2}k \rfloor T_s$	$\frac{m}{k(n-k)}T_c$	$\max(\lfloor \frac{3}{2}k \rfloor T_s, \frac{m}{k(n-k)}T_c)$
One-port	Day [7]	$O(k(n-k) \times (\lceil \frac{3}{2}k \rceil - 1)T_s)$	$O(k(n-k)mT_c)$	$O(k(n-k) \times ((\lceil \frac{3}{2}k \rceil - 1)T_s + mT_c))$
	Our	$O(k(n-k) \times (2k-1)T_s)$	$O(kmT_c)$	$O(k(n-k) \times ((2k-1)T_s + \frac{m}{n-k}T_c))$
	Optimal	$\max(\lfloor \frac{3}{2}k \rfloor T_s, \lceil \log \frac{n!}{(n-k)!} \rceil T_s)$	mT_c	$\max(\lfloor \frac{3}{2}k \rfloor T_s, \lceil \log \frac{n!}{(n-k)!} \rceil T_s, mT_c)$

or packing/unpacking packets). Let $D_{n,k} = k(n-k)$ denote the degree of $A_{n,k}$. The broadcast algorithm takes time in

$$T = D_{n,k}\left(h + \sqrt{\frac{m(h-1)T_c}{(n-k)T_s}} - 1\right) \times \left(T_s + \sqrt{\frac{mT_sT_c}{(h-1)(n-k)}}\right),$$

where $h = 2k$.

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.

Theorem 11: Under the one-port model, one-to-all broadcast can be implemented in $A_{n,k}$ within time $O(k(n-k)(2k-1)T_s + kmT_c)$.

By Theorem 11, we obtain an order of $k(n-k)(\sqrt{(2k-1)T_s} + \sqrt{\frac{mT_c}{n-k}})^2 = O(k(n-k) \times ((2k-1)T_s + (4k-2)\sqrt{\frac{m}{(2k-1)(n-k)}T_sT_c} + \frac{m}{n-k}T_c)) \approx O(k(n-k)(2k-1)T_s + kmT_c)$, where $h = 2k$. Day and Tripathi's result [7] obtained an order of $O(k(n-k)(\lceil \frac{3}{2}k \rceil - 1)T_s + k(n-k)mT_c)$, where message size m is sufficiently large. Our broadcast algorithm is faster than Day and Tripathi's algorithm [7], when $m > \frac{k}{2} \times \frac{T_s}{T_c}$. However, our algorithm is an order of $O(k)$ higher than optimum as summarized in Table 1.

In addition, a node with one-port communication capability can simulate an all-port node by delay factor of $n-k$. Therefore, we have following all-to-all broadcasting result.

Theorem 12: Under to one-port model, all-to-all broadcast can be implemented in $A_{n,k}$ with time

$$O\left(2k^2(n-k)T_s + \frac{kmn!}{(n-k)!}T_c\right).$$

5. Conclusions

This paper addresses the broadcast problem in one-port/multi-port arrangement graphs using packet-switching (or store-and-forward) technique. A broadcast algorithm is also proposed by constructing $n-k$ spanning trees. Under the all-port model, the one-to-all broadcast algorithm proposed herein can be executed in $A_{n,k}$ in time $O((2k-1)T_s + \frac{m}{n-k}T_c)$. Under the one-port model, the one-to-all broadcast algorithm can be performed in time $O(k(n-k)(2k-1)T_s + kmT_c)$. Under the all-port model, the all-to-all broadcast algorithm can be executed in time $O(2k \times T_s + \frac{mn!}{(n-k)(n-k)!}T_c)$. Under the one-port model, our one-to-all broadcast algorithm can be implemented in time $O(2k^2(n-k)T_s + \frac{kmn!}{(n-k)!}T_c)$. Comparing our results with those of Day and Tripathi [7] reveals that although the broadcast algorithm proposed herein is an order of $O(n-k)$ faster than the algorithm of Day and Tripathi [7] when message size $m > \frac{k}{2} \times \frac{T_s}{T_c}$, it is an order of $O(k)$ higher than optimum. Work is currently underway to develop an optimal or near-optimal broadcast algorithm in $A_{n,k}$ by exploiting $O(k(n-k))$ edge-disjoint spanning trees.

References

[1] S.B. Akers, D. Harel, and B. Krishnameurthy, "The star graph: An attractive alternative to the n-cube," Proc. ICPP '87, pp.393–400, Aug. 1987.

[2] L.Q. Bai, H. Ebara, H. Nakano, and H. Maeda, "Fault-tolerant broadcasting on the arrangement graph," The Computer Journal, vol.41, no.3, pp.171–184, 1998.

[3] Y.-S. Chen and J.-P. Sheu, "A fault-tolerant reconfiguration scheme in the faulty star graph," Journal of Information Science and Engineering, vol.16, no.1, Jan. 2000.

[4] K. Day and A. Tripathi, "Characterization of node disjoint path in arrangement graphs," Technical Report TR91-43, Computer Science Department, University of Minnesota, 1991.

[5] K. Day and A. Tripathi, "Arrangement grapha: A class of generalized star graphs," Information Processing Letter, vol.42, no.5, pp.235–241, 1992.

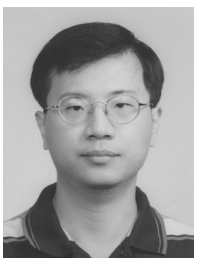
[6] K. Day and A. Tripathi, "Embedding of cycles in arangement graphs," IEEE Trans. Comput., vol.12, no.8, pp.1002–1006, 1992.

[7] K. Day and A. Tripathi, "Embedding grids, hypercube, and trees in arrangement graphs," Proc. International Conference on Parallel Processing, vol.III, pp.65–72, 1993.

[8] S.-Y. Hsieh, G.-H. Chen, and C.-W. Ho, "Fault-free Hamiltonian cycles in faulty arrangement graph," IEEE Trans. Parallel and Distributed Systems, vol.10, no.3, pp.223–237, March 1999.

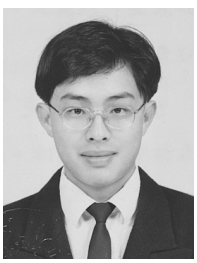
[9] J.-P. Sheu, C.-T. Wu, and T.-S. Chen, "An optimal broadcasting algorithm without message redundancy in star graphs," IEEE Trans. Parallel and Distributed Systems,

vol.6, no.6, pp.653–658, 1995.
[10] S.-S. Tsai and S.-J. Horng, "Efficient embedding hypercube on arrangement graphs," Journal of Information Science and Engineering, vol.12, no.3, pp.585–592, Dec. 1996.
[11] Y.-C. Tseng, S.-H. Chang, and J.-P. Sheu, "Fault-tolerant ring embedding in star graphs," IEEE Trans. Parallel and Distributed Systems, vol.8, no.12, pp.1185–1195, Dec. 1999.
[12] Y.-C. Tseng, Y.-S. Chen, T.-Y. Juang, and C.-J. Chang, "Congestion-free, dilation-2 embedding of complete binary tree in star graphs," Networks, vol.33, no.3, pp.221–231, May 1999.
[13] Y.-C. Tseng and J.-P. Sheu, "Toward optimal broadcast in a star graph using multiple spanning trees," IEEE Trans. Comput., vol.46, no.5, pp.593–599, 1997.



Yuh-Shyan Chen received the B.S. degree in computer science from Tamkang University, Taiwan, Republic of China, in June 1988 and the M.S. and Ph.D. degrees in Computer Science and Information Engineering from the National Central University, Taiwan, Republic of China, in June 1991 and January 1996, respectively. He joined the faculty of Department of Computer Science and Information Engineering at Chung-Hua University, Taiwan, Republic of China, as an associate professor in February 1996. His current research include parallel algorithm, collective communication, interconnection network, fault-tolerant algorithm, and mobile communication. Dr. Chen is a member of the IEEE Computer Society and Phi Tau Phi Society.

Tong-Ying Juang is an associate professor in the Department of Statistics at National Chung Hsing University. His research interests include distributed and parallel algorithms, fault-tolerant distributed computing, mobile computing, and interconnection networks. He received a B.S. in Naval architecture from National Taiwan University, and his M.S. and Ph.D. in computer science from the University of Texas at Dallas 1989 and 1992, respectively. Juang is a member of the IEEE. Contact him at the Dept. of Statistics, National Chung Hsing University, Taipei, 10433, Taiwan.



En-Huai Tseng received the B.S. and M.S. degrees in Computer Science and Information Engineering from Chung-Hua University, Taiwan, Republic of China, in June 1996 and June 1998, respectively. His current research interests include parallel and distributed processing and collective communication.