*Proc. Natl. Sci. Counc. ROC(A)*
Vol. 22, No. 6, 1998. pp. 811-819

***(Scientific Note)***

# Efficient All-to-All Broadcast in Star Graph Interconnection Networks

YU-CHEE TSENG, WU-LIN CHANG, AND JANG-PING SHEU

*Department of Computer Science and Information Engineering*
*National Central University*
*Chung-Li, Taiwan, R.O.C.*

(Received December 8, 1997; Accepted May 5, 1998)

**ABSTRACT**

In this paper, we study the all-to-all personalized and non-personalized broadcast problems in a star graph. This work is motivated by the observation that existing works in the literature only try to optimize the data transmission time, but ignore the start-up time required to initialize communication. As a result, existing algorithms, although claimed to be optimal, are only so when the start-up time is negligible. In this paper we try to optimize both costs at the same time. We develop an all-to-all personalized broadcast algorithm that is more efficient than existing results. We also present an all-to-all non-personalized broadcast algorithm that outperforms existing results in most cases except when the start-up time is very small (in a ratio of $O(\frac{1}{(n-1)!})$, where $n$ is the dimension of the star graph) compared to the transmission time. Extensive simulations have been conducted, which show 10%~80% improvement over existing results.

**Key Words:** all-to-all personalized/non-personalized broadcast, collective communication, interconnection network, routing, star graph

## I. Introduction

Designing and implementing multicomputer networks with versatile topologies, such as the linear array, ring, mesh, tree, hypercube, etc., has become possible due to fast advances in hardware technologies. One new interconnection network that has attracted much attention recently is the star graph (Akers *et al.*, 1987; Akers and Krishnamurthy, 1989). Part of the reason for this interest is its symmetric and recursive nature, and superior (lower) node degree and comparable diameter as opposed to hypercubes. Indeed, many studies have been done on the star graph's topological properties (Day and Tripathi, 1994; Qiu *et al.*, 1994), embedding capability (Jwo *et al.*, 1990; Nigam *et al.*, 1990; Tseng *et al.*, 1997), communication capability (Akl *et al.*, 1993; Fragopoulou and Akl, 1995; Mendia and Sarkar, 1992; Misic and Jovanovic, 1994; Qiu, 1995; Sheu *et al.*, 1995), and fault-tolerant capability (Bagherzadeh *et al.*, 1993; Jovanovic and Misic, 1994; Latifi, 1993).

Broadcasting is a preliminary function in an interconnection network. It has many applications, such as Fast Fourier Transformation (FFT), data re-distribution in High-Performance FORTRAN, parallel graph algorithms, and parallel matrix algorithms. This problem can be classified according to the communication pattern (*one-to-all* or *all-to-all*) and the message nature (*personalized* or *non-personalized*). Many studies have been focused on various versions of this problem, e.g., one-to-all broadcast (Akers *et al.*, 1987; Akl *et al.*, 1993; Chen *et al.*, 1996; Mendia and Sarkar, 1992; Misic and Jovanovic, 1994; Qiu, 1995; Sheu *et al.*, 1993, 1995) and all-to-all broadcast (Fragopoulou and Akl, 1995; Misic and Jovanovic, 1994).

In this paper, we study the all-to-all broadcast problem in a star graph, where each node has a message to be sent to all other nodes in the network. We consider the message to be personalized or non-personalized. If it is personalized, then the message sent from a source node to every destination node may vary; otherwise, the message is the same. The routing technology is assumed to use *packet switching* with the *single-port* capability. By means of the single-port capability, a node can send, and simultaneously receive, at most one message at a time, as opposed to the *all-port* capability, where a node can send and receive along all channels simultaneously.

This work is motivated by the observation that existing works (Fragopoulou and Akl, 1995; Misic and Jovanovic, 1994) in the literature only try to optimize the data transmission time (required for packets to travel in the communication channels), but ignore the start-up time (required to initialize the communication). In current technologies, each communication involves both a start-up cost and a transmission cost, and the former cost is typically significant, in an order or more larger than the later. As a result, these algorithms (Fragopoulou and Akl, 1995; Misic and Jovanovic, 1994), although claimed to be optimal, are only so when the start-up time is negligible.

In this paper, we try to optimize both costs at the same time. We develop an all-to-all personalized broadcast algorithm that is more efficient than existing results (Fragopoulou and Akl, 1995; Misic and Jovanovic, 1994). We also present an all-to-all non-personalized broadcast algorithm that outperforms Fragopoulou and Akl (1995) in most cases except when the start-up time is very small (in a ratio of $O(\frac{1}{(n-1)!})$, where $n$ is the dimension of the star graph) compared to the transmission time. Extensive simulations have been conducted, which show about 10%~80% improvement over existing results (Fragopoulou and Akl, 1995; Misic and Jovanovic, 1994).

The algorithms presented in this paper are all based on the following idea: to perform a broadcast, instead of sending messages one by one to each node, we first group several messages together into one packet and then forward it to another node, which will help in further distributing the messages to the destination nodes. The algorithms are easy to implement. Also, the number of messages required to be grouped is an adjustable factor. Our simulations show that large gain can be obtained by only grouping a small (<20) number of messages. Note that the size of packets also places a restriction on the hardware (e.g., buffer size). Therefore, the results presented here have much practical value.

The rest of this paper is organized as follows. Section II gives some preliminary results. Algorithms for all-to-all personalized and non-personalized broadcast are presented in Section III and Section IV, respectively. Conclusions are drawn in Section V.

## II. Preliminaries

An *n-dimensional star graph*, also referred to as an $S_n$ or *n-star*, is an undirected graph with $n!$ nodes. Each node is represented by a permutation of $n$ symbols $\{1, 2, ..., n\}$. The edges of $S_n$ are defined using $n-1$ functions, $g_2, g_3, ..., g_n$, where given any node $x=x_1...x_i...x_n$ in $S_n$, the function $g_i(x)=x_i...x_1...x_n$ (i.e., swap the first and the $i$-th symbols and keep the rest of the symbols unchanged). Two nodes $x$ and $y$ are connected by an edge *along* dimension $i$ iff $x=g_i(y)$ for any $i=2..n$. Figure 1 shows an example of an $S_4$.

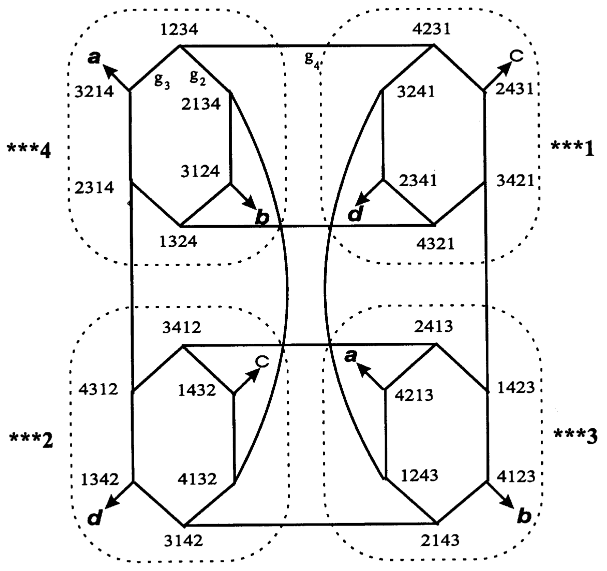

**Fig 1.** A 4-dimensional star graph $S_4$.

An $S_n$ is a recursive structure that contains many *substars*. Formally, a *k-dimensional substar*, or *k-substar*, is denoted as a string $X= x_1x_2...x_n$, where $x_1=*$ and $x_i\in\{*,1, 2, ..., n\}$, $2\le i\le n$. The symbol $*$ means "don't care". In $X$, there are exactly $k$ $*$'s, and no two non-$*$ symbols are identical. Substar $X$ is a subgraph of $S_n$ consisting of all the vertices obtained from $X$ by replacing all the $*$'s with arbitrary (but legal) digits $\{1, 2, ..., n\}$, and all the edges induced by these vertices. For instance, in Fig. 1, we show the four 3-substars $*^31$, $*^32$, $*^33$, and $*^34$ in $S_4$, where $*^k$ denotes a string of $k$ $*$'s. It is a simple fact that a $k$-substar is also an $S_k$.

In $S_n$, consider the problem of minimal routing from any node $x=x_1x_2...x_n$ to another node $y=y_1y_2...y_n$, $x\neq y$. It has been proved in Akers *et al.* (1987) that we can obtain a node $x'$ which is closer to $y$ than to $x$ using the following rules:

**R1**: If $x_1=y_1$, then let $x'=g_i(x)$, where $i$, $2\le i\le n$, is any integer such that $x_i\neq y_i$.

**R2**: If $x_1\neq y_1$, then let $x'=g_i(x)$ such that $x_1=y_i$.

Clearly, to construct a shortest path from $x$ to $y$, we can repeatedly apply **R1** and **R2**. In this paper, we are concerned with a more general *node-to-substar* routing problem as follows: Given any node $x=x_1x_2...x_n$ and any $k$-substar $Y=*^ky_{k+1}y_{k+2}...y_n$ such that $x\notin Y$, how to construct a route from $x$ to *any* node in $Y$ such that the routing distance is minimal. We propose to apply the following rules to obtain a node $x'$:

**R3**: If $x_1\notin \{y_{k+1}, y_{k+2}, ..., y_n\}$, let $x'=g_i(x)$, where $i$ is the smallest integer such that $x_i\in \{y_{k+1}, y_{k+2}, ..., y_n\}$ and $x_i\neq y_i$.

**R4**: If $x_1\in \{y_{k+1}, y_{k+2}, ..., y_n\}$, let $x'=g_i(x)$ such that $x_1=y_i$.

For example, if $x$=123456 and $Y=*^4 63$, then by **R3** the smallest $i$=3. However, if $Y=*^4 41$, then **R4** should be applied and $i$=6. It can be easily proved that at least one symbol between the ($k$+1)-th and the $n$-th positions will be corrected if we continuously apply **R3** and/or **R4** twice. By repeatedly applying the above rules, a path from $x$ to substar $Y$ can be constructed.

## III. All-to-All Personalized Broadcast

### 1. Algorithm

Consider any node $z$. Below, we will first describe how to perform a one-to-all personalized broadcast from $z$. First, we partition $S_n$ into $k$-substars, each with the format $*^kx_{k+1}x_{k+2}...x_n$ (there are totally $N=n!/k!$ such substars). Parameter $k$ is to be determined later for optimization purposes. For each $k$-substar $X$, node $z$ performs the following two steps: (1) $z$ packs the $k!$ personalized messages for the nodes in $X$ into a packet and then sends the packet to a representative node in $X$, and (2) the representative node un-packs the packet and further distributes the messages to each individual node in $X$. Steps (1) and (2) are repeated sequentially $N$ times for each $k$-substar, and the broadcast is completed.

To perform all-to-all personalized broadcast, we execute $n!$ copies of the above one-to-all broadcast from all nodes $z$ in $S_n$. Further, this is done by all $z$'s concurrently. However, note that for each $z$, the corresponding substar $X$ in step (1) may be distinct for different $z$'s. Also, each $z$ must pick a distinct representative node. Consequently, step (2) in fact becomes an all-to-all personalized broadcast in a smaller $k$-substar. Again, steps (1) and (2) will be repeated $N=n!/k!$ times sequentially.

The detailed algorithm for node $z$ is shown below. It is to be executed concurrently and synchronously by all nodes. The label of $z$ is denoted as $z_1z_2...z_n$.

**Table 1.** The Values of Dimensions $d_1, ..., d_j$, Node $y$, and Substar $Y$ when the Algorithm *Personalized_Broadcast*() is Executed by a Node $z$=3241 in an $S_4$ with $k$=2

| $X_i$ | $d_1, ..., d_j$ | $y$ | $Y$ |
|---|---|---|---|
| $X_1$=**12 | 3,2,4 | 1432 | **32 |
| $X_2$=**13 | 3,4 | 1234 | **34 |
| $X_3$=**14 | 3 | 4231 | **31 |
| $X_4$=**21 | 4,2,3 | 4123 | **23 |
| $X_5$=**23 | 2,3,4 | 1324 | **24 |
| $X_6$=**24 | 2,3 | 4321 | **21 |
| $X_7$=**31 | 4 | 1243 | **43 |
| $X_8$=**32 | 2,4 | 1342 | **42 |
| $X_9$=**34 | none | 3241 | **41 |
| $X_{10}$=**41 | 4,3 | 4213 | **13 |
| $X_{11}$=**42 | 2,4,3 | 4312 | **12 |
| $X_{12}$=**43 | 3,4,3 | 3214 | **14 |

```
Algorithm Personalized_Broadcast(z);
  (1) Partition $S_n$ into $N=n!/k!$ $k$-substars, each with
      the format $*^kx_{k+1}x_{k+2}...x_n$. Let these substars be
      denoted as $X_1, X_2, ..., X_N$.
  (2) for $i$=1 to $N$ do
      (i)   Let substar $X_i$ be labeled $*^kx_{k+1}x_{k+2}...x_n$. Use
            the node-to-substar routing rules R3 and R4
            to find a path from node 12...$n$ to substar
            $X_i$. Let the path found follow the sequence
            of dimensions $d_1, d_2, ..., d_j$ ($j$ is the length
            of the path) and lead to a node $x=x_1x_2...x_n$
            in $X_i$.
      (ii)  Let node $y=z_{x_1}z_{x_2}\cdots z_{x_n}$ and substar
            $Y=*^kz_{x_{k+1}}z_{x_{k+2}}\cdots z_{x_n}$.
      (iii) Group the $k!$ personalized messages for nodes
            of $Y$ into a packet and send the packet along
            dimensions $d_1, d_2, ..., d_j$ to the representative
            node $y$.
      (iv)  In substar $Y$, perform an all-to-all person-
            alized broadcast, by which $y$ distributes the
            $k!$ personalized messages it received in step
            (2iii).
```

In step (1), we generate a sequence of substars $X_1, X_2, ..., X_N$ in any order. This is the common counting problem of generating all the permutations of $n-k$ items from $n$ distinct items.

In step (2), note that node $z$ may not send messages to substars $X_1, X_2, ..., X_N$ in that order. Instead, $X_i$ is used in step (2i) in the $i$-th iteration to determine a sequence of dimensions $d_1, d_2, ..., d_j$, which lead from node 12...$n$ to $x$. (Clearly this is independent of the label of $z$.) Then, in step (2ii), a substar $Y$ and a representative node $y$ are determined using the label of $x$ as indices. In step (2iii), $z$ sends a packet following dimensions $d_1, d_2, ..., d_j$ to $y$ (to be proved later).

Finally, in step (2iv), the packet is un-packed in $y$ and further forwarded to the nodes of $Y$. This step can be done using any all-to-all personalized broadcast algorithm.

For example, in Table 1, we show the execution of the algorithm by a node $z$=3241 in an $S_4$ with $k$ set to 2.

The algorithm has the following important properties (with proof provided in parentheses).

(1) *In step (2iii), the routing is congestion-free.* (Recall that the dimensions $d_1$, $d_2$, ..., $d_j$ are independent of the value of $z$. From every source node $z$, the message will synchronously travel first along dimension $d_1$, then along $d_2$, . . ., and then along $d_j$. Thus, in the $i$-th step, every node will send one message and receive one message along dimension $d_i$, $i$=1...$j$. The freedom of congestion then follows.)

(2) *From z, the sequence of dimensions $d_1$, $d_2$, ..., $d_j$ indeed leads to the node y as defined in step (2ii).* (These dimensions lead from node 12...$n$ to node $x=x_1x_2...x_n$. The symbol $x_i$ in $x$ is the $x_i$-th symbol of 12...$n$. Similarly, the symbol $z_{x_i}$ is the $x_i$-th symbol of $z$, which proves the property.)

(3) *In each iteration of step (2), no two nodes z will pick the same representative node y. Thus, step (2iv) is indeed an all-to-all personalized broadcast in substar Y.* (This follows directly from the proof of property 1.)

(4) *In each iteration of step (2ii), node z will pick a distinct substar Y.* (Consider two distinct $k$-substars, $*^k x_{k+1}x_{k+2}...x_n$ and $*^k x'_{k+1}x'_{k+2}...x'_n$, generated in step (1). The corresponding $k$-substars generated in step (2ii) will be $*^k z_{x_{k+1}}z_{x_{k+2}}...z_{x_n}$ and $*^k z_{x'_{k+1}}z_{x'_{k+2}}...z_{x'_n}$, respectively, which must be distinct because there must exist some $i$ such that $x_i \neq x'_i$, $k<i\leq n$.)

From the above, the correctness of *Personalized_ Broadcast*() is readily seen. When $k$ is set to 1, step (2i) is a node-to-node routing, and our algorithm is equivalent to the ones proposed in Fragopoulou and Akl (1995) and Misic and Jovanovic (1994). When $1<k<n$, our algorithm is a recursive one, and we have the freedom of choosing any all-to-all personalized broadcast algorithm in step (2iv). Thus, our algorithm is a generalization of Fragopoulou and Akl (1995) and Misic and Jovanovic (1994).

## 2. Performance Analysis and Comparison

Now we will analyze the performance of our algorithm. In a packet-switching network, sending a packet of $i$ bytes takes $t_s+i\cdot t'_m$ time, where $t_s$ is the start-up time and $t'_m$ is the data transmission time per byte. For notational simplicity, we can let $t_m= p\cdot t'_m$, where $p$ is the size of a personalized message. Thus, step (2iii) takes $j(t_s+k!t_m)$ time, where $j$ is the number of hops. Let us denote by $l_{n,k}$ the average value of $j$ (i.e., the average distance from node 12...$n$ to the $k$-substars $X_1$, $X_2$, ..., $X_N$). The total cost of step (2iii) is $Nl_{n,k}(t_s+k!t_m)$.

Recall that step (2iv) is also an all-to-all personalized broadcast in a smaller substar. We will call the algorithm in Fragopoulou and Akl (1995) and Misic and Jovanovic (1994) in step (2iv) (or equivalently call our algorithm again by setting parameter $k$ to 1) and evaluate the performance for various values of $n$ and $k$. Therefore, node y needs to send $k!-1$ messages one by one to the other $k!-1$ nodes in $Y$. This gives a cost of $k!l_{k,1}(t_s+t_m)$. Note that $l_{n,1}$ is the average distance from node 12...$k$ to every node (including itself) in $S_k$[1]. Counting the outer for-loop, the total cost of step (2iv) is $n!l_{k,1}(t_s+t_m)$. Summing the above factors, we have the cost of the algorithm:

$$T(n,k)=(Nl_{n,k}+n!l_{k,1})t_s+(n!l_{n,k}+n!l_{n,1})t_m. \quad \textbf{(1)}$$

When $k$=1, the cost is reduced to $T(n,1)=n!l_{n,1}(t_s+t_m)$, which is exactly the time required by Fragopoulou and Akl (1995) and Misic and Jovanovic (1994).

To compare the performance of our algorithm with that of Fragopoulou and Akl (1995) and Misic and Jovanovic (1994), we would expect the condition $T(n,1)>T(n,k)$ to hold. We derive the inequality as follows:

$$T(n,1)>T(n,k)$$

$$\Leftrightarrow n!l_{n,1}(t_s+t_m)>(Nl_{n,k}+n!l_{k,1})t_s+(n!l_{n,k}+n!l_{k,1})t_m$$

$$\Leftrightarrow \frac{t_s}{t_m} > \frac{l_{n,k}+l_{k,1}-l_{n,1}}{l_{n,1}-l_{n,k}/k!-l_{k,1}}. \quad \textbf{(2)}$$

There are three factors which affect the above inequality: $n$, $k$, and the ratio $t_s/t_m$. Much to our surprise, as shown below, the value of $T(n,1)$ is always greater than $T(n,2)$.

**Lemma 1.** *For any $n\geq 2$, $l_{n,1}-l_{n,2}$=0.5.*

**Proof.** To calculate $l_{n,2}$, we will partition $S_n$ into 2-substars. Consider any 2-substar $X$. Let $x_1$ and $x_2$ be the two nodes of $X$. Suppose the minimum distance from 12...$n$ to $x_1$ and $x_2$ is $d$. Assume without loss of

[1]It can be derived that $l_{k,1}=(k+\frac{2}{k}+H_k-4)\frac{k!-1}{k!}$, where $H_k=\sum_{i=1}^{k}\frac{1}{i}$. See Akers *et al.* (1987) for details.

**Table 2.** The Minimum Value of $t_s/t_m$ Such That $T(n,1)>T(n,k)$ for Various $n$ and $k \geq 3$

| | | *n* | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | $t_s/t_m>$ | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 |
| | 3 | 0.190 | 0.150 | 0.124 | 0.106 | 0.092 | 0.082 | 0.074 | 0.067 | 0.062 |
| | 4 | | 0.288 | 0.239 | 0.205 | 0.179 | 0.160 | 0.144 | 0.131 | 0.120 |
| | 5 | | | 0.367 | 0.315 | 0.276 | 0.246 | 0.222 | 0.202 | 0.186 |
| | 6 | | | | 0.435 | 0.382 | 0.340 | 0.307 | 0.280 | 0.257 |
| *k* | 7 | | | | | 0.491 | 0.438 | 0.395 | 0.360 | 0.331 |
| | 8 | | | | | | 0.538 | 0.485 | 0.442 | 0.406 |
| | 9 | | | | | | | 0.577 | 0.526 | 0.483 |
| | 10 | | | | | | | | 0.610 | 0.560 |
| | 11 | | | | | | | | | 0.638 |

generality that the distance from $12...n$ to $x_1$ is $d$. As the distance between $x_1$ and $x_2$ is one, it is not hard to see that the distance between $12...n$ and $x_2$ can only be $d$ or $d$+1. However, the distance $d$ is impossible because this implies that we can construct a cycle from $12...n$ to $x_1$, then to $x_2$, and then back to $12...n$, of length $2d+1$. This is a contradiction because $S_n$ is known to be bipartite (Akers and Krishnamurthy, 1989). Therefore, we conclude that the average distance from $12...n$ to $x_1$ and $x_2$ is $d$+0.5. Since $l_{n,1}$ means the average distance from $12...n$ to all other nodes, the lemma then follows. ■

**Theorem 1.** *For any* $n \geq 3$, $T(n,1)>T(n,2)$.

**Proof.** Observe that $l_{2,1}$=0.5. Consider the dividend of the expression on the right-hand size of Eq. (2). By Lemma 1, the expression $l_{n,2}+l_{2,1}-l_{n,1}=0$. Applying this value to Eq. (2), we have the inequality $t_s/t_m> \frac{l_{n,k}+l_{k,1}-l_{n,1}}{l_{n,1}-l_{n,k}/k!-l_{k,1}}=0$, which must be true as $t_s$ and $t_m$ are positive non-zero reals. So Eq. (2) concludes that $T(n,1)>T(n,2)$. Note that the inequality does not hold for $n$=2 because the divisor will be 0. ■

The above theorem indicates that our algorithm is always better than that in Fragopoulou and Akl (1995) and Misic and Jovanovic (1994) if we set $k$ to 2. For other $k$'s, Eq. (2) indicates that the larger the ratio $t_s/t_m$ is, the more likely $T(n,1)>T(n,k)$ will hold. In Table 2, we calculate the minimum value of $t_s/t_m$ for various $n$ and $k$ such that $T(n,1)>T(n,k)$ holds. It can be seen that these values are all very small (<0.64). As long as the ratio $t_s/t_m$ is larger than these values, out algorithm performs better.

Given an $n$, it is also desirable to know which value of $k$ will give the best performance. To find this out, we define the *execution time improved (ETI)* as

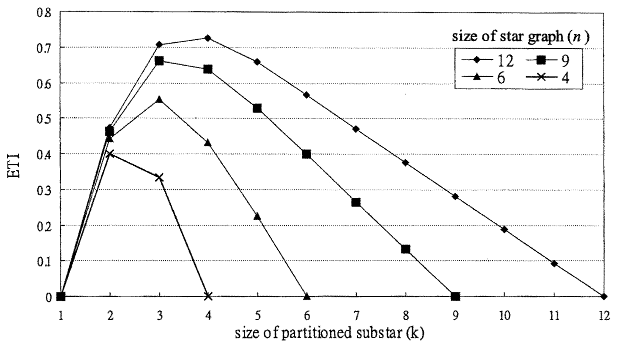

**Fig. 2.** The *ETI* of *Personalized_Broadcast*() under various $n$ and $k$ when $t_s/t_m$=100.

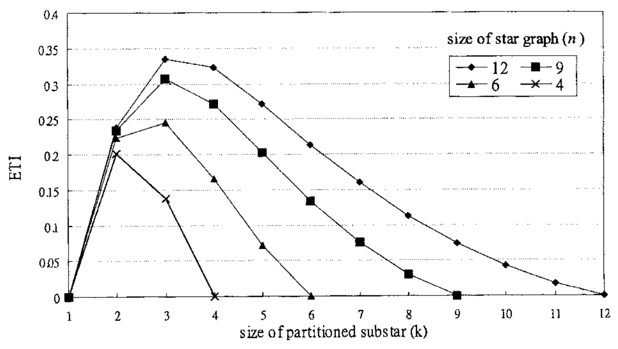

**Fig. 3.** The *ETI* of *Personalized_Broadcast*() under various $n$ and $k$ when $t_s/t_m$=1.

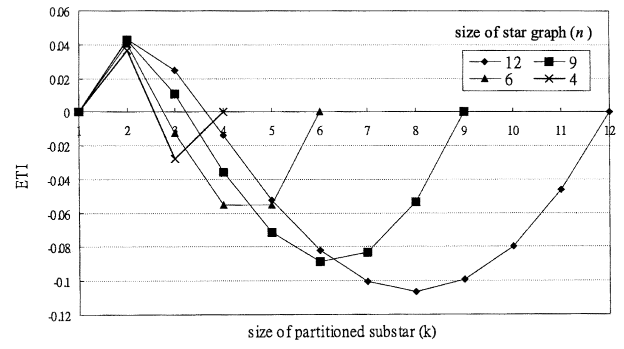

**Fig. 4.** The *ETI* of *Personalized_Broadcast*() under various $n$ and $k$ when $t_s/t_m$=0.1.

follows:

$$ETI = \frac{T(n,1)-T(n,k)}{T(n,1)}. \tag{3}$$

*ETI* indicates the execution time that is saved as opposed to the that required by the case of $k$=1 (or equivalently, the time required by Fragopoulou and Akl (1995) and Misic and Jovanovic (1994). The larger *ETI* is, the better. We have conducted various simulations based on the assumption that the ratios $t_s/t_m$ are large (=100), medium (=1), or small (=0.1). The results are shown in Fig. 2, Fig. 3 and Fig. 4, respectively. From these figures, we find that the best values of $k$ all range between 2 to 4 for all $n \le 12$ (a larger $n$ may not be practical as an $S_{12}$ already has $4.8 \times 10^8$ nodes). The range of improvement is between 0.2 to 0.8, depending on the ratio $t_s/t_m$. Hence, to get the best performance for our algorithm, a packet (in step (2iii)) only needs to contain 2!, 3!, or 4! messages, which is feasible in most current technologies.

## IV. All-to-All Non-personalized Broadcast

### 1. Algorithm

The algorithm is based on finding an embedding of an $n \times (n-1)!$ mesh in the $S_n$. The mesh in fact has wrap-around connections along the second dimension. The embedding has a dilation of 3. We first derive the embedding.

**Definition 1.** Given any node $v$ and any two symbols $x$ and $y$, the function $T_{x,y}(v)$ is defined as the node obtained from $v$ by swapping the symbols $x$ and $y$ in $v$. (For instance, in $S_5$, $T_{2,4}(51234)=51432$.)

**Lemma 2.** *Consider two* $(n-1)$*-substars* $*^{n-1}x$ and $*^{n-1}y$. *Let* $v$ *and* $v'$ *be two nodes in* $*^{n-1}x$ *such that* $v'=g_i(v)$, $2 \le i \le n-1$. *Then* $T_{x,y}(v)$ *and* $T_{x,y}(v')$ *are two nodes in* $*^{n-1}y$ *such that* $T_{x,y}(v')=g_i(T_{x,y}(v))$.

**Proof.** Let $v$ be labeled $a...b...x$, where $a$ and $b$ are the first and $i$-th symbols, respectively. Let node $v'$ be labeled $b...a...x$. We consider the location of symbol $y$ in $v$ in three cases: (1) $y=a$, (2) $y=b$, and (3) otherwise. One can easily verify for each case that $T_{x,y}(v)$ and $T_{x,y}(v')$ are two nodes in $*^{n-1}y$ connected by an edge along dimension $i$. ■

**Corollary 1.** *Let* $v_1, v_2, ..., v_p$ *be a ring in substar* $*^{n-1}x$. *Then,* $T_{x,y}(v_1), T_{x,y}(v_2), ..., T_{x,y}(v_p)$ *form a ring in substar* $*^{n-1}y$.

**Proof.** First, note that if $v_i$ and $v_j$, $1 \le i,j \le p$, are two distinct nodes, then $T_{x,y}(v_i)$ and $T_{x,y}(v_j)$ must be distinct. The corollary then follows directly from Lemma 2. ■

To embed an $n \times (n-1)!$ mesh, we can first construct a ring of length $(n-1)!$ in substar $*^{n-1}1$. This is possible because a star graph is Hamiltonian (Nigam *et al.*, 1990; Tseng *et al.*, 1997). Let's call this ring

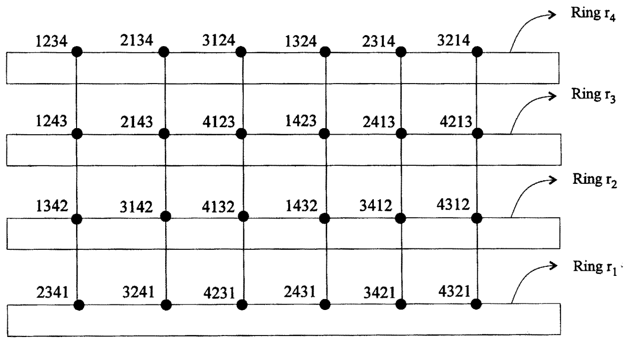

**Fig. 5.** The embedding of a $4 \times 3!$ mesh embedded in an $S_4$.

$r_1$. From $r_1$, by Corollary 1, we can construct a ring $r_2$ in substar $*^{n-1}2$ using function $T_{1,2}$. We can repeatedly generate a ring $r_i$ in substar $*^{n-1}i$ from $r_{i-1}$ using the operator $T_{i-1,i}$ for $i$=3...$n$. By regarding each $x \in r_{i-1}$ and $T_{i-1,i}(x) \in r_i$ as two mesh nodes connected by a mesh edge, we already obtain an $n \times (n-1)!$ mesh. For instance, Fig. 5 shows a $4 \times 3!$ mesh embedded in an $S_4$. We implement operator $T_{i-1,i}$ using either one or three edges as follows. Given a node $v=v_1v_2...v_{n-1}(i-1)$, define

$$T_{i-1,i}(v) = \begin{cases} g_n(v) & \text{if } i = v_1 \\ g_n(g_x(g_n(v))) & \text{if } (i = v_x) \wedge (2 \le x \le n-1) \end{cases} \qquad \textbf{(4)}$$

Hence, the embedding incurs a dilation of 3 for edges along the first dimension. Edges along the second dimension have no dilation.

Next, we derive our algorithm using the mesh. The broadcast is performed in two stages: *column-exchange* and *row-exchange*, where a column has $n$ mesh nodes and a row $(n-1)!$ mesh nodes. In the column-exchange stage, every set of $n$ nodes in the same column will exchange their messages. Then, in the row-exchange stage, each node will propagate $n$ messages (its own message plus the $n-1$ messages it received in the previous stage) to other nodes in the same row. Note that the column-exchange is a broadcast in a linear array of length $n$ while the row-exchange is one in a ring of length $(n-1)!$. The complete algorithm is derived in 3 steps as follows.

**Algorithm** *Nonpersonalized_Broadcast*();

(1) Along each column, each node sends its message, following the embedding, to the next node in the positive direction. A node receiving a message further propagates the message to the next node in the same direction. The propagation is repeated $n-1$ times and is performed in a

**Table 3.** The Minimum Value of $t_s/t_m$ Such That $T'>T$ in *Nonpersonalized_Broadcast*()

| $n$ | 5 | 6 | 7 | 8 | 9 | 10 | 12 | 14 | 16 | 18 | 20 |
|---|---|---|---|---|---|---|---|---|---|---|---|
| $t_s/t_m>$ | .27 | .043 | $7.0\times10^{-3}$ | $9.9\times10^{-4}$ | $1.2\times10^{-4}$ | $1.3\times10^{-5}$ | $1.2\times10^{-7}$ | $8.0\times10^{-10}$ | $3.8\times10^{-12}$ | $1.4\times10^{-14}$ | $4.1\times10^{-17}$ |

synchronous manner (by synchronous, a $T_{i-1,i}$ that traverses only 1 edge must wait for the others that need to traverse 3 edges to finish before the next propagation can be started).

(2) Similar to step (1), but the propagation proceeds in the negative direction.

(3) In each ring $r_i$, each node packs the $n$ messages it received in steps (1) and (2) into one packet and sends it along the positive direction. All other nodes in $r_i$ help propagate the packet in the same direction. The propagation is repeated $(n-1)!-1$ times in a synchronous manner.

Steps (1) and (2) implement the column-exchange stage, while step (3) does the row-exchange. The proof of correctness is trivial. It is also clear that step (3) is congestion-free. Below, we will further prove that the communications incurred by $T_{1,2}$, $T_{2,3}$, ..., $T_{n-1,n}$ together in steps (1) and (2) are free from congestion.

In step (1), in each propagation, the operator $T_{i-1,i}$ is equal to $g_n g_x g_n$ or $g_n$ (recall Eq. (4)). Below, we refer to these as the first, second (if any), and third (if any) traversals. We claim that in each traversal, *each node in the network will receive at most one message from other nodes*. If so, the communication is congestion-free since a node has at most one message to be forwarded in the next traversal. At the beginning, each node has one message to be sent. In the first traversal, each node will send, and simultaneously receive, one message along dimension $n$. So the above claim holds. In the second traversal, the communications all proceed from nodes with the format $(i-1)*^{n-1}$ to nodes with the format $i*^{n-1}$ for $i$=2...$n$. As a node with the format $i*^{n-1}$ can not be adjacent to two nodes with the format $(i-1)*^{n-1}$, each node will receive at most one message in this traversal. So the claim still holds. Finally, in the third traversal, a node may only send (and thus receive) one message along dimension $n$. The claim is clearly true. Step (2) is congestion-free by a proof similar to that above.

## 2. Performance Analysis and Comparison

Each of step (1) and step (2) will execute the operation $T_{i-1,i}$ $n-1$ times. Each operation is implemented by at most 3 traversals. As each time one message is sent, the total time required is $6(n-1)(t_s+t_m)$, where $t_m$ is the time required to transmit a broadcast message along a channel (not including the start-up time $t_s$). In step (3), each packet will travel along some $r_i$ for $(n-1)!-1$ steps. As a packet consists of $n$ messages, the total cost is $((n-1)!-1)(t_s+nt_m)$. Overall, the time taken by the algorithm can be calculated as follows:

$$T=((n-1)!+6n-7)t_s+(n!+5n-6)t_m.$$

We compare our algorithm against the one given in Fragopoulou and Akl (1995), where it was suggested that a Hamiltonian cycle in $S_n$ be used, along which the messages are propagated $n!-1$ times (which is similar to our step (3)). The cost of Fragopoulou and Akl (1995) is easily seen to be $T'=(n!-1)(t_s+t_m)$. Again, we expect the condition $T'>T$ to hold, which is equivalent to:

$$\begin{aligned}\frac{t_s}{t_m} &> \frac{5n-5}{n!-(n-1)!-6n+6}\\ &= \frac{5}{(n-1)!-6}\\ &= O\left(\frac{1}{(n-1)!}\right). \qquad (5)\end{aligned}$$

In Table 3, we calculate the minimum values of $t_s/t_m$ such that $T'>T$ holds for various $n$. As can be seen, this value ≈0.27 when $n$=5 and drops quickly nearly zero as $n$ increases. As long as $t_s/t_m$ is larger than this value, our algorithm performs better. In current tech-

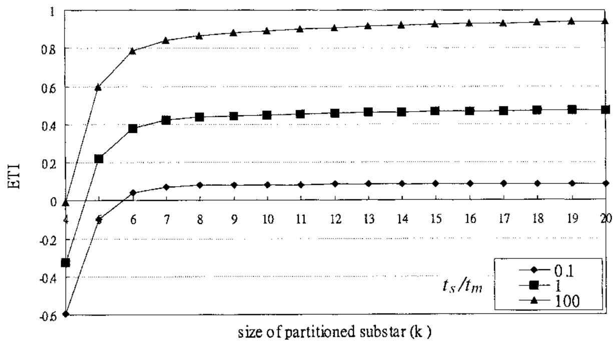

**Fig. 6.** The *ETI* of *Nonpersonalized_Broadcast*() under various $n$ when $t_s/t_m$=0.1, 1, and 100.

nologies, the start-up time is typically a large factor. So our algorithm should have very broad applicability.

Finally, we estimate the amount of improvement over Fragopoulou and Akl (1995). Again, we define *ETI* as follows:

$$ETI = \frac{T' - T}{T'} .$$

In Fig. 6, we plot *ETI* against $n$ when the ratio $t_s/t_m$ is large (=100), medium (=1), and small (=0.1). Only when $n$ is very small, will our algorithm result in no improvement. In most cases, the amount of improvement is significant (approximately 0.8, 0.4, and 0.1 when $t_s/t_m$ is large, medium, and small, respectively).

## V. Conclusions

In this paper, we have presented efficient all-to-all personalized and non-personalized broadcast algorithms. These algorithms rely on a node's capability of packing several messages together into a packet for delivery. The numbers of messages packed in a packet are fairly small, but the amount of improvement achieved over the existing algorithms has been shown to be significant. Therefore, the results have much practical value.

### Acknowledgment

This research was supported by the National Science Council of the Republic of China under Grant # NSC 86-2213-E-008-029 and Grant # NSC 86-2213-E-216-021.

### References

Akers, S. B. and B. Krishnamurthy (1989) Group-theoretic model for symmetric interconnection networks. *IEEE Trans. on Comput.*, **38**(4), 555-565.

Akers, S. B., D. Harel, and B. Krishnameurthy (1987) The star graph: an attractive alternative to the *n*-cube. *Int'l Conf. on Parallel Processing*, pp. 393-400. Chicago, IL, U.S.A.

Akl, S. G., K. Qiu, and I. Stojmenovic (1993) Fundamental algorithms for the star and pancake interconnection networks with applications to computational geometry. *Networks*, **23**(4), 215-225.

Bagherzadeh, N., N. Nassif, and S. Latifi (1993) A routing and broadcasting scheme on faulty star graphs. *IEEE Trans. on Comput.*, **42**(11), 1398-1403.

Chen, T. S., Y. C. Tseng, and J. P. Sheu (1996) Balanced spanning trees in complete and incomplete star graphs. *IEEE Trans. on Paral. and Distrib. Sys.*, **7**(7), 717-723.

Day, K. and A. Tripathi (1994) A comparative study of topological properties of hypercubes and star graphs. *IEEE Trans. on Paral. and Distrib. Sys.*, **5**(1), 31-38.

Fragopoulou, P. and S. G. Akl (1995) Optimal communication algorithms on star graphs using spanning tree constructions. *J. of Parallel and Distrib. Comput.*, **24**, 55-71.

Jovanovic, Z and J. Misic (1994) Fault tolerance of the star graph interconnection network. *Inf. Process. Lett.*, **49**, 145-150.

Jwo, J. S., S. Lakshmivarahan, and S. K. Dhall (1990) Embeddings of cycles and grids in star graphs. *Symp. on Parallel and Distrib. Processing*, pp. 540-547. Dallas, TX, U.S.A.

Latifi, S. (1993) On the fault-diameter of the star graph. *Inf. Process. Lett.*, **46**, 143-150.

Mendia, V. E. and D. Sarkar (1992) Optimal broadcasting on the star graph. *IEEE Trans. on Paral. and Distrib. Sys.*, **3**(4), 389-396.

Misic, J. and Z. Jovanovic (1994) Communication aspects of the star graph interconnection network. *IEEE Trans. on Paral. and Distrib. Sys.*, **5**(7), 678-687.

Nigam, M., S. Sahni, and B. Krishnamurthy (1990) Embedding hamiltonians and hypercubes in star interconnection graphs. *Int'l Conf. on Parallel Processing*, pp. III-340-III-343. Chicago, IL, U.S.A.

Qiu, K. (1995) Broadcasting on the star and pancake interconnection networks. *Int'l Parallel Processing Symp.*, pp. 660-665. Santa Barbara, CA, U.S.A.

Qiu, K., S. G. Akl, and H. Meijer (1994) On some properties and algorithms for the star and pancake interconnection networks. *J. of Parallel and Distrib. Comput.*, **22**, 16-25.

Sheu, J. P., W. H. Liaw, and T. S. Chen (1993) A broadcasting algorithm in star graph interconnection networks. *lnf. Process. Lett.*, **48**, 237-241.

Sheu, J. P., C. T. Wu, and T. S. Chen (1995) An optimal broadcasting algorithm without message redundancy in star graphs. *IEEE Trans. on Paral. and Distrib. Sys.*, **6**(6), 653-658.

Tseng, Y. C., S. H. Chang, and J. P. Sheu (1997) Fault-tolerant ring embedding in a star graph with both link and node failures. *IEEE Trans. on Paral. and Distrib. Sys.*, **8**(12), 1185-1195.

# 在星狀圖網路中有效率的多對多廣播

曾煜棋　許健平　章戊霖

國立中央大學資訊工程學系

## 摘　要

在本文中，我們探討在星狀圖網路中的二種通訊問題：『多對多個人化』和『多對多非個人化』廣播。前人之研究僅將資料傳輸時間（data transmission time）予以最佳化，但忽略了準備資料傳輸的啟動時間（start-up time）。因此，前人之結果雖然宣稱為最佳解，其實僅在啟動時間可以忽略的情況下成立。本文中，我們考量將啟動時間和傳輸時間同時予以最佳化。我們發展的多對多個人化廣播演算法較前人之結果更有效率。我們發展的多對多非個人化廣播演算法也優於前人之結果，除非啟動時間相較於傳輸時間極小的情形之外（其比例約為$O(1/(n-1)!)$，其中$n$為星狀圖之維度）。模擬結果顯示我們的結果相較於前人之結果約有10%~80%之改進。