# Broadcasting Multiple Messages in the Multiport Model

Amotz Bar-Noy, *Senior Member, IEEE,* and Ching-Tien Ho, *Member, IEEE*

**Abstract**—We consider the problem of broadcasting multiple messages from one processor to many processors in the $k$-port model for message-passing systems. In such systems, processors communicate in rounds, where in every round, each processor can send $k$ messages to $k$ processors and receive $k$ messages from $k$ processors. In this paper, we first present a simple and practical algorithm based on variations of $k$ complete $k$-ary trees. We then present an optimal algorithm up to an additive term of one for this problem for any number of processors, any number of messages, and any value for $k$.

**Index Terms**—Broadcast, one-to-all broadcast, multiport model, message-passing system, collective communication.

## 1 Introduction

This paper explores the broadcast problem in the *multiport* model for message-passing systems. In particular, we consider (one-to-all) broadcast problem on a message-passing system modeled by a complete graph of $n$ nodes with $k$-port model. We assume that there are $n$ processors (nodes) in the system, denoted by $0, 1, \ldots, n-1$, where the source of the broadcast (the *broadcaster*) is processor $0$. We also assume that the source has $m$ messages, denoted by $M_1, M_2, \ldots, M_m$, to broadcast to all the other processors. In the $k$-port model, each of the $n$ processors has $k$ distinct input ports and $k$ distinct output ports. In each communication round, every processor can send $k$ distinct messages to $k$ other processors and in the same round each processor can receive $k$ distinct messages that were sent out from $k$ other processors.

Broadcasting is an important communication operation in many multiprocessor systems. Application domains that use this operation extensively include scientific computations, network management protocols, database transactions, and multimedia applications. Due to the significance of this operation it is important to design efficient algorithms for it. The broadcasting operation is frequently used in many applications for message-passing systems (see [11]). Several collective communication libraries, such as Express [9] by Parasoft and the Message Passing Library (MPL) [1], [2] of IBM SP2 parallel systems, provide the broadcast primitive. This operation has also been included as part of the collective communication routines in the Message-Passing Interface (MPI) standard proposal [8].

Several variations of the broadcasting problem were studied in the literature. (See [13] for a comprehensive survey.) Most of this research focused on designing broadcasting algorithms for specific network topologies such as rings, trees, meshes, and hypercubes. However, an emerging trend in many communication systems is to treat the system as a fully connected collection of processors in which every pair of processors can communicate directly. This trend can be identified in a number of modern multiprocessor systems, such as IBM's Vulcan [17], Thinking Machines' CM-5 [15], NCUBE's nCUBE/2 [16], Intel's Paragon [12], and IBM's SP2, as well as in some high-speed communication networks (e.g., PARIS [6]).

When communicating large amounts of data, many systems break the data into sequences of messages (or packets) that are sent and received individually. This approach motivates research into the problem of how to disseminate multiple messages efficiently in such systems. Here, we focus on the problem of broadcasting multiple messages from one source. (Broadcasting a single message in our model is a simpler task.)

The problem of broadcasting multiple messages in fully connected systems was studied in several communication models. Cockayne and Thomason [7] and Farley [10] presented optimal-time solutions for this problem in a model in which each processor can either send one message *or* receive one message in any communication round, but not both. (This model is sometimes referred to as the unidirectional telephone model or the telegraph model.) In this model, the optimal number of rounds for odd $n$ is $2m - 1 + \lfloor \log n \rfloor$ and the optimal number of rounds for even $n$ is $2m + \lfloor \log n \rfloor - \lfloor \frac{m-1+2^{\lceil \log n \rceil}}{n/2} \rfloor$. In the bidirectional telephone model, Bar-Noy et al. [5] provided an optimal algorithm that requires $(m-1) + \lceil \log n \rceil$ rounds for even $n$. For odd $n$, they presented an algorithm that is optimal up to an additive term of 1 and requires $(m-1) + \lceil \log n \rceil + \frac{m}{n-1} + c$ rounds. They also solved the broadcasting problem optimally in the simultaneous send/receive model. In this model, in every round, each processor can send a message to one processor and receive a message from another. (Note that the send/receive model is equivalent to the one-port model.) Their solution requires $(m-1) + \lceil \log n \rceil$ rounds.

- *A. Bar-Noy is with Tel Aviv University, Ramat Aviv Tel Aviv, 69978, Israel. E-mail: amotz@eng.tau.ac.il.*
- *C.-T. Ho is with IBM Almaden Research Center, San Jose, CA 95120. E-mail: ho@almaden.ibm.com.*

*Manuscript received 12 Feb. 1996; revised 8 July 1997.*
*For information on obtaining reprints of this article, please send e-mail to: tpds@computer.org, and reference IEEECS Log Number 100116.*

Bar-Noy and Kipnis [3], [4] as well as Karp et al. [14] also investigated the problem of broadcasting multiple messages in the Postal and LogP models of communication. In these models, each processor can simultaneously send one message and receive another message, but message delivery involves some communication latency. In these models, no optimal solutions for the problem of broadcasting multiple messages are known for nontrivial values of the communication latency.

The multiport model generalizes the one-port model that has been widely investigated. There are examples of parallel systems with $k$-port capabilities for $k > 1$, such as the nCUBE/2 [16], the CM-2 (where $k$ is the dimension of the hypercube in both machines), and transputer-based machines.

### 1.1 Our Results

In this paper, we present two algorithms for broadcasting $m$ messages within an $n$-node complete graph in the $k$-port model for any $n$ and any $k \geq 2$. The first algorithm, called the $k$-tree algorithm, is very simple and practical. It has a time complexity of $\lceil m/k \rceil + \max(2, \lceil \log_k(n+2k) \rceil)$, compared to a simple lower bound of $\lceil m/k \rceil + \lceil \log_{k+1} n \rceil - 1$. Thus, the delay of each message is optimal up to a small multiplicative factor of $\log(k+1)/\log k$. The second algorithm, called the rotation algorithm, is optimal up to an additive term of one. Specifically, our algorithm requires $\lceil m/k \rceil + \lceil \log_{k+1} n \rceil$ rounds. The second algorithm is more complicated and uses the first algorithm as a subroutine. Throughout the paper, we assume $k \geq 2$.

## 2 Some Bounds

In this section, we present some bounds regarding the multiple messages broadcasting problem. The first two lemmas are simple extensions of the well-known lower bound for the one-port case. The first observation is that the broadcasting time of a single message among $n$ processors in the $k$-port model must take at least $\lceil \log_{k+1} n \rceil$ rounds. This is because after one round at most $k+1$ processors know the message, after two rounds at most $(k+1)^2$ know the message, etc.

**Lemma 1.** *The broadcasting time of one message among $n$ processors in the $k$-port model is at least $\lceil \log_{k+1} n \rceil$ rounds.*

Our second observation is that the earliest round the broadcaster can send the $m$th message is after round $\lceil m/k \rceil - 1$ since, in each round, it can send at most $k$ messages. Thus, the simple lower bound follows:

**Lemma 2.** *The broadcasting time of $m$ messages among $n$ processors in the $k$-port model is at least $\lceil \frac{m}{k} \rceil - 1 + \lceil \log_{k+1} n \rceil$ rounds.*

However, for many combinations of $n$, $m$, and $k$, we have a lower bound which is one larger than the previous lower bound. The broadcast time of our second algorithm is also one larger than the previous lower bound. Our second lower bound shows that in many cases this algorithm is optimal. Specifically, we have the following lemma:

**Lemma 3.** *Let $n' = (k+1)^{\lceil \log_{k+1} n \rceil}$ and let*

$$\beta = ((m-1) \bmod k) + 1.$$

*If $(n-1)\beta > n' - 1$, then the lower bound for broadcasting $m$ messages among $n$ processors in the $k$-port model is*

$$\left\lceil \frac{m}{k} \right\rceil + \lceil \log_{k+1} n \rceil.$$

**Proof.** Following the proof of Lemma 2, the broadcaster needs at least $t = \lceil m/k \rceil$ rounds to send out all $m$ messages. Furthermore, in the tightest schedule with respect to the broadcaster, there are $\beta$ messages (where $1 \leq \beta \leq k$) need to be sent out at round $t$. For these $\beta$ messages to reach all other $n-1$ processors, a total "bandwidth" of $(n-1)\beta$ is needed starting from round $t$. However, the maximum bandwidth that can be used for these $\beta$ messages, starting from round $t$, are $k, (k+1)k, (k+1)^2 k, \cdots$. That is, at round $t + \lceil \log_{k+1} n \rceil - 1$, a total of $n' - 1$ bandwidth can be used for these $\beta$ messages. Thus, if $(n-1)\beta > n' - 1$, at least one more round is needed. □

For example, when $k = 3$ and $m \bmod k = 2$, the lower bound is $\lceil m/k \rceil + \lceil \log_{k+1} n \rceil$ for $33 \leq n \leq 64$ among all $n$ in the range of $16 < n \leq 64$. Also, when $k = 3$ and $m \bmod k = 0$ (i.e., $\beta = 3$), the lower bound is $\lceil m/k \rceil + \lceil \log_{k+1} n \rceil$ for $23 \leq n \leq 64$ among all $n$ in the range of $16 < n \leq 64$. As a special case when $n$ is a power of $k+1$, we have the following corollary:

**Corollary 4.** *When $n$ is a power of $k+1$, the number of rounds required in broadcasting $m$ messages among $n$ processors in a $k$-port model, $k \geq 2$, is at least*

$$\begin{cases} \lceil \frac{m}{k} \rceil + \log_{k+1} n - 1, & if\ m \bmod k = 1, \\ \lceil \frac{m}{k} \rceil + \log_{k+1} n, & otherwise. \end{cases}$$

Note that the $-1$ term appears only in the case where, in the last round of sending by the broadcaster, the broadcaster has only one message to send. Only in such cases the broadcaster could start a complete broadcasting tree.

In this paper, we circumvent this distinction between different values for $m$ and $k$ by considering the *minimum broadcasting time* version of the problem. In this version, we assume that the broadcaster has an infinite number of messages and in each round it sends $k$ new messages to some $k$ or less processors. These processors are responsible for broadcasting these messages among the rest of the processors. The goal is to minimize the broadcasting time of all messages. If we show that the maximum broadcasting time of any message is $T$ rounds, then the broadcasting time for $m$ messages can be achieved in $\lceil m/k \rceil - 1 + T$ rounds simply by instructing the broadcaster to be idle after it finishes sending all the $m$ messages. Such a reduction yields an algorithm which is optimal up to an additive term of $T - \lceil \log_{k+1} n \rceil$ from the optimum. We summarize the above discussion in the following lemma.

**Lemma 5.** *Suppose that there exists an algorithm for the minimum broadcasting time problem the complexity of which is T rounds. Then there exists an algorithm for the multiple messages broadcasting the complexity of which is far from the optimum by an additive term of at most $T - \lceil \log_{k+1} n \rceil$ rounds.*

## 3 The $k$-Tree Algorithm

In this section, we describe a very simple algorithm, called the $k$-tree algorithm. The time complexity of the algorithm is $\lceil m/k \rceil + \max(2, \lceil \log_k(n+2k) \rceil)$. Thus, the delay of each message is optimal up to a small multiplicative factor of $\log(k+1)/\log k$.

### 3.1 A General $k$-Tree Theorem

Our algorithm is based on a construction of $k$ spanning trees of size $n$ each and a proper labeling of the tree nodes. We first give a general theorem regarding $k$-port broadcast based on $k$ spanning trees.

**Theorem 6.** *If one can construct $k$ spanning trees (of size $n$ each) with the properties that*

1. *for each tree, the $n$ nodes are uniquely labeled from 0 through $n-1$, and the root is labeled 0 (the broadcaster),*
2. *the height of any tree is less than $h$, and*
3. *for each node (identified by a label) the number of children summing over all $k$ trees is at most $k$,*

*then broadcasting $m$ messages among $n$ nodes in the $k$-port model can be finished in time $\lceil m/k \rceil + h - 1$.*

**Proof.** For each round, the root (broadcaster) can send out a distinct message in each tree. The messages are propagated down the tree with pipelining one level down per round. To make sure such scheduling does not violate the $k$-port model, the number of incoming messages (and outgoing messages, respectively) per round for each node must not exceed $k$. Clearly, each nonroot node will receive at most $k$ messages per round because it has one parent per tree. Since, by Property 3, each node has at most $k$ children summing over all $k$ trees, the number of outgoing messages per round is also bounded from above by $k$. The time complexity then follows from Property 2. □

### 3.2 Almost Complete $k$-ary Trees

The following definition is needed for our algorithm:

**Definition 1.** *An* almost complete $k$-ary tree *of $n$ nodes, denoted $T_k(n)$, can be constructed as follows: Start from the root by adding nodes level by level in a top-down manner. Within each level $l$, $k$ leaf nodes are attached to each node of the level $l-1$ from left to right until either all nodes at this level have been filled or the tree has reached a total of $n$ nodes.*

We say that a node in a tree is an *internal* node if it is neither the root of the tree nor a leaf node. Also, the root and the internal nodes are jointly referred to as *nonleaf* nodes. Clearly, all the nonleaf nodes in $T_k(n)$ have $k$ children except for the last nonleaf node which has $(n-1) \bmod k$ children. Also, only the last two levels can have leaf nodes. Since in a complete $k$-ary tree of height $h$ there are $n = \sum_{i=0}^{h} k^i = \frac{k^{h+1}-1}{k-1}$ nodes, it follows that $h = \log_k(nk - n + 1) - 1$ in such trees. Consequently, for other values of $n$ the height of $T_k(n)$ can be derived as $h = \lceil \log_k(nk - n + 1) \rceil - 1$.

For convenience, we will also define $T'_k(n)$ as a tree which is derived by attaching the root of a $T_k(n-1)$ to a new node, serving as the new root of $T'_k(n)$. We will broadcast based on $k$ spanning trees, where each tree has the topology of $T'_k(n)$ possibly with some minor modification. The goal is to find a mapping of $\{0, 1, \cdots, n-1\}$ to each tree with node 0 mapped to the root, such that Property 3 of Theorem 6 is satisfied. We consider three cases separately in the following: 1) $k$ divides $n-2$ and $n \geq k+2$, 2) $k$ does not divide $n-2$ and $n \geq k+2$, and 3) $n < k+2$.

### 3.3 $k$ Divides $n-2$ and $n \geq k+2$

We use $k$ spanning trees for broadcasting, each of topology $T'_k(n)$. In a complete $k$-ary tree with $n$ nodes, all the nodes can be counted in groups of $k$ except of the root. Therefore, in a complete $k$-ary tree $k$ divides $n-1$. Since $T'_k(n)$ is composed of $T_k(n-1)$ trees, it follows that when $k$ divides $n-2$, every internal node in each of the $T_k(n-1)$ tree has *full* fanout, i.e., $k$ children. Hence, in this case, the number of internal nodes per tree is $(n-2)/k$ and there are a total of $n-2$ internal nodes over all $k$ trees. Note that the broadcaster is the root of each of the $k$ trees. For all the other $n-1$ processors, we can choose $n-2$ of them and define a one-to-one mapping to the $n-2$ internal nodes. Since each processor is mapped to an *internal* node at most once (i.e., it is mapped to leaf nodes in all other trees), it has at most $k$ children summing over all $k$ trees. Thus, by Theorem 6, the algorithm finishes in time $\lceil m/k \rceil + h - 1$, where $h$ is the height of these trees. Fig. 1 shows an example of the $k=5$ spanning trees, $T'_5(12)$, used in broadcasting among $n = 12$ processors with five-port communication model.

### 3.4 $k$ Does Not Divide $n-2$ and $n \geq k+2$

Let $\alpha = (n-2) \bmod k$ and $1 \leq \alpha \leq k-1$. We first construct $k$ trees, each having the topology of $T'_k(n-\alpha)$. These trees are labeled according to that described in the above case. We then add $\alpha$ nodes to each tree as follows:

For clarity, call these trees 0 through $k-1$. We will add nodes to trees in the order from 0 to $k-1$. For convenience, we refer to the *first* (resp. *second*) leaf node as the leaf node which is of the first (resp. second) rank among all leaf nodes ordered in the top-down manner and from left to right within each level. Note that since $n \geq k+2$, the tree $T'_k(n-\alpha)$ contains at least $k+2$ nodes and, therefore, has at least two leaf nodes. Let $p_0, p_1, \cdots, p_{\alpha-1}$ be the added $\alpha$ processors. In the process of adding these $\alpha$ nodes we create new internal nodes. The following algorithm has two tasks. First, it should specify to which parents these nodes are attached. Second, it should assign a processor to the new internal nodes in a way that does not violates Property 2 of Theorem 6. Note that there is no need to describe the assignment of processors to leaves since any assignment is valid. The algorithm of attaching $\alpha$ nodes to each tree and

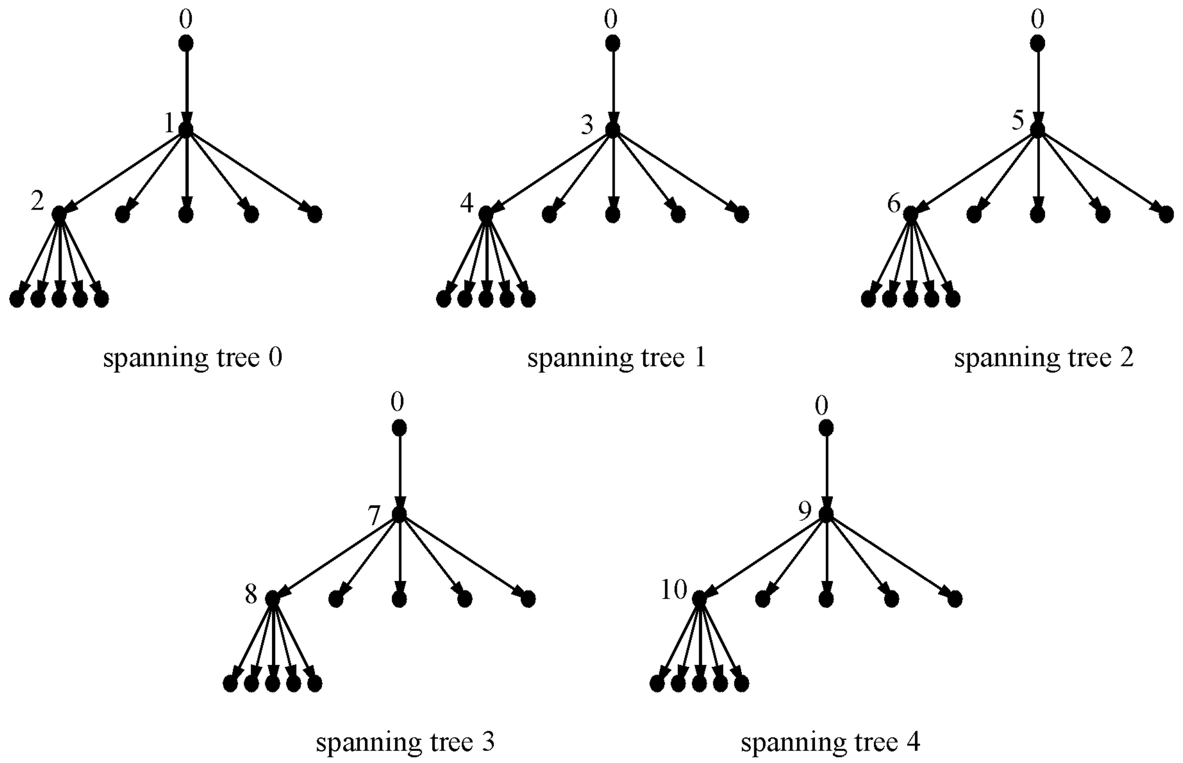

Fig. 1. An example of the five spanning trees used in broadcasting among 12 processors with five-port communication model.

assigning new internal nodes to the $\alpha$ new processors is as follows:

```
$c = 0$
$i = 0$
for $j = 0$ to $k - 1$ do
    $c = c + \alpha$
    if $(c \leq k)$ then
      attach $\alpha$ nodes to the first leaf node
          of the $j$th tree and assign processor $p_i$
          to the "new" internal node.
      if $(c = k)$ then
        $c = 0$
        $i = i + 1$
      endif
    else /* $c > k$ */
      $c = c - k$
      attach $\alpha - c$ nodes to the first leaf node
          of the $j$th tree and assign processor $p_i$
          to the "new" internal node.
      $i = i + 1$
      attach $c$ nodes to the second leaf node
          of the $j$th tree and assign processor $p_i$
          to the "new" internal node.
    endif
endfor
```

Following the algorithm, the new $\alpha$ nodes are either entirely attached to the first leaf node or spread between the first and the second leaf nodes of $T_k'(n - \alpha)$. The counter $c$ is used to make sure that each processor $p_i$ will have at most $k$ children summing over all $k$ trees. Since there are $\alpha k$ new children need to be covered and each new processor can have up to $k$ children, there are enough processors to act as new internal nodes. Fig. 2 shows an example of the tree structure after adding $\alpha = 3$ nodes to that of Fig. 1. Note that in this example $p_0 = 11$, $p_1 = 12$, and $p_2 = 13$. As in the previous example, processor 14 is not assigned to any internal node.

### 3.5 $n < k + 2$

For $n = 1$ and $n = 2$, the case is trivial. When $2 < n < k + 2$, the above approach of adding $\alpha = n - 2 < k$ extra nodes to the $n = 2$ case does not work because there is only one leaf node in $T_k'(2)$. Thus, we need to redefine the "second" leaf in a dynamic way. Specifically, we redefine the second leaf node as the first child of the first leaf node for this case. Then the algorithm of attaching the $\alpha$ nodes described above still holds. It is easy to show that the maximum height of these trees is 3. Fig. 3 shows an example of the trees for $n = 6$ and $k = 5$.

### 3.6 The Time Complexity

Let $h$ be the maximum height of these trees. Then by Theorem 6, broadcasting $m$ messages can be realized in $\lceil \frac{m}{k} \rceil + h - 1$ rounds. When $n < k + 2$, we showed in Section 3.5 that $h \leq 3$. We now derive $h$ for the other cases.

Let $f(n, k) = \lceil \log_k(nk - n + 1) \rceil - 1$, which is the height of $T_k(n)$. The maximum height of the $k$ trees defined in Section 3.3 is $1 + f(n - 1, k)$. The maximum height of the $k$ trees defined in Section 3.4 is $1 + f(n - 1 - \alpha + 2k, k)$, where $\alpha = (n - 2) \bmod k$. Here, the additive term of $2k$ in the first operand of the function $f$ is an upper bound for the maximum number of nodes that could be added while creating at most two new internal nodes. Now, since $f(n, k)$ is a monotonely increasing function with respect to $n$, we only focus on the height of the second case which is

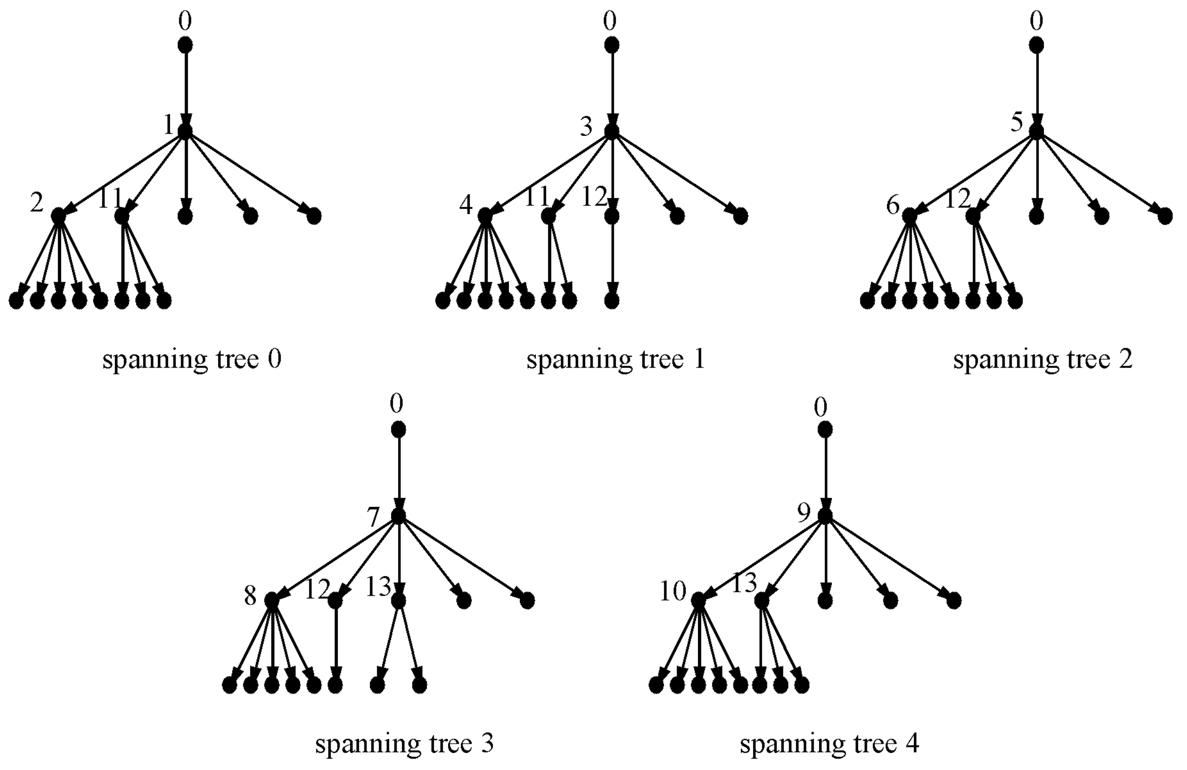

Fig. 2. An example of the five spanning trees used in broadcasting among 15 processors with five-port communication model.

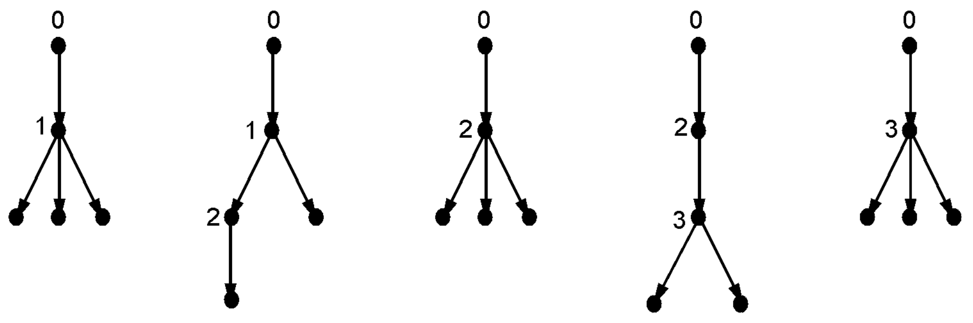

Fig. 3. An example of the five spanning trees used in broadcasting among five processors with five-port communication model.

$$\begin{aligned}1+&f(n-1-\alpha+2k,k)\\ &=\lceil\log_k((n-1-\alpha+2k)(k-1)+1)\rceil\\ &\le\lceil\log_k((n+2k)k)\rceil\\ &\le\lceil\log_k(n+2k)\rceil+1.\end{aligned}$$

Thus, the time complexity of our algorithm is at most

$$\left\lceil\frac{m}{k}\right\rceil+\lceil\log_k(n+2k)\rceil.$$

Note that $2\le k\le n-2$, thus, the time complexity is also bounded from above by $\lceil m/k\rceil+\lceil\log_k n\rceil+\lceil\log_k 3\rceil$.

Overall, for any $n$ and any $k\ge 2$, the time complexity is bounded from above by

$$\left\lceil\frac{m}{k}\right\rceil+max(2,\lceil\log_k(n+2k)\rceil).$$

Recall that the simple lower bound is $\lceil m/k\rceil+\lceil\log_{k+1}n\rceil-1$. Thus, the algorithm is about a multiplicative factor of $\log(k+1)/\log k$ above the lower bound in the delay-term, while the bandwidth-term is tight. Table 1 lists, for some selected $k$ and $n$, the number of additional rounds required by this algorithm from the simple lower bound. For accuracy, we use the time complexity of the $k$-tree algorithm before the approximation, i.e.,

$$\left\lceil\frac{m}{k}\right\rceil+\lceil\log_k((n-1-\alpha+2k)(k-1)+1)\rceil-1$$

when $n\ge k+2$.

## 4 The Rotation Algorithm

In this section, we describe a more complicated algorithm. This algorithm is based on three *broadcasting black-boxes* described later. A broadcasting black box $\mathcal{BBB}(h,t,\delta)$ (referred also as a system) is defined as follows:

- There are $h$ processors in the system.

TABLE 1
The Number of Additional Rounds Required by the $k$-tree Algorithm from the Simple Lower Bound

| | $n=32$ | $n=1024$ | $n=32768$ |
|---|---|---|---|
| $k=2$ | 2 | 4 | 6 |
| $k=3$ | 1 | 2 | 3 |
| $k=4$ | 1 | 1 | 2 |

- In each round, $k$ messages are injected into the system and are received by $k$ or less processors out of the $h$ processors.
- After $\delta$ rounds these $k$ messages are sent out of the system by $k$ or less processors (not necessarily the same processors).
- All the $h$ processors know these $k$ messages after at most $t$ rounds.

The parameter $t$ stands for the broadcasting time in this broadcasting black-box and the parameter $\delta$ stands for the delay time of the stream of messages from the time it is injected into the system to the time it is ejected out of the system.

The trivial broadcasting black-box is the broadcaster itself. We denote this special black-box by $\mathcal{BBB}(1,0,0)$ since we assume that the broadcaster already knows all the messages and sends them with no delay.

Using broadcasting black-boxes, we can generate the broadcasting algorithm by chaining black-boxes as follows: Let $\mathcal{BBB}_0,\dots,\mathcal{BBB}_\ell$ be $\ell+1$ broadcasting black-boxes, where $\mathcal{BBB}_0$ is the broadcaster black-box $\mathcal{BBB}(1,0,0)$ and $\mathcal{BBB}_i$ is of the form $\mathcal{BBB}(h_i,t_i,\delta_i)$. For all $1\le i\le\ell$, we connect the output stream of messages of $\mathcal{BBB}_{i-1}$ to the input stream of messages of $\mathcal{BBB}_i$. The output stream of $\mathcal{BBB}_\ell$ need not be sent. We refer to this algorithm as the *chain* algorithm. The overall number of processors in the system is $\sum_{i=0}^{\ell} h_i$. It is not difficult to verify that the processors in $\mathcal{BBB}_i$ know a message after $\sum_{j=0}^{i-1}\delta_j+t_i$ rounds from the time it was sent by the broadcaster. We get the following theorem:

**Theorem 7.** *For $1\le i\le\ell$, let $\mathcal{BBB}_i=\mathcal{BBB}(h_i,t_i,\delta_i)$ be nontrivial broadcasting black-boxes and let $\mathcal{BBB}_0=(1,0,0)$ be the trivial black-box consisting of the broadcaster. Then the chain algorithm for $\mathcal{BBB}_0,\mathcal{BBB}_1,\dots,\mathcal{BBB}_\ell$ is a broadcasting algorithm among $1+\sum_{j=1}^{\ell}h_j$ processors which takes $\max\{t_1,\delta_1+t_2,\delta_1+\delta_2+t_3,\dots,\sum_{j=1}^{\ell-1}\delta_j+t_\ell\}$ rounds.*

Our algorithm is based on the following proposition regarding a representation of any number $n$ as a sum of $\ell+2$ numbers with some special properties.

**Proposition 8.** *Any $n\ge1$ can be represented as*

$$n=1+n_1+n_2+\cdots+n_{\ell+1}$$

*with the following properties:*

1. *Either $0<n_{\ell+1}<2k$, or $n_{\ell+1}=0$ and $n_\ell=(k+1)^{d_\ell}-1$ for some $d_\ell\ge1$.*
2. *Depending on the previous property, for any value of $i$ between 1 and either $\ell$ or $\ell-1$ we have, $n_i=a_i(k+1)^{d_i-1}+(k-a_i)$ for some $1\le a_i\le k$ and $d_i\ge1$.*
3. *$1<d_\ell<d_{\ell-1}<\cdots<d_2<d_1\le\lceil\log_{k+1}n\rceil$ and, therefore, $\ell<\lceil\log_{k+1}n\rceil$.*

**Proof.** The proof is by construction. We first check whether $n-1=(k+1)^{d_1}-1$ for some $d_1\ge1$. If this is the case we are done. Otherwise, let $(d_1,a_1)$ be the largest pair (lexicographically) such that

$$n_1=a_1(k+1)^{d_1-1}+(k-a_1)\le n.$$

We set $n\leftarrow n-n_1$ and continue the same process for finding $n_2,n_3,\cdots$. We are done either by finding $n_\ell=(k+1)^{d_\ell}-1$ for some $d_\ell>1$ or when $0\le n_{\ell+1}<2k$.

Clearly, $d_1\le\lceil\log_{k+1}n\rceil$. In order to prove Property 3, we show that $d_2<d_1$. The rest follows the recursive construction. Assume to the contrary that $d_2=d_1$. We distinguish two cases: 1) $n_2=a_2(k+1)^{d_1-1}+(k-a_2)$ and 2) $n_2=(k+1)^{d_1}-1$. In the first case, we get $n\ge(a_1+a_2)(k+1)^{d_1-1}+(2k-a_1-a_2)$. If $a_1+a_2\le k$, then the above inequality contradicts the maximality of $a_1$ for a given choice of $d_1$. If $a_1+a_2\ge k+1$, then $n\ge(k+1)^{d_1}+(k-1)$. This inequality contradicts the maximality of $d_1$. In the second case the contradiction is achieved since again $n\ge(k+1)^{d_1}+(k-1)$. □

For the rest of the section we will describe the following three broadcasting black-boxes:

1. $\mathcal{BBB}((k+1)^d-1,d,d)$ for some $d\ge1$;
2. $\mathcal{BBB}(a(k+1)^{d-1}+(k-a),d,1)$ for some $d\ge1$ and $1\le a\le k$;
3. $\mathcal{BBB}(n,2,\infty)$ for $n<2k$.

We now use Proposition 8 to construct our chain algorithm. We apply the chain algorithm on the black-boxes $\mathcal{BBB}(1,0,0)$, $\mathcal{BBB}(n_1,d_1,1)$, $\dots$, $\mathcal{BBB}(n_\ell,d_\ell,1)$, $\mathcal{BBB}(n_{\ell+1},2,\infty)$ in case $n_\ell\ne(k+1)^{d_\ell}-1$ for some $d_\ell>1$, or $\mathcal{BBB}(1,0,0)$, $\mathcal{BBB}(n_1,d_1,1)$, $\dots$, $\mathcal{BBB}(n_\ell,d_\ell,1)$ otherwise. We get the following corollary:

**Corollary 9.** *In the above chain algorithm, the broadcasting time of any message is at most $\lceil\log_{k+1}n\rceil+1$ rounds.*

**Proof.** By Theorem 7, the complexity of the algorithm is $\max\{d_1,1+d_2,\dots,(\ell-1)+d_\ell,\ell+2\}$ rounds. By the third property of Proposition 8, we get $d_1\ge(j-1)+d_j$ for all $2\le j\le\ell$ and, hence, the round complexity is bounded by $\max\{d_1,\ell+2\}$. The corollary follows since $d_1\le\lceil\log_{k+1}n\rceil$ and $\ell+2\le\lceil\log_{k+1}n\rceil+1$. □

Note that in the above chain algorithm the delay of the stream of messages in the last black-box is insignificant because the output stream is no longer needed. Therefore, we can use the types of black-boxes the delay of which is $\infty$.

Now we return to our original multiple messages broadcasting algorithm. If all the messages after the $m$th message are null messages, then the chain algorithm yields the following theorem:

**Theorem 10.** *There exists a broadcasting algorithm among $n$ processors which takes at most $\lceil \frac{m}{k} \rceil + \lceil \log_{k+1} n \rceil$ rounds.*

Note that this bound is greater than the simple lower bound by one. It matches the second lower bound for many values of $k$, $n$, and $m$.

## 4.1 The Black-Box for "Nice" Numbers

In this section, we describe the broadcasting black-box $\mathcal{BBB}((k+1)^d - 1, d, d)$ for some $d \geq 1$. Let $n = (k+1)^d - 1$. This is a black-box for "nice" numbers since, together with the broadcaster the system, consists of $(k+1)^d$ processors and since this algorithm is based on some structure of the $d$-dimensional cube to the base of $k+1$. Recall that in each round $k$, new messages enter the system and after a delay of $d$ rounds, in each round, $k$ different messages leave the system. We denote these messages as input and output messages correspondingly.

Throughout the algorithm, the processors are dynamically partitioned into disjoint sets. In each round, each set of processors is instructed to send a message so that all the processors in the set send the same message to $k$ processors in other sets. It will be verified that any processor receives at most $k$ messages in this round. At the end of each round, processors are instructed to move to a new set. However, the size of the sets remain the same.

First we define the partition of the processors. The partition consists of $k \cdot d$ sets $S_j^i$ for $1 \leq i \leq k$ and $0 \leq j \leq d-1$. The sets are arranged as a matrix of size $k \times d$. The size of set $S_j^i$ is $(k+1)^j$. Indeed, $k \sum_{j=0}^{d-1} (k+1)^j = (k+1)^d - 1 = n$ and, hence, these sets include all the processors.

Denote the messages by $M_1, \ldots, M_m$ following the order they arrive to the black-box. Next we define for each set what message to send. This definition depends on the round. Assume that $M_1, \ldots, M_k$ enters the black-box at round 0 and let $r \geq 1$ be the current round. All the processors in $S_j^i$ are assigned the message $M_{(r-1-j)k+i}$. If $(r-1-j)k+i < 1$ (or $(r-1-j)k+i > m$ in the case of a finite number of message) then they are assigned no message.

Now we define the recipients of the messages sent by each set of processors. For $1 \leq i \leq k$, the processors in $S_0^i, \ldots, S_{d-2}^i$ each sends $k$ copies of its assigned message to processors in $S_{d-1}^i$. In addition, the $k$ input messages each goes to one of the sets $S_{d-1}^i$ for all $1 \leq i \leq k$. Indeed, $k \sum_{j=0}^{d-2} |S_j^i| + 1 = k \sum_{j=0}^{d-2} (k+1)^j + 1 = (k+1)^{d-1} = |S_{d-1}^i|$. For $1 \leq i \leq k$, the processors in the set $S_{d-1}^i$ send their assigned message to $k|S_{d-1}^i| - 1 = k(k+1)^{d-1} - 1$ other processors and one message as an output message. Indeed, this number is equal to $(k+1)^d - (k+1)^{d-1} - 1$, which is the number of all other processors.

We now verify that any processor receives at most $k$ messages. The processors in $S_j^i$, for $1 \leq i \leq k$ and $0 \leq j \leq d-2$, receive $k$ messages from the sets $S_{d-1}^1, \ldots, S_{d-1}^k$. The processors in $S_{d-1}^i$, for $1 \leq i \leq k$, receive $k-1$ messages from the sets $S_{d-1}^1, \ldots, S_{d-1}^{i-1}, S_{d-1}^{i+1}, \ldots, S_{d-1}^k$ and one message from a set $S_j^i$ for some $1 \leq j \leq d-2$ or an input message.

We conclude the description of the algorithm by defining the new partition of the processors. The sets $S_0^1, \ldots, S_0^k$ consist of the $k$ processors that received the $k$ input messages. The sets $S_j^i$, for $1 \leq i \leq k$ and $1 \leq j \leq d-1$, consist of the processors in $S_{j-1}^i$ and all the processors that received a message from them in this round. Note that processors change sets by going in a circle manner among the sets $S_j^i, S_{j+1}^i, \ldots, S_{d-1}^i$ for some $0 \leq j \leq d-1$. Some of the processors remain in the set $S_{d-1}^i$ throughout the algorithm.

**Example.** To demonstrate the flow of our algorithm, we follow a message arriving at the black-box until it is sent out of the black-box. Consider a processor $x$ from the set $S_{d-1}^i$ who has just received the message $M$ from the outside in round $r$. In round $r+1$, processor $x$ is the only one who belongs to the set $S_0^i$. In this round, $x$ sends $M$ to $k$ processors from the set $S_{d-1}^i$. These processors join $x$ to create the new $S_1^i$ set of round $r+2$. In round $r+2$, these $k+1$ processors send $k(k+1)$ copies of $M$ to $k(k+1)$ processors from the set $S_{d-1}^i$. The recipients of the message $M$ then join the senders to create the new $S_2^i$ set of round $r+3$ the size of which is $(k+1)^2$. This process continues in the same manner until round $r+d$. At the beginning of this round, $x$ belongs to $S_{d-1}^i$ and all the processors in this set know $M$ and are assigned to send $M$ to all other processors. We showed before that after this round all the processors know $M$. Moreover, one of the processors, say $x$, can send $M$ outside exactly $d$ rounds after $x$ receives $M$. At the same round, $x$ (or another processor from $S_{d-1}^i$) gets a new message from outside and the process starts again.

**Correctness.** The correctness of the algorithm is implied by the next lemma, which states the invariants maintained throughout the algorithm. Whenever we refer to a message $M_\ell$ for $\ell < 1$, we mean the null message.

**Lemma 11.** *In the beginning of round $r$:*

1. *Message $M_{(r-1-j)k+i}$ is known to all processors in $S_j^i$ for all $1 \leq i \leq k$ and $0 \leq j \leq d-1$.*
2. *Messages $M_1, \ldots, M_{(r-d)k}$ are known to all processors.*

**Proof.** The proof is by induction on the round number and follows the send, receive, and movement instructions of the algorithm. □

Following the dissemination of a particular message, it is not hard to see that the above lemma implies the correctness of the $\mathcal{BBB}$ as stated in the next theorem.

**Theorem 12.** *Any message is known to all processors after $d$ rounds and leaves the system as an output message after $d$ rounds.*

## 4.2 The Black-Box for "Special" Numbers

In this section, we describe the broadcasting black-box $\mathcal{BBB}(a(k+1)^{d-1} + (k-a), d, 1)$ for some $d \geq 1$ and $1 \leq a \leq k$. Let $n = a(k+1)^{d-1} + (k-a)$. This is a black-box for "special" types of numbers which do not cover all numbers. Recall that in each round $k$, new messages enter the system and, after a delay of one round, these $k$ messages leave the system. We denote these messages as input and output messages. The algorithm is a variation of the algorithm described in the previous section.

First, we define the partition of the processors. The partition consists of $k \cdot d$ sets $S_j^i$ for $1 \le i \le k$ and $0 \le j \le d-1$. The sets are arranged as a matrix of size $k \times d$. For $1 \le i \le k$ and $1 \le j \le d-1$, the size of $S_j^i$ is $a(k+1)^{j-1}$ and the size of $S_0^i$ is one. Indeed, $k\sum_{j=1}^{d-1} a(k+1)^{j-1} + k = a(k+1)^{d-1} + (k-a) = n$ and, hence, these sets include all the processors. The assignment of messages is the same as in the previous section.

Now we define the recipients of the messages sent by each set of processors. For $1 \le i \le k$, the processor in $S_0^i$ sends $k$ copies of its assigned message as follows: $a-1$ copies to processors from $S_{d-1}^i$, $k-a$ copies to processors to be specified later, and one copy is sent as an output message. For $1 \le i \le k$, each of the processors in $S_1^i, \ldots, S_{d-2}^i$ sends all of its $k$ copies to processors in $S_{d-1}^i$. In addition, the $k$ input messages each goes to one of the sets $S_{d-1}^i$ for all $0 \le i \le k$. Indeed,

$$k\sum_{j=1}^{d-2} a(k+1)^{j-1} + (a-1) + 1 = a(k+1)^{d-2} = |S_{d-1}^i|.$$

For $1 \le i \le k$, the processors in the sets $S_{d-1}^i$ send their assigned message to $k(k+1)^{d-1} - 1$ other processors. This number is $k|S_{d-1}^i| = n - |S_{d-1}^i| - (k-a)$. This means that there are $k-a$ processors that do not get the message assigned to the sets $S_{d-1}^i$. We choose these processors as processors that always remain in their sets $S_{d-1}^i$ and they receive this message $d$ rounds earlier from the processor of the set $S_0^i$.

The verification that each processor receives at most $k$ messages and the new partition is similar to the one appears in the previous section. Again, the correctness of the algorithm follows the next lemma which states the invariants maintained throughout the algorithm.

**Lemma 13.** *In the beginning of round $r$:*

1. *Message $M_{(r-1-j)k+i}$ is known to all processors in $S_j^i$ for all $1 \le i \le k$ and $0 \le j \le d-1$.*
2. *Message $M_{(r-2)k+i}$ is known to $k-a$ additional processors from a set $S_{d-1}^{i'}$ for some $i' \ne i$.*
3. *Messages $M_1, \ldots, M_{(r-d)k}$ are known to all processors.*

**Proof.** The proof is by induction on the round number and follows the send, receive, and movement instructions of the algorithm. □

Following the dissemination of a particular message, it is not hard to see that the above lemma implies the correctness of the $\mathcal{BBB}$ as stated in the next theorem.

**Theorem 14.** *Any message is known to all processors after d rounds and leaves the system as an output message after one round.*

## 4.3 The Black Box for "Small" Numbers

In this section, we describe the broadcasting black-box $\mathcal{BBB}(x, 2, \infty)$ for $0 \le x < 2k$. Note that from the broadcasting black-box definition, the broadcaster is outside the box. Thus, we consider broadcasting for $1 \le n \le 2k$ ($n$ includes the broadcaster) using the $k$-tree algorithm. For $1 \le n < k+2$, we use the construction in Section 3.5 and the height of the $k$ trees is at most 3, which means the delay within the black box is at most 2. For $k+2 \le n \le 2k$, we use the construction in Section 3.4 and the height of the $k$ trees is bounded by $h = 1 + f(n-1-\alpha+2k, k)$. From $n \le 2k$ and $\alpha = (n-2) \bmod k$, we get $n - \alpha \le k+2$. Thus, the height $h \le 1 + f(3k+1, k) \le 3$, which means the delay within the black-box is at most 2. Note that we have in fact given a construction for $\mathcal{BBB}(x, 2, 2)$.

## 5 Summary

We have presented two algorithms for broadcasting multiple messages in the multiport model. The $k$-tree algorithm has a very simple structure and scheduling policy. Furthermore, its time complexity is very close to the lower bound for all practical $k$ and $n$ (see Table 1). The rotation algorithm is optimal up to an additive term of one. For certain values of $n$, we can use the broadcaster to help broadcasting the messages to achieve an optimal algorithm. Since this method does not work for all values of $n$, we omit the description. Also, for some values of $m$ and $k$, our algorithm is optimal. The exact characterization and finding optimal algorithms for all values of $n$, $m$, and $k$ are still open.

As mentioned in the introduction, in the Postal model, even for $k = 1$, an optimal algorithm for all values of $n$ is not known. Actually, for very few values of $n$ do optimal algorithms exist . The ultimate problem is to find an optimal algorithm for the $k$-port postal model for the multiple messages broadcasting problem for any value of $n$, $m$, $k$, and $\lambda$ where $\lambda$ is the delay parameter in the postal model (see [3]).

## References

[1] V. Bala, J. Bruck, R. Bryant, R. Cypher, P. deJong, P. Elustondo, D. Frye, A. Ho, C.T. Ho, G. Irwin, S. Kipnis, R. Lawrence, and M. Snir, "The IBM External User Interface for Scalable Parallel Systems," *Parallel Computing,* vol. 20, no. 4, pp. 445–462, Apr. 1994.

[2] V. Bala, J. Bruck, R. Cypher, P. Elustondo, A. Ho, C.T. Ho, S. Kipnis, and M. Snir, "CCL: A Portable and Tunable Collective Communication Library for Scalable Parallel Computers," *Eighth Int'l Parallel Processing Symp.,* pp. 835–844, Apr. 1994.

[3] A. Bar-Noy and S. Kipnis, "Designing Broadcasting Algorithms in the Postal Model for Message-Passing Systems," *Mathematical Systems Theory,* vol. 27, no. 5, pp. 431–452, 1994.

[4] A. Bar-Noy and S. Kipnis, "Multiple Message Broadcasting in the Postal Model," *Networks,* vol. 29, no. 1, pp. 1–10, 1997.

[5] A. Bar-Noy, S. Kipnis, and B. Schieber, "Optimal Multiple Message Broadcasting in Telephone-Like Communication Systems," *Proc. Sixth Symp. Parallel and Distributed Processing,* pp. 216–223, Oct. 1994.

[6] I. Cidon and I. Gopal, "PARIS: An Approach to Integrated High-Speed Private Networks," *Int'l J. Digital and Analog Cabled Systems,* vol. 1, no. 2, pp. 77-85 Apr.-June 1988.

[7] E. Cockayne and A. Thomason, "Optimal Multi-Message Broadcasting in Complete Graphs," *Proc. 11th SE Conf. Combinatorics, Graph Theory, and Computing,* pp. 181–199 1980.

[8] J. Dongarra et al., "Document for a Standard Message-Passing Interface," *Message Passing Interface Forum,* Nov. 1993.

[9] *Express 3. 0 Introductory Guide,*Parasoft Corp., 1990.

[10] A.M. Farley, "Broadcast Time in Communication Networks," *SIAM J. Applied Math.,* vol. 39, no. 2, pp. 385–390, Oct. 1980.

[11] G. Fox, M. Johnson, G. Lyzenga, S. Otto, J. Salmon, and D. Walker, *Solving Problems on Concurrent Processors, Vol. I: General Techniques and Regular Problems.* Englewood Cliffs, N.J.: Prentice Hall, 1988.

[12] W. Groscup, "The Intel Paragon XP/S Supercomputer," *Proc. Fifth ECMWF Workshop on the Use of Parallel Processors in Meteorology,* pp. 173–187 1993.

[13] S.M. Hedetniemi, S.T. Hedetniemi, A. L. Liestman, "A Survey of Gossiping and Broadcasting in Communication Networks," *Networks,* vol. 18, no. 4, pp. 319–349, 1988.

[14] R. Karp, A. Sahay, E. Santos, K.E. Schauser, "Optimal Broadcast and Summation in the LogP Model," *Proc. Fifth Ann. Symp. Parallel Algorithms and Architectures,* June 1993.

[15] C.E. Leiserson et al. "The Network Architecture of the Connection Machine CM-5," *Proc. Fourth Ann. Symp. Parallel Algorithms and Architectures,* June 1992.

[16] J.F. Palmer, "The NCUBE Family of Parallel Supercomputers," *IEEE Int'l Conf. Computer Design,* 1986.

[17] C.B. Stunkel et al. "Architectures and Implementation of Vulcan," *Proc. Eighth Int. Parallel Processing Symp.,* pp. 268–274, Apr. 1994.

**Amotz Bar-Noy** received the BSc degree in 1981 in mathematics and computer science and the PhD degree in 1987 in computer science, both from the Hebrew University, Israel. From 1987 to 1989, he was a post-doctoral fellow at Stanford University, California. From 1989 to 1996, he was a research staff member with IBM T.J. Watson Research Center, New York. Since 1995, he has been an associate professor with the Electrical Engineering-Systems Department of Tel Aviv University, Israel. His main areas of interest are communication networks, combinatorial optimization and algorithms, and parallel and distributed computing. He is a senior member of the IEEE.

**Ching-Tien Ho** received a BS degree in electrical engineering from the National Taiwan University in 1979 and MS, MPhil, and PhD degrees in computer science from Yale University in 1985, 1986, and 1990, respectively.

He joined IBM Almaden Research Center as a research staff member in 1989. He was manager of the Foundations of Massively Parallel Computing group from 1994 to 1996, where he led the development of collective communication, as part of IBM MPL and MPI, for IBM SP-1 and SP-2 parallel systems. His primary research interests include communication issues for interconnection networks, algorithms for collective communications, graph embeddings, fault tolerance, and parallel algorithms and architectures. His current interests are parallel data mining and on-line analytical processing. He has published more than 80 journal and conference papers in these areas.

Dr. Ho is a co-recipient of the 1986 Outstanding Paper Award of the International Conference on Parallel Processing. He has received an IBM Outstanding Innovation Award, two IBM Outstanding Technical Achievement Awards, and four IBM Plateau Invention Achievement Awards. He has 11 patents granted or pending. He is on the editorial board of the IEEE Transactions on Parallel and Distributed Systems. He was one of program vice-chairs for the 1998 International Conference on Parallel Processing and has served on program committees of many parallel processing conferences and workshops. He is a member of the ACM and the IEEE.