Broadcasting Multiple Messages in the Multiport Model

Amotz Bar-Noy*
Tel Aviv University
Ramat Aviv Tel Aviv, 69978, Israel
amotz@eng.tau.ac.il

Ching-Tien Ho
IBM Almaden Research Center
San Jose, CA 95120
ho@almaden.ibm.com

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.

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, ..., n−1, where the source of the broadcast (the broadcaster) is processor 0. We also assume that the source has m messages, denoted by M1, M2, ..., Mm, 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 [12]). Several collective communication libraries, such as Express [10] 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 [9].

Several variations of the broadcasting problem were studied in the literature. (See [14] 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 [18], Thinking Machines’ CM-5 [16], NCUBE’s nCUBE/2 [17], Intel’s Paragon [13], and IBM’s SP2, as well as in some high-speed communication networks including PARIS [6] and AURORA [7].

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 trivial.)

The problem of broadcasting multiple messages in fully-connected systems was studied in several communication models. Cockayne and Thomason [8] and Farley [11] 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 - \left\lfloor \frac{m−1+ \lfloor \log n \rfloor}{n/2} \right\rfloor. In the bidirectional
telephone model, Bar-Noy, Kipnis, and Schieber [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 1-port model.) Their solution requires \((m - 1) + \lceil \log n \rceil\) rounds. Bar-Noy and Kipnis [3, 4] as well as Karp et al. [15] 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 [17], the CM-2 (where \(k\) is the dimension of the hypercube in both machines) and transputer-based machines.

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 \(k > 2\). 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. 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 1-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 knows 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 \(\lceil m/k \rceil - 1 + \lceil \log_{k + 1} n \rceil\) rounds.

However, for many combinations of \(n, m, k\), we have a lower bound which is larger by one from the previous lower bound. Specifically, we have the following lemma.

**Lemma 3** Let \(n' = (k + 1)\lceil \log_{k + 1} n \rceil\) and let \(\beta = ((m - 1) \mod k) + 1\). If \((n - 1)\beta > n' - 1\), then the lower bound for broadcasting \(m\) messages among \(n\) processors in the \(k\)-port model is \(\lceil m/k \rceil + \lceil \log_{k + 1} n \rceil\).

**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 \(t\) is \(k, (k + 1), (k + 1)^2, k, \ldots\). That is, at round \(t + \lceil \log_{k + 1} n \rceil\), 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. \(\Box\)

For example, when \(k = 3\) and \(m \mod k = 2\), 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 < \(n\) < 64. Also, when \(k = 3\) and \(m \mod k = 0\) (i.e., \(\beta = 3\)), the lower bound is \(\lceil m/k \rceil + \lceil \log_{k + 1} n \rceil\) for 23 < \(n\) < 64 among all \(n\) in the range of 16 < \(n\) < 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

\[
\begin{cases} 
\lceil m/k \rceil + \lceil \log_{k + 1} n \rceil - 1, & \text{if } m \mod k = 1, \\
\lceil m/k \rceil + \lceil \log_{k + 1} n \rceil, & \text{otherwise}.
\end{cases}
\]

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 $[m/k] - 1 + T$ 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 $[m/k] + \max(2, \lceil \log_k(n + 2k) \rceil)$. Note that a simple lower bound is $[m/k] + \lceil \log_{k+1} n \rceil - 1$. Thus, this algorithm is a multiplicative factor of $\log(k + 1)/\log k$ of the delay part of the lower bound.

#### 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 maximum height over all $k$ trees is $h$, 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 $[m/k] + h - 1$.

**Proof:** Ignore the restriction of the $k$-port model first. 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 non-root node will receive at most $k$ messages per rounds, 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 naturally from the scheduling.

### 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. Starts 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 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 internal nodes are jointly referred to as non-leaf nodes. Clearly, all non-leaf nodes in $T_k(n)$ have $k$ children except for the last non-leaf node which has $(n - 1) \mod k$ children. All the leaf nodes at the bottom level are pushed to the left. Also, only the last two levels can have leaf nodes. 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 $T'_k(n)$ 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 $\{0, 1, \ldots, n - 1\} \to$ 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: (i) $n \mod k = 2$ and $n \geq k + 2$, (ii) $n \mod k \neq 2$ and $n \geq k + 2$, and (iii) $n < k + 2$.

#### 3.3. Broadcasting with $n \mod k = 2$ and $n \geq k + 2$

We use $k$ spanning trees for broadcasting, each of topology $T'_k(n)$. When $n \mod k = 2$, every internal node has full fanout, i.e., $k$ children. The number of internal nodes per tree is $(n - 2)/k$. Thus, 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 easily 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 $|m/k| + h - 1$, where $h$ is the height of these trees. Figure 1 shows an example of the $k = 5$ spanning trees, $T^k_2$, used in broadcasting among $n = 12$ processors with 5-port communication model.

### 3.4. Broadcasting with $n \mod k \neq 2$ and $n \geq k + 2$

Let $\alpha = (n - 2) \mod k$ and $1 \leq \alpha \leq k - 1$. We first construct $k$ trees, each having the topology of $T^k_2(n - \alpha)$. 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 $T^k_2(n - \alpha)$ contains at least $k + 2$ nodes and, therefore, has at least two leaf nodes. Let $p_0, p_1, \ldots, p_{\alpha - 1}$ be the added $\alpha$ processors. The algorithm of attaching $\alpha$ nodes to each tree and assigning new internal nodes to the $\alpha$ new processors is as follows.

```plaintext
for $j = 0$ to $k - 1$ do 
  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.
  endif
  if ($c = k$) then
    $c = 0$
    $i = i + 1$
  endif
else /* $c > k$ */
  $c = c - k$
  attach $c - \alpha$ 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.
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 $T^k_2(n - \alpha)$. 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. Figure 2 shows an example of the tree structure after adding $\alpha = 3$ nodes to that of Figure 1.

### 3.5. Broadcasting with $n < k + 2$

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. Figure 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, broadcasting $m$ messages can be realized in $[m/k] + h - 1$ rounds, Theorem 6. When $n < k + 2$, $h \leq 3$. We now derive $h$ for the other cases.

Let $f(n, k) = \lfloor \log_k(n - 1 + \alpha + 2k, k) \rfloor$ where $\alpha = (n - 2) \mod k$. Since $f(n, k)$ is a monotonically increasing function with respect to $n$, we only focus on the height of the second case which is of

$$1 + f(n - 1 - \alpha + 2k, k) \leq \lfloor \log_k((n - 1 - \alpha + 2k)(k - 1 + 1)) \rfloor \leq \lfloor \log_k((n + 2k)) \rfloor + 1.$$

Thus, the time complexity of our algorithm is at most

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

Note that $2 \leq k \leq n - 2$, thus the time complexity is also bounded from above by $[m/k] + \lfloor \log_k(n) \rfloor + [\log_k 3]$.

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

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

Recall that the simple lower bound is $[m/k] + \lfloor \log_k(n) \rfloor - 1$. Thus, the algorithm is about a multiplicative factor of $\log(k + 1)/\log k$ of the lower bound in the delay-term, while the bandwidth-term is tight.

### 4. The Rotation Algorithm

In this section we describe our algorithm for any value of $n$. This algorithm is based on three broadcasting black-boxes described later. A broadcasting black box $\text{BBB}(h, t, \delta)$ is defined as follows:
There are $h$ processors in the system.

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 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 $BBB(1,0,0)$ since we assume that the broadcaster already knows all the messages and sends them with no delay.

Using broadcasting black-boxes, we can generate broadcasting algorithm by chaining black-boxes as follows. Let $BBB_1, \ldots, BBB_\ell$ be $\ell + 1$ broadcasting black-boxes where $BBB_1$ is the broadcasting black-box $BBB(1,0,0)$ and $BBB_\ell$ is of the form $BBB(h_{\ell}, t_{\ell}, \delta_{\ell})$. For all $1 \leq i \leq \ell$, we connect the output stream of messages of $BBB_{i-1}$ to the input stream of messages of $BBB_i$. The output stream of $BBB_\ell$ 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 $BBB_i$ know a message after $\sum_{j=1}^{i-1} \delta_j + t_i$ rounds from the time it was sent by the broadcaster. We get the following theorem:

**Theorem 7** For $1 \leq i \leq \ell$, let $BBB_i = BBB(h_i, t_i, \delta_i)$ be non-trivial broadcasting black-boxes and let $BBB_0 = BBB(1, 0, 0)$ be the trivial black-box consisting of the broadcaster. Then the chain algorithm for $BBB_0, BBB_1, \ldots, BBB_\ell$ is a broadcasting algorithm among $1 + \sum_{j=1}^{\ell} h_j$ processors which takes max{$t_1, \delta_1 + t_2, \delta_2 + t_3, \ldots, \sum_{j=1}^{\ell-1} \delta_j + t_\ell$} 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 $n \geq 1$ 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 $d_\ell \geq 1$.

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 $1 \leq a_i \leq k$ and $d_i \geq 1$.

3. $1 < d_\ell < d_{\ell-1} < \cdots < d_2 < d_1 \leq \left\lceil \log_{k+1} n \right\rceil$, and therefore $\ell < \left\lceil \log_{k+1} n \right\rceil$.

**Proof:** The proof is by construction. We first check whether $n - 1 = (k+1)^d - 1$ for some $d_1 \geq 1$. If this is the case we are done. Otherwise, let $(d_1, a_1)$ be the largest pair (lexicographically) such that $n_1 = a_1 (k+1)^{d_1} - 1 + (k - a_1) \leq n$. We set $n \leftarrow n - n_1$ and continue the same process for finding $n_2, n_3, \ldots$. We are done either by finding $n_\ell = (k+1)^{d_\ell} - 1$ for some $d_\ell > 1$ or when $0 \leq n_{\ell+1} < 2k$. \( \square \)

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

1. $BBB((k+1)^d - 1, d, d)$ for some $d \geq 1$;

2. $BBB(a (k+1)^{d-1} + (k - a), d, 1)$ for some $d \geq 1$ and $1 \leq a \leq k$;

3. $BBB(n, 2, \infty)$ for $n < 2k$.

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

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

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

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 $\left\lceil \frac{n}{2} \right\rceil + \left\lceil \log_{k+1} n \right\rceil$ rounds.

Note that this bound is far by at most an additive term of one from the lower bound.

### 4.1. The Broadcasting Black Box for “nice” numbers

In this subsection we describe the broadcasting black-box $BBB((k+1)^d - 1, d, d)$ for some $d \geq 1$. Let $n =$
strategy is to partition the processors in sets such that each goes to one of the sets \( S_i \) for \( 0 \leq i \leq d - 1 \). The size of the sets remain the same.

For \( 0 \leq i \leq d - 1 \), each set \( S_i \) consists of \( (k + 1)^d \) processors. Let \( k \) be any positive integer greater than \( 0 \). The sets are arranged as a matrix of size \( k \times d \). The size of set \( S_{ij} \) is \( (k + 1)^j \). Indeed, \( k \sum_{j=1}^{d-1} (k+1)^j = (k+1)^d - 1 = n, \) and hence these sets include all the processors.

Next we define for each set \( S_i \) a message to send. This definition depends on the round. Let \( r \) be the current round. All the processors in \( S_{i} \) send the message \( M_{(r-1)k+i} \) where \( (r-1)k+i \leq n \leq (r-1)k+i+1 \) or \( (r-1)k+i \geq n \) in the case of a finite number of messages after \( d \) rounds. In this round, \( (r-1)k+i \) processors in \( S_{i} \) change their assigned message to \((k+1)^d - 1 - 1 \) other processors and one message as an output message. Indeed, \( (k+1)^d - 1 \) is the number of all processors.

We now verify that any processor receives at most \( k \) messages. The processors in \( S_{i} \), for \( 1 \leq i \leq k \) and \( 0 \leq j \leq d - 2 \), receive \( k \) messages from the sets \( S_{i_j}, \ldots, S_{i_{j+1}} \). The processors in \( S_{i+1} \), for \( 1 \leq i \leq k \) and \( 1 \leq j \leq d - 1 \), receive \( k \) messages from the sets \( S_{i-1}^{j+1}, \ldots, S_{i+1}, S_{i+2}^{j+1}, \ldots, S_{d-1}^{j+1} \), and one message from a set \( S_{i}^{j} \) 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 \( S_{i+1}^{j+1}, \ldots, S_{d-1}^{j+1} \) consist of the processors that received the \( k \) input messages. The set \( S_{i}^{j} \), for \( 1 \leq i \leq k \) and \( 1 \leq j \leq d - 1 \), consist of the processors in \( S_{i+1}^{j+1} \) and all the processors that received a message from them in this round. Note that all processors change sets by going in a circle manner among the sets \( S_{i}^{j}, S_{i+1}^{j+1}, \ldots, S_{d-1} \) for some \( 0 \leq j \leq d - 1 \). Some of the processors remain in the set \( S_{d-1} \) throughout the algorithm.

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 \( \tau \):

1. Message \( M_{(\tau-1)k+i} \) is known to all processors in \( S_{i}^{j} \) for all \( 1 \leq i \leq k \) and \( 0 \leq j \leq d - 1 \).

2. Messages \( M_{1}, \ldots, M_{(\tau-1)k} \) 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 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 Broadcasting Black Box for “special” numbers

In this subsection we describe the broadcasting black-box \( \text{BBB}(a(k+1)^{d-1}+(k-a),d,1) \) 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” type 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 rounds 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 subsection.

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

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

We now verify that any processor receives at most \( k \) messages. The processors in \( S_{i}^{j} \), for \( 1 \leq i \leq k \) and \( 0 \leq j \leq d - 2 \), receive \( k \) messages from the sets \( S_{i}^{j+1}, \ldots, S_{i}^{j+k} \). The processors in \( S_{i+1}^{j} \), for \( 1 \leq i \leq k \) and \( 1 \leq j \leq d - 1 \), receive \( k \) messages from the sets \( S_{i}^{j+1}, \ldots, S_{i}^{j+k}, S_{i}^{j+k+1}, \ldots, S_{i}^{j+k+2} \) and one message from a set \( S_{i+1}^{j} \) 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 \( S_{i+1}^{j+1}, \ldots, S_{i+k}^{j+1} \) consist of the processors that received the \( k \) input messages. The set \( S_{i+1}^{j+1}, \ldots, S_{i+k}^{j+1} \) for \( 1 \leq i \leq k \) and \( 1 \leq j \leq d - 1 \), consist of the processors in \( S_{i+1}^{j+1} \) 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 \( S_{i+1}^{j+1}, \ldots, S_{i+k}^{j+1} \) for some \( 0 \leq j \leq d - 1 \). Some of the processors remain in the set \( S_{i+k}^{j+1} \) throughout the algorithm.
goes to one of the sets $S'_{j-1}$ for all $0 \leq i \leq k$. Indeed,
\[ k \sum_{i=1}^{d-1} (a(k+1)+1) = a(k+1)^{d-2} = |S'_{d-1}|. \]
For $1 \leq i \leq k$, the processors in the sets $S'_{d-1}$ send their
assigned message to $k(k+1)^{d-1} - 1$ other processors. This
number is $k|S'_{d-1}| = n - |S'_{d-1}| = (k - a)$. This means
that there are $k - a$ processors that do not get the message
assigned to the sets $S'_{d-1}$. We choose these processors as
processors that always remain in their sets $S'_{d-1}$ and they
receive this message $d$ rounds earlier from the processor of
the set $S'_{d-1}$.

The verification that each processor receives at most $k$
messages and the new partition is similar to the one appears
in the previous subsection. 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 $\tau$:

1. Message $M_{(\tau-1-j)k+i}$ is known to all processors in
$S'_{j}$ for all $1 \leq i \leq k$ and $0 \leq j \leq d-1$.

2. Message $M_{(\tau-2)k+i}$ is known to $k-a$ additional processors from a set $S'_{d-1}$ for some $d' \neq i$.

3. Messages $M_1, \ldots, M_{(\tau-d)k}$ 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 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 Broadcasting Black Box for “small” numbers

In this subsection, we describe the broadcasting black-box $BBB(x, 2, \infty)$ for $0 \leq x < 2k$. Note that from the
broadcasting black-box definition, the broadcaster is outside
the box. Thus, we consider broadcasting for $1 \leq n \leq 2k$ ($n$ includes the broadcaster) using the $k$-tree algorithm.
For $1 \leq n < k + 2$, 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 $k + 2 \leq n \leq 2k$, 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
$n \leq 2k$ and $\alpha = (n-2) \mod k$, we get $n - \alpha \leq k + 2$.
Thus, the height $h \leq 1 + f(3k+1, k) \leq 3$, which means the
delay within the black box is at most 2. It should be noted
these constructions can be easily modified to $BBB(x, 2, 2)$,
for $x < 2k$, although the outstream is not needed for our case.

### 5. Open Problems

Our algorithm is still one round higher than the lower bound. For certain values of $n$ we can use the broadcaster
to help broadcasting the messages to achieve an optimal al-
gorithm. 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 $k = 1$ there are not known optimal algorithm for
all values of $n$. Actually, for very few values of $n$ there exist
optimal algorithms. The ultimate problem is to find an
optimal algorithm for the $k$-port postal model for the multi-
ple messages broadcasting problem for any value of $n, m,
and $k$ where $\lambda$ is the delay parameter in the postal model
(see [3]).

### References


Figure 1. An example of the 5 spanning trees used in broadcasting among 12 processors with 5-port communication model.

Figure 2. An example of the 5 spanning trees used in broadcasting among 15 processors with 5-port communication model.

Figure 3. An example of the 5 spanning trees used in broadcasting among 5 processors with 5-port communication model.