Optimal Multiple Message Broadcasting in Telephone-Like Communication Systems

Amotz Bar-Noy* Shlomo Kipnis† Baruch Schieber*

Abstract

We consider the problem of broadcasting multiple messages from one processor to many processors in telephone-like communication systems. In such systems, processors communicate in rounds, where in every round, each processor can communicate with exactly one other processor by exchanging messages with it. Finding an optimal solution for this problem was open for over a decade. In this paper, we present an optimal algorithm for this problem when the number of processors is even. For an odd number of processors, we provide an algorithm which is within an additive term of 1 of the optimum. A by-product of our solution is an algorithm for the problem of broadcasting multiple messages for any number of processors in the simultaneous send/receive model. In this latter model, in every round, each processor can send a message to one processor and receive a message from another processor.

1 Introduction

Broadcasting is an important communication operation in many multi-processor 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.

Several variations of the broadcasting problem were studied in the literature. (See [11] 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 multi-processor systems, such as IBM's Vulcan [4, 18], Thinking Machines' CM-5 [12], NCUBE's nCUBE/2 [17], Intel's Paragon [10], and IBM's Scalable POWERparallel Systems SP1 and 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. We assume that there are n processors 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.

The problem of broadcasting multiple messages in fully-connected systems was studied in several communication models. Cockayne and Thomason [8] and Farley [9] 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 $2m + \lfloor \log n \rfloor - \left\lfloor \frac{m-1+2^{\lceil \log n \rceil}}{n/2} \right\rfloor$.

More recently, Bruck, Cypher, and Ho [5] as well as Bar-Noy and Kipnis [3] investigated the problem of broadcasting multiple messages in the simultaneous send/receive model. In this model, in every round,

*IBM T.J. Watson Research Center, Yorktown Heights, NY 10598. Email: {amotz,sbar}@watson.ibm.com.

†IBM Israel Science and Technology, Haifa, ISRAEL 31905. Email: kipnis@haifasc3.vnet.ibm.com.

0-8186-6427-4/94 $04.00 © 1994 IEEE

each processor can send a message to one processor and receive a message from another. The solution of [5] is within an additive term of $O(\log\log n)$ of the optimum, while the solution of [3] is within an additive term of 1 of the optimum. Recently, Ho [13] sketched an optimal solution to the problem of broadcasting multiple messages in the simultaneous send/receive model, which requires $(m-1)+\lceil\log n\rceil$ rounds. Ho [14] also provided an optimal solution (up to an additive term of 1) to the problem of broadcasting multiple messages in the multi-port send/receive model. Bar-Noy and Kipnis [1, 2] as well as Karp, Sahay, Santos, and Schauser [16] 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.

This paper investigates the problem of broadcasting multiple messages in another model of communication for multi-processor systems, namely the *bidirectional telephone model.* In this model, processors communicate in rounds. In every round, each processor p may communicate with exactly one other processor q by having processors p and q exchange messages (i.e., processor p sends a message to processor q and processor q sends a message to processor p). This model, in addition to its being theoretically interesting, describes the situation in some parallel architectures that use circuit switching techniques, such as certain SIMD hypercube or mesh architectures.

Optimal-time solutions for the problem of broadcasting multiple messages in bidirectional telephone systems are known only for specific values of m and n. For $m=1$ and for any value of n, a simple folklore algorithm based on recursive doubling provides an optimal solution for this problem. The running time of this algorithm is $\lceil\log n\rceil$ rounds. (In fact, this solution is valid also in the unidirectional telephone model.) For $n=2^k$, Ho [15] provided an optimal broadcasting algorithm based on disjoint spanning trees in a binary hypercube. The running time of this algorithm is $(m-1)+\log n$ rounds. However, Ho's solution relies heavily on the fact that n is a power of 2 and cannot be extended to other values of n. Finding an optimal solution for this problem for any value of n was open for over a decade.

In this paper, we present the following solutions for the broadcasting problem in telephone systems. For even n, we provide an optimal algorithm that requires $(m-1)+\lceil\log n\rceil$ rounds. For odd n, we present an algorithm that is optimal up to an additive term of 1. This algorithm requires $(m-1)+\lceil\log n\rceil+\frac{m}{n-1}+c$ rounds. (The exact value of the additive term $c\in\{-2,-1,0,1,2\}$, depends on the particular values of m and n. However, our algorithm is optimal to within an additive term of 1 for all possible values of m and for odd n.) As a by-product of our solution in the telephone model, we solve the broadcasting problem optimally in the simultaneous send/receive model. (This result was independently reported by Ho [13]). For even n, the solution for the simultaneous send/receive model is identical to the solution for the bidirectional telephone model. For odd n, a slight modification of the algorithm for even n yields an optimal solution in the simultaneous send/receive model.

2 Lower bounds

We first present the lower bound for an even number of processors. The lower bound for this case is implied by the earliest time at which the last message sent by the broadcaster can become known to all the processors.

Theorem 1 *Broadcasting m messages among n processors, when n is even, requires at least $(m-1)+\lceil\log n\rceil$ rounds.*

Proof: The last message sent by the broadcaster cannot be sent before round m. To broadcast this message among the n processors, at least $\lceil\log n\rceil$ rounds are required. Therefore, the total number of rounds cannot be less than $(m-1)+\lceil\log n\rceil$. □

Now, we present the lower bound for an odd number of processors. The lower bound for this case follows from a counting argument by comparing the number of necessary receive operations and the number of send operations that carry new information. We call such send operations *useful send operations.*

Theorem 2 *Broadcasting m messages among n processors, when n is odd, requires at least $m+\left\lceil\frac{m}{n-1}+\frac{n-2}{n-1}\lceil\log n\rceil-\frac{2^{\lceil\log n\rceil}-1}{n-1}\right\rceil$ rounds.*

Proof: Since $n-1$ processors need to receive m messages each, it follows that any algorithm must have $m(n-1)$ useful send operations.

Assume an algorithm that requires r rounds. Partition the r rounds into three disjoint sets as follows. (i) The first set consists of the first $\lceil \log n \rceil$ rounds. (ii) The second set consists of all the rounds, excluding the first $\lceil \log n \rceil$ rounds, in which the broadcaster participates. Since the broadcaster must participate in at least m rounds, in each of which the broadcaster introduces a new message, it follows that the second set must contain at least $m - \lceil \log n \rceil$ rounds. Let the number of rounds in the second set be $m - \lceil \log n \rceil + y$. (iii) The third set consists of the remaining rounds not included in the first and the second sets. Let x be the number of rounds in the third set. Then, we have $r = m + y + x$.

We now count the number of useful send operations in the three disjoint sets. In the first set, in round i, for $1 \leq i \leq \lceil \log n \rceil$, there could be at most 2^{i-1} useful send operations. Therefore, all together, in the first set there are at most $2^{\lceil \log n \rceil} - 1$ useful send operations. In the second set, in each round, the partner of the broadcaster is idle and one more processor is idle due to the fact that n is odd. Therefore, in each round of the second set there are at most $n-2$ useful send operations. All together, in the second set, there are at most $(m - \lceil \log n \rceil + y)(n-2)$ useful send operations. Finally, in each one of the x rounds of the third set, there could be at most $n-1$ useful send operations since the broadcaster is idle. All together, in the third set there could be at most $x(n-1)$ useful send operations. Summing over the three sets we get,

$$(2^{\lceil \log n \rceil} - 1) + (m - \lceil \log n \rceil + y)(n-2) + x(n-1) \geq m(n-1) .$$

By noting that $y(n-1) \geq y(n-2)$, we get

$$y + x \geq m - \frac{2^{\lceil \log n \rceil} - 1}{n-1} - \frac{(m - \lceil \log n \rceil)(n-2)}{n-1} .$$

Now, since $r = m + y + x$, we have

$$r \geq m + \frac{m}{n-1} + \frac{n-2}{n-1} \lceil \log n \rceil - \frac{2^{\lceil \log n \rceil} - 1}{n-1} .$$

The claim follows since r must be an integer. □

Notice that the lower bound for odd n is at least $(m-1) + \lceil \log n \rceil - 2 + \left\lceil \frac{m}{n-1} \right\rceil$. This means that, up to an additive term of 2, the lower bound for the odd case is greater than the lower bound for the even case by $\frac{m}{n-1}$.

3 The algorithm for $n = 2^k$

In this section, we describe the algorithm for the case when the number of processors is a power of two, that is, when $n = 2^k$, for some integer $k > 0$. The algorithm for this case is not new and was described in at least two different ways (see [15, 3]). Here, we present a third description of this algorithm which clarifies the development of the algorithms for other values of n. We first describe the algorithm assuming that the source has an infinite number of messages to broadcast. We then prove that using this algorithm it takes exactly $k+1$ rounds for any message to become known to all the processors. Thus, if we have a finite number of messages m, the truncation of this algorithm would take $m + k$ rounds. Finally, we modify the algorithm for the case of a finite number of messages m, and we prove that the modified algorithm is optimal and takes $(m-1) + k = (m-1) + \log n$ rounds.

Denote the infinite stream of messages to be sent by $M_1, M_2, \ldots$. In the description of the algorithm, whenever a reference is made to a message M_i, where $i < 1$, it is assumed that M_i is an empty message. In describing the algorithm, we describe each round $t \geq 1$ of it. As will be shown later, at the beginning of each round $t \geq 1$ of the algorithm, the following two invariants hold:

1. Messages $M_1, M_2, \ldots, M_{t-k-1}$ are known to all the processors.
2. The $n-1$ processors, excluding the source, can be partitioned into k sets $T_0, T_1, \ldots, T_{k-1}$, such that set T_i is of size 2^i, for $i = 0, 1, \ldots, k-1$. This partition has the following property: for each $i = 0, 1, \ldots, k-1$, message M_{t-i-1}, is known to all the processors in set T_i.

The algorithm is as follows. Each round $t \geq 1$ is defined by a perfect matching on the set of n processors. This matching determines which pairs of processors communicate with one another in round t. In this matching, the 2^{k-1} processors of set T_{k-1} are matched with the other 2^{k-1} processors. More specifically, one

special processor in T_{k-1}, denoted by r, is matched with the broadcaster, and the remaining $2^{k-1}-1$ processors in T_{k-1} are matched with the processors in sets $T_0, T_1, \ldots, T_{k-2}$. We now specify the messages communicated by each of the matched pairs of processors in round t. Processor r receives message M_t from the broadcaster. For any other pair of matched processors, u and v, where $u \in T_{k-1}$ and $v \in T_i$, processor u sends message M_{t-k} to processor v and processor v sends message M_{t-i-1} to processor u.

To complete the description of the algorithm and show its correctness, we prove by induction that the two invariants above can be made to hold at the beginning of each round $t \geq 1$. For $t = 1$, the two invariants hold trivially. Now, assume that the two invariants hold at the beginning of round t and show that they can also be made to hold at the beginning of round $t+1$. In round t, the processors in T_{k-1} communicate message M_{t-k} to the processors in all the other sets. Therefore, at the beginning of round $t+1$, message M_{t-k} is known to all the processors, which proves Invariant 1. To prove Invariant 2, we define a new partition $T_0, T_1, \ldots, T_{k-1}$ for round $t+1$. The new set T_0 is $\{r\}$. Note that processor r indeed knows message $M_{(t+1)-0-1} = M_t$, which it received from the broadcaster in round t. For $i = 1, 2, \ldots, k-1$, the new set T_i consists of the processors in the old set T_{i-1} and the processors from the old set T_{k-1} that were matched to them in round t. It follows that the size of the new set T_i is 2^i as required. Also, all the processors in the new set T_i know message $M_{(t+1)-i-1} = M_{t-(i-1)-1}$. This is because the processors from the old set T_{i-1} know this message at the beginning of round t (by invariant 2) and they communicate this message to the processors from the old set T_{k-1} which were matched with them in round t.

Finally, to make the description of the algorithm more intuitive, we describe it from the point of view of a message M_t. In round t, message M_t is sent from the broadcaster to some processor r. This processor r is assigned to the singleton set T_0 at the beginning of round $t+1$. During the next k rounds, the number of processors (excluding the broadcaster) that know message M_t is doubled in every round. In particular, at the end of round $t+i$, the number of processors that know message M_t is 2^i. These 2^i processors are assigned to the set T_i for round $t+i+1$. At the end of round $t+k$, all the processors know message M_t.

The above algorithm and analysis assume that the source has an infinite number of messages to broadcast. However, the same algorithm can be used when the number of messages m is finite, by having the source stop sending messages after it sends the last message M_m. The running time of such a *truncated* algorithm is $m+k = m+\log n$, which is not optimal. To obtain an optimal algorithm for finite m, we need to modify only the tail of the algorithm as follows:

- In rounds t, for $m \leq t \leq (m-1)+k$, the source sends message M_m.
- The algorithm terminates at the end of round $(m-1)+k$.

This modification can be viewed as adding messages $M_{m+1}, M_{m+2}, \ldots, M_{m+k-1}$, all of which are identical to message M_m. Consider the two invariants after $m-1+k$ rounds (i.e., at the beginning of round $m+k$). By Invariant 1, messages $M_1, M_2, \ldots, M_{m-1}$ are known to all the processors. By Invariant 2, there exists a partition of the $n-1$ processors into sets $T_0, T_1, \ldots, T_{k-1}$, such that all the processors in set T_i know message $M_{m+k-i-1}$. However, since $M_m = M_{m+1} = \cdots = M_{m+k-1}$, it follows that all the processors know message M_m. The following theorem summarizes this modified algorithm.

Theorem 3 *Broadcasting m messages among n processors, when n is a power of two, can be done in $(m-1)+\log n$ rounds, which is optimal.*

4 The algorithm for an even n

In this section, we describe a variation of the algorithm of Section 3 for the case when the number of processors is even. As in Section 3, we first describe the algorithm assuming that the source has an infinite number of messages to broadcast, and we prove that it takes exactly $\lceil \log n \rceil + 1$ rounds for any message to become known to all the processors. Thus, if we have a finite number of messages m, then the truncation of this algorithm takes $m+\lceil \log n \rceil$ rounds. We then modify the algorithm for a finite number of messages m, and we prove that the modified algorithm is optimal and takes $(m-1)+\lceil \log n \rceil$ rounds.

Since n is even and not a power of two, let $n = 2^k + 2\ell$, where $0 < \ell < 2^{k-1}$. Note that $k = \lfloor \log n \rfloor$. Let $0 \leq j \leq k-1$ be such that $2^{j-1} < \ell \leq 2^j$.

In describing the algorithm, we describe round t of it, for $t \geq 1$. We will prove that, at the beginning of round t, for $t \geq 1$, the following two invariants hold:

1. Messages $M_1, M_2, \ldots, M_{t-k-2}$ are known to all the processors.
2. The set of $n-1$ processors (excluding the broadcaster) can be partitioned into $k+3$ sets $T_0, T_1, \ldots, T_{k-1}, X, Y, Z$ as follows. The size of set T_i is 2^i, for $i = 0, 1, \ldots, k-1$; the size of set X is ℓ; the size of set Y is $2^j - \ell$; and the size of set Z is $2\ell - 2^j$. This partition has the following properties:
   - For $i = 0, 1, \ldots, j-1$, message M_{t-i-1} is known to all the processors in set T_i.
   - For $i = j, j+1, \ldots, k-1$, message M_{t-i-2} is known to all the processors in set T_i.
   - Message M_{t-j-1} is known to all the processors in set X.
   - Messages $M_{t-k-1}, M_{t-k}, \ldots, M_{t-j-1}$ are known to all the processors in set Y.
   - Messages $M_{t-k-1}, M_{t-k}, \ldots, M_{t-j-2}$ are known to all the processors in set Z.

The algorithm is as follows. Round $t \geq 1$ is defined by a perfect matching on the set of n processors. This matching determines which pairs of processors communicate with one another in round t. For the matching, we further partition set T_{j-1} into two subsets $T_{j-1}^{(1)}$ of size $\ell - 2^{j-1}$ and $T_{j-1}^{(2)}$ of size $2^{j-1} - (\ell - 2^{j-1}) = 2^j - \ell$. We also partition set X into two subsets $X^{(1)}$ of size $2^j - \ell$ and $X^{(2)}$ of size $\ell - (2^j - \ell) = 2\ell - 2^j$. The matching for round t is defined as follows.

- The $2^j - \ell$ processors in set Y are matched with the processors in the subset $T_{j-1}^{(2)}$.
- The $2\ell - 2^j$ processors in set Z are matched with the processors in the subset $X^{(2)}$.
- The 2^{k-1} processors in set T_{k-1} are matched with the remaining processors. More specifically, one special processor in T_{k-1}, denoted by r, is matched with the broadcaster, and the other $2^{k-1}-1$ processors in T_{k-1} are matched with all the processors in sets $T_0, \ldots, T_{j-2}, T_{j-1}^{(1)}, X^{(1)}, T_j, \ldots, T_{k-2}$.

We now specify the messages communicated for each of the matched pairs in round t.

- The special processor r in T_{k-1} receives message M_t from the broadcaster.
- For any pair of two matched processors u and v, where $u \in T_{k-1}$ and $v \in T_i$, for $i = 0, 1, \ldots, j-2$, processor u sends M_{t-k-1} to v and processor v sends M_{t-i-1} to u.
- For any pair of two matched processors u and v, where $u \in T_{k-1}$ and $v \in T_i$, for $i = j, j+1, \ldots, k-2$, processor u sends M_{t-k-1} to v and processor v sends M_{t-i-2} to u.
- For any pair of two matched processors u and v, where $v \in T_{j-1}$ (and u is either in Y or in T_{k-1}), processor u sends M_{t-k-1} to v and processor v sends M_{t-j} to u.
- For any pair of two matched processors u and v, where $v \in X$ (and u is either in Z or in T_{k-1}), processor u sends M_{t-k-1} to v and processor v sends M_{t-j-1} to u.

To complete the description of the algorithm and show its correctness, we prove by induction that the two invariants above can be made to hold at the beginning of each round $t \geq 1$. For $t = 1$, the two invariants hold trivially. Now, assume that the two invariants hold at the beginning of round t and show that they can also hold at the beginning of round $t+1$. In round t, the processors in sets T_{k-1}, Y, and Z communicate message M_{t-k-1} to all the other processors. Therefore, at the beginning of round $t+1$, message M_{t-k-1} is known to all the processors, which proves Invariant 1. To prove Invariant 2, we define a new partition $T_0, T_1, \ldots, T_{k-1}$, X, Y, Z for round $t+1$ as follows.

- The new set T_0 is $\{r\}$. Note that processor r indeed knows message $M_{(t+1)-0-1} = M_t$, which it received from the broadcaster in round t.
- For $i = 1, 2, \ldots, j-1$, the new set T_i consists of the processors in the old set T_{i-1} and the processors from the old set T_{k-1} that were matched to them. The size of new set T_i is 2^i. Also, since in round t processors in old set T_{i-1} communicate message $M_{t-(i-1)-1}$, it follows that all the processors in new set T_i know message $M_{(t+1)-i-1}$.
- For $i = j+1, j+2, \ldots, k-1$, the new set T_i consists of the processors in the old set T_{i-1} and the

processors from the old set T_{k-1} that were matched to them. The size of new set T_i is 2^i. Also, since in round t processors in old set T_{i-1} communicate message $M_{t-(i-1)-2}$, it follows that all the processors in new set T_i know message $M_{(t+1)-i-2}$.

- The new set X consists of the processors in the old set T_{j-1} and the processors from the old set T_{k-1} that were matched to the subset $T_{j-1}^{(1)}$. The size of the new set X is ℓ. Also, since in round t processors in old set T_{j-1} communicate message $M_{t-(j-1)-1}$, it follows that all the processors in new set X know message $M_{(t+1)-j-1}$.

- The new set T_j consists of the processors in the old set X and the processors from the old set T_{k-1} that were matched to the subset $X^{(1)}$. The size of new set T_j is 2^j. Also, since in round t processors in old set X communicate message M_{t-j-1}, it follows that all the processors in new set T_j know message $M_{(t+1)-j-2}$.

- Sets Y and Z remain the same. Note that since in round t the processors in set Y receive message $M_{t-(j-1)-1} = M_{(t+1)-j-1}$ from processors in old set T_{j-1} and the processors in set Z received message $M_{t-j-1} = M_{(t+1)-j-2}$ from processors in old set X, the invariant holds also for these sets.

Again, to make the description of the algorithm more intuitive, we describe it from the point of view of a message M_t. In round t, message M_t is sent from the broadcaster to some processor r. This processor r is assigned to the singleton set T_0 at the beginning of round $t+1$. During the next j rounds, the number of processors (excluding the broadcaster) that know message M_t is doubled in every round. In round $t+j+1$, only ℓ out of the 2^j processors that currently know message M_t (that is, the set X) participate in disseminating this message. Consequently, at the end of this round, $2^j + \ell$ processors know message M_t. In the next round, only 2^j of these processors (that is, the set T_j) participate in disseminating message M_t by sending it to another set of 2^j processors. In rounds $i = t+j+3, t+j+4, \ldots, t+k$, only 2^{i-t-2} of the processors that know message M_t participate in disseminating it. Finally, in round $t+k+1$, the $2^{k-1}+\ell$ processors that know message M_t send it to the remaining processors.

The above algorithm and analysis assume that the source has an infinite number of messages to broadcast. The truncation of this algorithm to handle a finite number of messages is not optimal by an additive term of 1. To obtain an optimal algorithm for finite m, we modify the tail of this algorithm in a similar manner to the modified algorithm at the end of Section 3.

- In rounds t, such that $m \leq t \leq (m-1)+k+1$, the source sends message M_m.

- The algorithm terminates at the end of round $(m-1)+k+1$.

The correctness and optimality of this modified algorithm follow from arguments similar to those presented at the end of Section 3. The following theorem summarizes this algorithm.

Theorem 4 *Broadcasting m messages among n processors, when n is even, can be done in $(m-1) + \lceil \log n \rceil$ rounds.*

We note that when n is not a power of 2, there is some redundancy in this modified algorithm. Indeed, in the above description, processors in sets Y and Z receive message M_m more than once.

5 The algorithm for an odd n

In this section, we describe the algorithm for an odd number of processors. The basic idea behind the algorithm for odd n is, in each round, to idle one non-broadcaster processor and to use the algorithm for an even number of processors on the remaining $n-1$ processors. The crux of this algorithm is in scheduling the processors in such a way that the idle times will be distributed as evenly as possible among the $n-1$ non-broadcaster processors.

Suppose that we managed to balance the idle times in such a way that after all the rounds of the algorithm for even n terminate we have: (i) each processor knows all the m messages but at most $\left\lceil \frac{m}{n-1} \right\rceil$ of them, and (ii) each message is known to $n-2$ out of the $n-1$ non-broadcaster processors. Then, by adding $\left\lceil \frac{m}{n-1} \right\rceil$ rounds of perfect matching between appropriate pairs of processors, all the $n-1$ non-broadcaster processors will be able to know all the m messages. Now, observe that the lower bound for the odd case is greater than

the upper bound for the even case by an additive term of $\left\lceil \frac{m}{n-1} \right\rceil + c$, where the $-2 \leq c \leq 1$. (The exact value of c depends on the values of m and n.) Therefore, if we manage to balance the idle times as stated above, we will have an algorithm with a running time that is greater by at most 2 from the optimum. We show that the running time of our algorithm is even closer to the lower bound by counting the number of useful send operations (as was done in the lower bound proof of Theorem 2).

We now sketch how we distribute the idle times as evenly as possible between the $n-1$ non-broadcaster processors. The simplest way would be to run $n-1$ instances of the algorithm, each for $\left\lceil \frac{m}{n-1} \right\rceil$ messages (except maybe for the last instance), where in each such instance, a different processor would be idle. However, while changing the identity of the idle processors, some send operations become useless. Our goal is to minimize the number of such useless send operations.

Towards this goal, we employ a "pipelining" scheme — we replace the idle processor without stopping the algorithm. This may cause some processors not to receive their assigned messages. To minimize this effect, we make a processor idle only when it belongs to one of the sets T_{k-1}, Y, and Z. A close inspection of the algorithm reveals that every processor eventually belongs to one of these sets. Moreover, if at a certain round, a processor does not belong to $T_{k-1} \cup Y \cup Z$, then it will belong to T_{k-1} in one of the next $\lceil \log n \rceil$ rounds.

Consider the set S which consists of all the non-broadcaster processors excluding those in sets Y and Z. Our algorithm defines a schedule on the set S such that each processor from S is guaranteed to belong to T_{k-1} in the round when it has to become idle. This ensures that all useful send operations in the original algorithm remain useful. However this is not the case when a processor in Y and Z becomes idle. We prove that each processor does not receive at most one message as a result of this problem. All together, each processor does not know at most $\left\lceil \frac{m}{n-1} \right\rceil + 1$ messages. Therefore, by adding $\left\lceil \frac{m}{n-1} \right\rceil + 1$ matching rounds, all the processors will know all the messages.

6 Simultaneous send/receive

In this section, we describe an optimal algorithm for the problem of broadcasting multiple messages in the simultaneous send/receive model. This algorithm is a by-product of the algorithm for the bidirectional telephone model described in Section 4.

In the simultaneous send/receive model, in every round, each processor can send a message to one processor and receive a message from another. Not all the lower bounds of Section 2 are applicable to the simultaneous send/receive model. The lower bound arguments of Section 2 for odd values of n are not valid in this model. However, the lower bound arguments of Section 2 for even values of n are valid in this model for all values of n, as implied by the following theorem.

Theorem 5 *In the simultaneous send/receive model, broadcasting m messages among n processors requires at least $(m-1) + \lceil \log n \rceil$ rounds.*

For even n, the algorithm in the simultaneous send/receive model is identical to the algorithm in the bidirectional telephone model. For odd n, the following two modifications of the algorithm for $n+1$, which is an even number, give an optimal algorithm in the simultaneous send/receive model.

- One of the processors in set T_{k-1}, which is matched with a processor in T_{k-2}, is identified as a "dummy" processor. (This is the extra processor in the algorithm for $n+1$.) The processor in T_{k-2} which is instructed to send a message to this "dummy" processor in T_{k-1} does not send any message.

- Processor r, which is matched with the broadcaster, sends to the partner of the "dummy" processor, which is in set T_{k-2}, the message that this processor was supposed to receive from the "dummy" processor.

Note that processor r indeed knows the message it is instructed to send since it belongs to set T_{k-1} — the same set that the "dummy" processor belongs to. Also note that the processors of set T_{k-1} which are matched with the processors in set T_{k-2} remain in set T_{k-1} in the next round. Therefore, the "dummy" processor can be in set T_{k-1} during the entire duration of the algorithm (i.e., it could be the same processor throughout the algorithm).

The next theorem follows from the above modifications to the algorithm of Section 4 and from Theorem 4.

Theorem 6 *Broadcasting m messages among n processors in the simultaneous send/receive model can be done in $(m-1)+\lceil \log n \rceil$ rounds.*

Observe that the algorithm in the simultaneous send/receive model for an even number of processors employs a perfect matching in each round. For an odd number of processors, the algorithm employs a perfect matching on $n-3$ processors excluding the broadcaster. The only deviation from a matching is that the processor that receives a message from the broadcaster sends a message to a third processor.

References

[1] A. Bar-Noy and S. Kipnis, "Designing broadcasting algorithms in the postal model for message-passing systems", to appear in *Mathematical Systems Theory*, 1994. Also Appeared in *Proceedings of the 4th Annual Symposium on Parallel Algorithms and Architectures*, ACM, June 1992, pp. 13–22.

[2] A. Bar-Noy and S. Kipnis, "Multiple message broadcasting in the postal model", *Proceedings of the 7th International Parallel Processing Symposium*, IEEE, April 1993, pp. 463–470.

[3] A. Bar-Noy and S. Kipnis, "Broadcasting multiple messages in simultaneous send/receive systems", to appear in *Discrete Applied Mathematics*, 1994. Appeared in *Proceedings of the 5th Symposium on Parallel and Distributed Processing*, IEEE, December 1993, pp. 344–347.

[4] J. Bruck, R. Cypher, L. Gravano, A. Ho, C.T. Ho, S. Kipnis, S. Konstantinidou, M. Snir and E. Upfal, "Survey of routing issues for the Vulcan parallel computer", *IBM Research Report*, RJ-8839, June 1992.

[5] J. Bruck, R. Cypher, and C.T. Ho, "Multiple message broadcasting with generalized Fibonacci trees", *Proceedings of the 4th Symposium on Parallel and Distributed Processing*, IEEE, December 1992, pp. 424–431.

[6] I. Cidon and I. Gopal, "PARIS: an approach to integrated high-speed private networks", *International Journal of Digital and Analog Cabled Systems*, Vol. 1, No. 2, April–June 1988, pp. 77–85.

[7] D. Clark, B. Davie, D. Farber, I. Gopal, B. Kadaba, D. Sincoskie, J. Smith, and D. Tennenhouse, "The AURORA gigabit testbed", *Computer Networks and ISDN*, 1991.

[8] E. Cockayne and A. Thomason, "Optimal multimessage broadcasting in complete graphs," *Proceedings of the 11th SE Conference on Combinatorics, Graph Theory, and Computing*, 1980, pp. 181–199.

[9] A. M. Farley, "Broadcast time in communication networks," *SIAM Journal on Applied Mathematics*, Vol. 39, No. 2, October 1980, pp. 385–390.

[10] W. Groscup, "The Intel Paragon XP/S supercomputer", *Proceedings of the 5th ECMWF Workshop on the use of Parallel Processors in Meteorology*, 1993, pp. 173–187.

[11] S. M. Hedetniemi, S. T. Hedetniemi, and A. L. Liestman, "A survey of gossiping and broadcasting in communication networks", *Networks*, Vol. 18, No. 4, 1988, pp. 319–349.

[12] W. D. Hillis and L. W. Tucker, "The CM 5 connection machine: a scalable supercomputer", *Communications of the ACM*, Vol. 36, No. 11, November 1993, pp. 30–40.

[13] C.T. Ho, "Optimal multiple messages broadcasting in the send/receive model", personal communication, December 1993.

[14] C.T. Ho, "Optimal multiple messages broadcasting in the multi-port send/receive model", personal communication, December 1993.

[15] C.T. Ho, "Optimal broadcasting on SIMD hypercubes without indirect addressing capability," *Journal on Parallel and Distributed Computing*, Vol. 13, No. 2, October 1991, pp. 246–255.

[16] R. Karp, A. Sahay, E. Santos, and K. E. Schauser, "Optimal broadcast and summation in the LogP model", *Proceedings of the 5th Annual Symposium on Parallel Algorithms and Architectures*, ACM, June 1993, pp. 142–153.

[17] M. V. Schmidt, "Efficient parallel communication with the nCUBE 2S processor", *Parallel Computing*, Vol. 20, No. 4, April 1994, pp. 509–530.

[18] C. B. Stunkel, D. G. Shea, B. Abali, M. M. Denneau, P. H. Hochschild, D. J. Joseph, B. J. Nathanson, M. Tsao, and P. R. Varker, "Architecture and implementation of Vulcan", *Proceedings of the 8th International Parallel Processing Symposium*, IEEE, April 1994, pp. 268–274.