Optimal Broadcast and Summation in the LogP Model

Richard Karp
Abhijit Sahay
Eunice Santos

Computer Science Division,
University of California, Berkeley

Abstract

We consider several natural broadcasting problems for the LogP model of distributed memory machines recently proposed by Culler et al. For each of these problems, we present algorithms that yield an optimal communication schedule. Our algorithms are absolutely best possible in that non even the constant factors can be improved upon. We also devise an (absolutely) optimal algorithm for summing a list of elements (using a non-commutative operation) using one of the optimal broadcast algorithms.

1 Introduction

Most models of parallel computation reflect the communication bottlenecks of real parallel machines inadequately. The PRAM [9], for example, allows inter-processor communication at zero cost. Researchers have proposed several variations on the PRAM [14, 13, 15, 1, 2] that address different communication issues to different degrees. There have also been a number of studies of communication requirements of algorithms on specific networks of processors such as the hypercube or the mesh.

Recently, Culler et al [7] have proposed a general purpose model, the *LogP model* for distributed memory machines. In this model, processors communicate by point-to-point messages which travel through a network. The network imposes costs on communication due to its limited bandwidth, the processors' network interface overhead and the latency that messages incur in traversing paths in the network. The LogP model treats the network as a black box with specified latency, overhead and bandwidth parameters and is thus applicable to many distributed memory machines. For a justification of this model and a more detailed description, we refer the reader to the original paper [7].

In this paper we consider some fundamental problems in parallel computation and describe algorithms that solve them optimally in the LogP model. The problems we consider are various kinds of *broadcast* and *summation*. While our algorithms for optimal broadcast can be described directly, the proof of optimality for our summation algorithm requires an "inverted" problem formulation: instead of determining the minimum time to sum n operands on P processors, we determine the maximum number of operands that can be summed by P processors in t steps. The solution to both problems involves a generalized Fibonacci recurrence relation with parameters and initial conditions that depend on the L, o, g and P.

Given the considerable importance of broadcasting problems in parallel and distributed computation, several variations of it have been well studied in the literature[5, 10]. Much of this work has focused on the design of efficient algorithms for broadcasting on specific networks such as rings, trees, meshes and hypercubes [11, 12]. For fully connected systems (such as those modeled by LogP) broadcast problems have been studied in several communication models, but without latency. Cockayne and Thomason [6] and Farley [8] gave optimal algorithms for a model where each processor can either send or receive a message in one time step. Alon, Barak and Manber [3] studied reliable broadcast in a model that allows simultaneous send and receive. Bar-Noy and Kipnis [4] introduced the *postal model* which incorporates a latency parameter and solved the single message broadcast problem in it. The postal model turns out to be a special cast of *LogP* (with o=0 and $g = 1$) and is quite adequate for studying many communication problems. If we incorporate the send/receive overhead of the LogP model into the latency parameter and normalize the unit of time so that $g = 1$, we are essentially reduced to the postal model. In some later sections, we will find it more convenient to work with the postal model.

The rest of the paper is organized as follows. Section 2 describes the LogP model of parallel computation. Section 3 defines the P-way broadcast problem and an optimal algorithm for it. We also present a running-time analysis of our optimal algorithm. Section 4 describes the summing problem, its inversion and our optimal algorithm for summing. Section 5 returns to other broadcast problems and optimal algorithms for them. Some concluding remarks are made in Section 6.

2 The LogP Model

LogP is a model for distributed-memory multiprocessors in which processors communicate by point-to-point messages [7]. The model specifies the performance characteristics of the interconnection network, but does not describe the structure of the network.

The main parameters of the model are:

P: the number of processor/memory modules. Local operations execute in unit time (a processor cycle).

L: an upper bound on the *latency*, or delay, incurred in communicating a message containing a word (or small number of words) from its source module to its target module.

o: the *overhead*, defined as the length of time that a processor is engaged in the transmission or reception of each message; during this time, the processor cannot perform other operations.

g: the *gap*, defined as the minimum time interval between consecutive message transmissions or consecutive message receptions at a processor. The reciprocal of g corresponds to the available per-processor communication bandwidth.

Furthermore, it is assumed that the network has a *finite capacity*, such that at most $\lceil L/g \rceil$ messages can be in transit from any processor or to any processor at any time. If a processor attempts to transmit a message that would exceed this limit, it stalls until the message can be sent without exceeding the capacity limit.

The parameters L, o and g are measured as multiples of the processor cycle. The model is *asynchronous*, i.e., processors work asynchronously and the latency experienced by any message is unpredictable, but is bounded above by L in the absence of stalls. Because of variations in latency, the messages directed to a given target module may not arrive in the same order as they are sent. In order to be considered correct, an algorithm must produce correct results under all interleavings of messages consistent with the upper bound of L on latency.

In estimating the running time of the algorithms, we assume that that all processors work synchronously and that each message incurs precisely a latency of L. All algorithms presented in this paper satisfy the capacity constraint of the LogP model, and we do not mention it henceforth.

3 Optimal Broadcast

3.1 The Broadcast Problem

The *P-way broadcast* problem is that of finding a schedule of communication among P processors (numbered $1, \ldots, P$) so that a datum initially available at processor 1 is made available to all P processors in the shortest possible time.

Definition 3.1 *Let $\mathcal{A}$ be an algorithm for P-way broadcast. The* delay *of processor i in $\mathcal{A}$, denoted $t_{\mathcal{A}}(i)$, is defined as the time at which the datum is first available at processor i in $\mathcal{A}$. The running time of $\mathcal{A}$, denoted $t_{\mathcal{A}}$, is*

$$t_{\mathcal{A}} = \max_{1 \le i \le P} \{t_{\mathcal{A}}(i)\}$$

The complexity of P-way broadcast in the LogP model is defined as

$$B(P; L, o, g) = \min_{\mathcal{A}} t_{\mathcal{A}}$$

$\mathcal{A}$ is optimal for P-way broadcast if $t_{\mathcal{A}} = B(P; L, o, g)$.

Definition 3.2 *Given $t \ge 0$, the number of time steps available, let*

$$P(t; L, o, g) = \max\{i : B(i; L, o, g) \le t\}$$

denote the maximum number of processors that can be reached by a broadcast algorithm in t steps.

When there is no danger of confusion, we will omit explicit mention of L, o and g and write $B(P)$ and $P(t)$ instead of $B(P; L, o, g)$ and $P(t; L, o, g)$. Note that for any broadcast algorithm $\mathcal{A}$, $t_{\mathcal{A}}(1) = 0$.

3.2 An Optimal Broadcast Algorithm

In this section, we develop an algorithm for P-way broadcast that takes $B(P)$ steps. Our algorithm is simple and intuitive: all informed processors send messages repeatedly to uninformed processors as early and as frequently as possible. We establish optimality of our algorithm by arguing that among optimal algorithms for broadcast, there must be one which is minimal in the following senses: there are no redundant messages and there are no unforced delays in sending messages. We then show a simple rule for finding the (essentially unique) optimal minimal algorithm. These ideas are made more precise below.

Definition 3.3 *A broadcast algorithm is said to be* minimal *if*

1. *No processor receives more than one message.*
2. *If processor p sends messages to processors $p_0, \ldots, p_k$ (in that order) then $t_{\mathcal{A}}(p_j) = t_{\mathcal{A}}(p) + jg + L + 2o,\ 0 \leq j \leq k$*

Lemma 3.1 *Given algorithm $\mathcal{A}$ for broadcast, there exists a minimal algorithm $\mathcal{B}$ which reaches as many processors as $\mathcal{A}$ and for which $t_{\mathcal{B}} \leq t_{\mathcal{A}}$.*

Proof: If any processor receives more than one message in $\mathcal{A}$ then the algorithm $\mathcal{A}'$ derived from $\mathcal{A}$ by discarding all but the first message to any processor completes no later than $\mathcal{A}$ and reaches as many processors as $\mathcal{A}$. Also, if in $\mathcal{A}'$, processor p initiates messages to processors $p_0, \ldots, p_k$ at times $T_0, \ldots, T_k$ (in order), we must have $T_0 \geq t_{\mathcal{A}'}(p)$ (since the datum is not available at p earlier than $t_{\mathcal{A}'}(p)$) and $T_{i+1} \geq T_i + g$ because of the model's bandwidth constraint. However, if any of these inequalities is strict, consider the algorithm $\mathcal{A}''$ derived from $\mathcal{A}'$ by having $T_{i+1} = T_i + g$. Clearly, no processor has a larger information time in $\mathcal{A}''$ than in $\mathcal{A}'$ (and in fact, at least one processor has a strictly smaller information time.)

It is clear that after a finite sequence of such transformations, all the inequalities will be strict and we will have a minimal algorithm which completes no later than $\mathcal{A}$ and reaches as many processors as $\mathcal{A}$. □

3.2.1 Broadcast Trees

We shall often find it useful to think of a minimal broadcast algorithm $\mathcal{A}$ in terms of its *broadcast tree* of processors, $T_{\mathcal{A}}$. $T_{\mathcal{A}}$ is a rooted, ordered tree with a node for each processor that participates in the broadcast. The root of $T_{\mathcal{A}}$ is processor 1 (the source of the broadcast) and if processor p sends messages to processors $p_0, \ldots, p_k$ (in that order) in $\mathcal{A}$, then in $T_{\mathcal{A}}$ node p has $p_0, \ldots, p_k$ as its (ordered) children. If, in addition, we label nodes by the information times of the corresponding processors, minimality implies that a parent's label is exactly $L + 2o$ smaller than its oldest child's, while the labels of successive siblings differ by exactly g.

Definition 3.4 *The* universal minimal broadcast tree*, denoted $\mathcal{B}$, is defined to be the infinite labeled ordered tree in which the root has label 0 and a node with label t has children labeled $t + ig + L + 2o,\ i \geq 0$.*

Lemma 3.2 *For any minimal broadcast algorithm $\mathcal{A}$, the broadcast tree $T_{\mathcal{A}}$ is a rooted subtree of $\mathcal{B}$ with the label on a node corresponding to the information time of the corresponding processor.*

Proof: Follows from definition of $\mathcal{B}$ and minimality. □

From Lemma 3.1 we know that the optimal minimal broadcast algorithm reaches P processors in time $B(P)$. Lemma 3.2 characterizes all minimal broadcast algorithms. We thus have the following simple algorithm for optimal P-way broadcast.

Definition 3.5 *Let $\mathcal{B}(P)$ be the rooted subtree of $\mathcal{B}$ consisting of the P nodes with smallest labels (ties being broken arbitrarily.)*

Theorem 3.1 *$\mathcal{B}(P)$ is optimal for P-way broadcast.*

Proof: By construction, every minimal broadcast tree will have a node with information time at least as large as the largest information time for $\mathcal{B}(P)$. $\square$

3.3 Running time analysis

We will carry out the analysis in the postal model (by incorporating o into the latency and normalizing so that g is 1.)

Definition 3.6 *Let $L > 0$ be a fixed integer. Define the sequence $\{f_i\}$ by:*

1. $f_i = 1$ *for* $0 \le i < L$.
2. $f_i = f_{i-1} + f_{i-L}$ *otherwise.*

Fact 3.1 *For each t,* $1 + \Sigma_{i=0}^{i=t} f_i = f_{t+L}$.

Theorem 3.2 *For $t \ge 0$ and $L > 0$,* $p(t; L, 0, 1) = f_t$

Proof: Consider the first t steps of our optimal broadcast algorithm with $o - 0$ and $g = 1$. Clearly, if $t < L$, the theorem holds. Otherwise, the last message is sent at time $t - L$. We claim that the number of messages sent at time i is f_i. This is clearly true for $i < L$ since only the source can send messages. For larger t, messages are sent at time t by every processor that sent a message at time $t - 1$ and by processors who received their messages at time t. Our claim follows by induction.

Thus the number of processors for t-step optimal broadcast is given by $1 + \Sigma_{i=0}^{t-L} f_i = f_t$.$\square$

It is not hard to establish that there is a constant γ (that depends only on L), $1 < \gamma \le 2$, such that $\gamma^{t-L+1} \le f_t \le \gamma^t$. Thus, Theorem 3.2 implies that the broadcast time for P processors is at most $L - 1 + \log_\gamma P$. Denormalizing to the complete LogP model, we get the following.

Theorem 3.3 *The time for optimal broadcast of a single item in the LogP model $B(P; L, o, g)$ is at most $L + 2o + g \log_\gamma P$ where $\gamma \in (1, 2]$ is the only real root of the equation $x^d - x^{d-1} - 1 = 0$ with $d = \lceil (L + 2o)/g \rceil$.*

3.4 Σ-Optimality

In this section, we prove a property of $\mathcal{B}(P)$ that is crucial to our optimal summation algorithm of Section 4.

Definition 3.7 *Let $\mathcal{A}$ be an algorithm for P-way broadcast. We say that $\mathcal{A}$ is* Σ-optimal *if it minimizes* $\sum_{i=1}^{P} t_{\mathcal{A}}(i)$.

Even though the objective in Σ-optimal algorithms is quite different – minimization of the sum of information times rather than the maximum information time – it turns out that $\mathcal{B}(P)$ is also Σ-optimal for P-way broadcast.

Theorem 3.4 *Algorithm $\mathcal{B}(P)$ is Σ-optimal for P-way broadcast.*

Proof: The arguments used in the proof of Lemma 3.1 are easily seen to imply that there is a minimal broadcast algorithm which is Σ-optimal. Let $b_1 \leq b_2 \leq \ldots \leq b_P$ denote the information times of the processors in $\mathcal{B}(P)$ and let $\mathcal{A}$ be a minimal algorithm for P-way broadcast with information times $a_1 \leq \ldots \leq a_P$. By construction, $b_i \leq a_i$ for each i and Σ-optimality of $\mathcal{B}(P)$ follows. □

4 Optimal Summation

In this section, we consider the problem of computing the sum of n operands in the LogP model. The input to the problem is a set of operands $X_1, \ldots, X_n$ and we assume that an algorithm can choose how these are initially distributed among the P processors. We assume that the addition operation is commutative [1] and that each addition operation takes unit time. Our algorithm to compute $X_1 + \ldots + X_n$ in the shortest possible time can be thought of as consisting of a local phase (during which each processor performs local additions) followed by a 'reverse broadcast' or reduction in which the P partial sums are combined. Our proof of optimality is based on the following inversion of the problem: instead of finding an algorithm to add n operands in minimum time, find an algorithm that adds the maximum number of operands in t units of time.

4.1 Summation Trees

Consider an algorithm $\mathcal{A}$ that computes $X = X_1 + \ldots + X_n$ in time t. Let p be a processor at which X is available at time t. We may assume that p performs an addition step $X = Y_L + Y_R$ in the t-th time step, with $Y_L = X_1 + \ldots + X_i$ and $Y_R = X_{i+1} + \ldots + X_n$ for some $i \in \{1, \ldots, n-1\}$. Clearly, each of the partial sums Y_L and Y_R must be available at p by time $t-1$, either as a result of local computation or in the form of a message received from another processor. In each case, we can identify the addition steps that resulted in the creation of these partial sums.

We can thus think of $\mathcal{A}$'s computation as a binary tree, $T_{\mathcal{A}}$, induced by the order of additions. Each internal node of $T_{\mathcal{A}}$ represents an addition performed by $\mathcal{A}$ in the computation of X and each leaf corresponds to some input operand X_i. The root of $T_{\mathcal{A}}$ represents the addition $X = Y_L + Y_R$ identified above, and the left and right children of the root represent the additions that resulted in the creation of Y_L and Y_R at the processors from which p received them during $\mathcal{A}$'s computation. (Of course, neither of these processors need be distinct from p.) In general, if an internal node v in $T_{\mathcal{A}}$ represents an addition $Y_l + Y_r$ at processor p_k its left and right children represent respectively the additions resulting in the partial sums Y_l and Y_r at the processors from which p_k received them; if such a partial sum is one of the X_i, the corresponding node is a leaf of $T_{\mathcal{A}}$.

As with broadcast algorithms, Lemma 4.1 below shows that the search for optimal summing algorithms can be confined to algorithms of a certain canonical form. For the purpose of the following definition, we will assume that the processor responsible for the final addition in $\mathcal{A}$ (at time t) initiates a message transmission at time t.

Definition 4.1 *A summation algorithm $\mathcal{A}$ is called* busy-lazy *if*

1. *No processor sends more than one message.*
2. *No processor idles until it sends a message. (A processor is said to idle at time i if it is neither adding nor receiving a message during the i-th time period.)*

[1] Our optimal algorithm for commutative summation can be used for non-commutative summation with an appropriate renumbering of the operands.

3. *If processor p initiates message receptions at times $R_p(1) < \ldots < R_p(k)$ and a message transmission at time S_p, then $R_p(j) = S_p - (o+1) - (k-j)g$.*

Lemma 4.1 *Given algorithm $\mathcal{A}$ for summation, there exists a busy-lazy algorithm $\mathcal{B}$ which sums at least as many operands as $\mathcal{A}$ and takes no longer than $\mathcal{A}$.*

Proof: Suppose that some processor sends messages $S_1, \ldots, S_k$ $(k > 1)$ at time steps $T_1, \ldots, T_k$ in $\mathcal{A}$. Identifying these messages with the corresponding nodes in $T_{\mathcal{A}}$, we see that S_i and S_j must represent sums of disjoint sets of input operands. We can modify algorithm $\mathcal{A}$ so that for $i < k$, it adds S_i to some leaf of S_k at time T_i and sends only the (new) message S_k at time T_k. (Processors that receive S_i in $\mathcal{A}$ ignore those messages in the modified algorithm.) It is clear that the modified algorithm takes no longer than $\mathcal{A}$ and sums the same number of operands.

Secondly, note that a processor that idles at some time step can always be made to perform an addition instead (with a fresh input as one of the operands,) yielding an algorithm which sums more operands in the same number of time steps.

Finally, consider a processor p that initiates receptions at times $R_p(1) < \ldots < R_p(k)$ and a transmission at time S_p. Clearly, $S_p \geq R_p(j) + o + 1 + (k-j)g$, $j = 1, \ldots, k$. But if any of these inequalities is strict, some message reception at p can be delayed by at least one unit, allowing the processor that sends this message to send a partial sum of more operands. The modified algorithm again sums more operands than the original in the same number of time steps.□

Let $\mathcal{A}$ be a busy-lazy summation algorithm that runs for t steps and in which the processors initiate their message transmissions at times $S_1 \leq S_2 \leq \ldots \leq S_P = t$. Consider the P-way broadcast algorithm $\mathcal{A}'$ resulting from the following reversal of each message of $\mathcal{A}$: if processor i initiates transmission to processor j at time S_i in $\mathcal{A}$, processor i completes reception from processor j at time $t - S_i$ in $\mathcal{A}'$. Observe that $\mathcal{A}'$ is a P-way broadcast algorithm in which sending processors "wait" for one time step before transmissions. (This corresponds in $\mathcal{A}$ to the steps that receiving processors spend just after receptions in adding the received partial sums.) The algorithm $\mathcal{A}'$ can be viewed as a minimal broadcast algorithm for a machine which has a send overhead of $o+1$ (or equivalently, a latency of $L+1$.) Thus, busy-lazy summation algorithms for a machine with parameters L, o, g, P are in one-to-one correspondence with minimal broadcast algorithms for a machine with parameters $L+1, o, g, P$. Moreover, if the busy-lazy algorithm runs for t steps and has message initiation time S_i for processor i, the corresponding broadcast algorithm will have information time $t - S_i$ for processor i.

Lemma 4.2 *Let $\mathcal{A}$ be a busy-lazy summation algorithm that adds n operands in t steps and in which the processors initiate their message transmissions at times $S_1 \leq S_2 \leq \ldots \leq S_P = t$. Then $n = \Sigma_{i=1}^{P} S_i - op + 1$*

Proof: Let k_i be the number of messages received by processor i. Then, the number of input operands that are tackled by processor i is exactly $S_i - (o+1)k_i + 1$. Summing over the processors yields the result.□

This lemma characterizes optimal t-step busy-lazy summation algorithms as those which maximize ΣS_i. From our correspondence with minimal broadcast algorithms, we see that the corresponding broadcast algorithms must be those that minimize $\Sigma(t - S_i)$, viz. Σ-optimal broadcast algorithms.

We are thus led to the following algorithm for t-step summation assuming that $t \geq B(P; L+1, o, g)$:

Algorithm $\mathcal{S}(t)$:

1. For processor i, let t_i be the information time in optimal P-way broadcast with parameters $L+1, o, g$.
2. Processor i adds locally for $t - t_i$ steps and then participates in a P-way reduction (for t_i steps).

We have proved the following:

Theorem 4.1 *For $t > B(P; L+1, o, g)$, $\mathcal{S}(t)$ is optimal for t step summation.*

5 Generalized Broadcast Problems

In this section, we generalize the broadcast problem in two directions and give optimal algorithms for each generalized problem. The generalizations of P-way broadcast that we consider are based on whether the broadcast is “one-to-all" or “all-to-all", and whether the data originating at different processors can be combined. In the rest of this section, we describe these variations in more detail and present our algorithms for solving them.

5.1 All-to-All Broadcast

In the P-way all-to-all broadcast problem, each of P processors has a data item that is to be made available to every processor. Of course, one could solve this problem using P one-to-all broadcasts but a more efficient (and simpler) solution is possible.

Observe that since each processor must receive $(P-1)$ items and the first one cannot be received until time $L+2o$, a lower bound on the time for P-way all-to-all broadcast is $L+2o+(P-2)g$. On the other hand, if processor i sends out its data item to processors $i+1 \pmod P), \ldots, i+(P-1) \pmod P)$ (in that order) at time $0, g, \ldots, (P-2)g$, each processor would receive messages at time $L+2o, L+2o+g, \ldots, L+2o+(P-2)g$. We thus have a simple optimal algorithm for all-to-all broadcast.

The lower bound argument and our algorithm extends easily to the situation where each of the P processors has k items to broadcast: since a processor must receive $k(P-1)$ items and the first one cannot be received until $L+2o$, we have a lower bound of $L+2o+g(k(P-1)-1)$. This bound is matched by k repetitions of the previous algorithm.

We remark that the order of transmission for processor i does not have to be the one indicated above. Any collection of permutations of the set $S=\{1,\ldots,P\}$, one for each processor, such that no processor is the target of two messages at the same time, will yield an optimal algorithm. (For the k-item broadcast, analogous permutations of the multiset consisting of k copies of S will lead to optimality.)

5.2 Broadcast With Combining

In parallel computation, one frequently encounters situations where each processor holds a value and the P values are to be combined into one (or *reduced*) using some simple operation (such as max or $+$.) Indeed, we have seen an example of such a situation in Section 4 where we gave an optimal algorithm for summation based on an optimal algorithm for reduction. As indicated in that context, reduction can be viewed as “all-to-one" broadcast (with a slight change in model parameters) and is thus solved optimally by simply reversing the directions of messages in optimal broadcast.

If the reduced value is to be made available to all processors, we get a problem which we may think of as an all-to-all broadcast with combining. Clearly, this problem can be solved by a reduction followed by broadcast, which is optimal to within a factor of 2. However, we show below that all-to-all broadcast with combining takes no longer than all-to-one reduction. Our algorithm is based on the observation that optimal reduction exhibits a great deal of imbalance in terms of the number of messages received by processors. We show how a judicious replication of messages (and computation) allows each processor to compute the reduced value at the same time as the sink processor in all-to-one reduction.

5.2.1 An Optimal Algorithm for Broadcast with Combining

Let x_i be a value initially available at processor $i,\ i=0,\ldots,P-1$. The problem is to make $x_0+\ldots+x_{P-1}$ available to each processor in the shortest possible time. The ‘+’ operation is assumed to be commutative. In the sequel, all arithmetic on processor indices will be modulo P and for indices $i,j,\ x[i:j]$ will denote the quantity $x_j + x_{j-1} + \ldots + x_i$. Thus, for example, if $P = 5$, $x[1 : 3] = x_3 + x_2 + x_1$ while $x[3 : 1] = x_1 + x_0 + x_4 + x_3$.

We will carry out the analysis in the postal model and with the assumption that the combining operation takes no time. With these assumptions, it is clear that if a message is initiated to processor i at time t, it can be received at time $t + L$, combined with x_i and the result transmitted to another processor at time $t + L$ (arriving at its destination at time $t + 2L$.)

Our algorithm will be described in terms of the sequence $\{f_i\}$ defined in Section 3.3. Let T, the amount of time for the algorithm be fixed, and let $P = p(T; L, 0, 1)$. Our algorithm has the following simple description: at time $j = 0, 1, \ldots, T - L$, processor i, $i = 0, \ldots, P - 1$ sends its current value to processor $i + f_{j+L-1}$. (As indicated earlier, the values sent at time j arrive at their destinations at time $j + L$ and are instantaneously combined into the current value at the destination processor before transmission at time $j + L$.)

Theorem 5.1 *The algorithm presented above leaves the value $x[0 : P - 1]$ at each of the $P = p(T; L, 0, 1)$ processors at time T.*

Proof: We observe first that the model constraints on the rate of reception are met since no processor is the target of more than one message during any time step. Thus, the algorithm terminates at time T. To show that every processor has $x[0 : P - 1]$ when the algorithm terminates, we inductively prove that at time j processor i has the value $x[i - f_j + 1 : i]$. Thus, at time T, processor i holds $x[i - f_T + 1 : i]$. By Theorem 3.2, $f_T = P$ proving our theorem.

For the basis of the induction, note that by definition $f_j = 1$ for $j < L$ and each processor certainly has its own value x_i. Thus, the inductive claim is true for $j < L$. Consider now some value $t \geq L$ and assume that the claim holds for all $j < t$. At time step t, processor i would receive a message initiated at time $(t - L)$ by processor $i - f_{(t-L)+L-1} = i - f_{t-1}$. By inductive hypothesis, this message carries the value $x[i - f_{t-1} - f_{t-L} + 1 : i - f_{t-1}] = x[i - f_t + 1 : i - f_{t-1}]$. Processor i can combine this with the value it holds at time $t - 1$, viz. $x[i - f_{t-1} + 1 : i]$, to form $x[i - f_t + 1 : i]$ at time t. $\Box$

6 Conclusion

We have presented optimal algorithms for a variety of broadcast problems in the LogP model in which the only communication primitive is point-to-point message passing. Our algorithms for broadcasting an item from a single source, broadcasting an item (or several items) from all P processors and broadcasting a single item from all P processors with combining, are all optimal even in the constant factors. In addition, our broadcast algorithm can be "time-reversed" to yield an optimal algorithm for summing a list of numbers optimally. Some problems that we are currently engaged in studying are optimal broadcasting of multiple messages and optimal parallel prefix computation.

References

[1] A Aggarwal, A. K. Chandra, and M. Snir. On Communication Latency in PRAM Computation. In *Proceedings of the ACM Symposium on Parallel Algorithms and Architectures*, pages 11–21. ACM, June 1989.

[2] A. Aggarwal, A. K. Chandra, and M. Snir. Communication Complexity of PRAMs. In *Theoretical Computer Science*, pages 3–28, March 1990.

[3] N. Alon, A. Barak, and U. Manber. On dissiminating infomation reliably without broadcasting. In *Proceedings of the International Conference on Distributed Computing Systems*, 1987.

[4] A. Bar-Noy and S. Kipnis. Designing Broadcasting Algorithms in the Postal Model for Message-Passing Systems. In *Proceedings of the ACM Symposium on Parallel Algorithms and Architectures*, pages 11–22, June 1992.

[5] D. P. Bertsekas and J. N. Tsitsiklis. *Parallel and Distributed Computation: Numerical Methods*. Prentice-Hall, 1989.

[6] E. Cockayne and A. Thomason. Optimal multi-message broadcasting in complete graphs. In *Proceedings of the 11th SE Conference on Combinatorics, Graph Theory, and Computing*, pages 181–199, 1980.

[7] D. E. Culler, R. M. Karp, D. A. Patterson, A. Sahay, K. E. Schauser, E. Santos, R. Subramonian, and T. von Eicken. LogP: Towards a Realistic Model of Parallel Computation. In *Fourth ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming*, May 1993. (to appear) Also appears as UCB/CS/92 713 report.

[8] A. M. Farley. Broadcast time in communication networks. *SIAM Journal on Applied Mathematics*, 39(2):385–390, October 1980.

[9] S. Fortune and J. Wyllie. Parallelism in Random Access Machines. In *Proceedings of the 10th Annual Symposium on Theory of Computing*, pages 114–118, 1978.

[10] S. M. Hedetniemi, S. T. Hedetniemi, and A. L. Liestman. A Survey of Gossiping and Broadcasting in Communication Networks. *Networks*, 18(4):319–349, 1988.

[11] C-T. Ho. Optimal Comunication Primitives and Graph Embeddings on Hypercubes. In *Ph.D. Thesis, Yale University*, 1990.

[12] C-T. Ho and S.L. Johnsson. Distributed routing algorithms for broadcasting and personalized communication in hypercubes. In *Proceedings of the 1986 International Conference on Parallel Processing*, pages 640–648. IEEE Computer Society, 1986.

[13] R. M. Karp, M. Luby, and F. Meyer auf der Heide. Efficient PRAM Simulation on a Distributed Memory Machine. In *Proceedings of the Twenty-Fourth Annual ACM Symposium of the Theory of Computing*, pages 318–326. ACM, ACM, May 1992.

[14] K. Mehlhorn and U. Vishkin. Randomized and deterministic simulations of PRAMs by parallel machines with restricted granularity of parallel memories. *Acta Informatica*, 21:339–374, 1984.

[15] C. H. Papadimitriou and M. Yannakakis. Towards an Architecture-Independent Analysis of Parallel Algorithms. In *Proceedings of the Twentieth Annual ACM Symposium of the Theory of Computing*, pages 510–513. ACM, 1988.