Alcom-FT Technical Report Series
ALCOMFT-TR-01-70

Optimal Broadcast on Parallel Locality Models*

Ben H.H. Juurlink†
Laboratory of Computer Architecture and Digital Technique
Faculty of Information Technology and Systems
Delft Technical University
Mekelweg 4, 2628 CD Delft, The Netherlands

Petr Kolman†
Institute for Theoretical Computer Science and
Department of Applied Mathematics
Charles University
Malostranské nám. 25, 118 00 Prague, Czech Republic

Friedhelm Meyer auf der Heide
Heinz Nixdorf Institute and
Department of Mathematics and Computer Science
Paderborn University
Fürstenallee 11, 33102 Paderborn, Germany

Ingo Rieping
Heinz Nixdorf Institute and
Department of Mathematics and Computer Science
Paderborn University
Fürstenallee 11, 33102 Paderborn, Germany

Abstract

In this paper matching upper and lower bounds for broadcast on general purpose parallel computation models that exploit network locality are proven. These models try to capture both the general purpose properties of models like the PRAM or BSP on the one hand, and to exploit network locality of special purpose models like meshes, hypercubes, etc., on the other hand. They do so by charging a cost $l(|i-j|)$ for a communication between processors i and j, where l is a suitably chosen latency function.

An upper bound $T(p) = \sum_{i=0}^{\log\log p} 2^i \cdot l(p^{1/2^i})$ on the runtime of a broadcast on a p processor H-PRAM is given, for an arbitrary latency function $l(k)$.

The main contribution of the paper is a matching lower bound, holding for all latency functions in the range from $l(k) = \Omega(\log k / \log\log k)$ to $l(k) = O(\log^2 k)$. This is not a severe restriction since for latency functions $l(k) = O(\log k / \log^{1+\epsilon} \log(k))$ with arbitrary $\epsilon > 0$, the runtime of the algorithm matches the trivial lower bound $\Omega(\log p)$ and for

*Partially supported by the DFG-Sonderforschungsbereich 376, the DFG-Graduiertenkolleg 124, and the IST Programme of the EU under contract number IST-1999-14186 (ALCOM-FT). A preliminary version of this paper appeared in *Proceedings of the 7th Colloquium on Structural Information and Communication Complexity (SIROCCO)*, pages 211–225, 2000.

†This research was conducted in part while the author was staying at Heinz Nixdorf Institute in Paderborn.

$l(k) = \Theta(\log^{1+\epsilon} k)$ or $l(k) = \Theta(k^\epsilon)$, the runtime matches the other trivial lower bound $\Omega(l(p))$. Both upper and lower bounds apply for other parallel locality models like Y-PRAM, D-BSP and E-BSP, too.

1 Introduction

An impressive amount of work was dedicated to investigations of explicit network topologies (e.g., hypercubes, meshes, trees, butterflies, De Bruijn networks, expanders) and to designing parallel algorithms that exploit locality of the respective topology (Leighton [13] gives a good survey). As algorithms designed for a specific topology are portable to other topologies only up to a certain extent (e.g., [14]) and classical parallel computational models (PRAM, e.g.) do not consider communication cost, general purpose models that do account for communication cost were defined. Famous examples are the BSP [18], the LogP [5], and the QSM model [1]. However, they account for the communication in a uniform way - no locality is considered. As experiments showed that locality is an issue that influences runtime of algorithms significantly [12], models were designed that try to keep the "general purpose" character while exploiting the topological locality. Examples of these models are H-PRAM [7], D-BSP [17] or E-BSP [11]. We refer to them as *parallel locality models*. Roughly speaking, these models try to capture network locality by charging a communication between processors i and j with $l(|i-j|)$ where l is a suitably chosen function. It turns out that for many important networks there exists a numbering π of the nodes and a function l such that for any two nodes u and v of the network $\mathrm{dist}(u,v) \leq l(|\pi(u) - \pi(v)|)$, where $\mathrm{dist}(u,v)$ denotes the distance between u and v in the network (e.g., the Gray code numbering and $l(k) = \log k$ for the hypercube, or the Peano indexing scheme and $l(k) = k^{1/d}$ for the d-dimensional mesh [4]; other examples are given by Juurlink and Wijshoff [11]). Since such an l function reflects well the distance it is called the *latency function*. We only assume that it is a non-decreasing function with $l(1) \geq 1$.

In this paper we study the complexity of the broadcast problem on such locality models. This problem consists of distributing a broadcast item, initially stored in the first processor, to all other processors.

1.1 Known Results

Heywood and Ranka [7] introduced the H-PRAM model. It is an EREW PRAM that accounts for communication with respect to locality. Moreover, to some extent it allows the processors to work asynchronously. Heywood and Ranka described several basic algorithms on this model, e.g. algorithms for broadcast, summation and sorting [8, 6]. For latency function $l(k) = \log k$, the runtime of their broadcast algorithm is $O(\log^{3/2} p)$, and for $l(k) = \sqrt{k}$ it is $O(\sqrt{p})$. Besides the H-PRAM other locality models were proposed, e.g. Y-PRAM [16], D-BSP [17], E-BSP [11], Fat-Tree [2]. More details on these models can be found in Section 2.2. For the Y-PRAM, De la Torre and Kruskal [16] gave broadcast and prefix sum algorithms with runtime $O(\log^2 p)$, for latency function $l(k) = \log k$ which reflects the hypercube abilities. Bilardi et al. [2] dealt with the broadcast problem on fat-trees and gave a few upper and lower bounds for various capacities of fat trees. However, their lower bound technique cannot be applied in our settings since they assume that all processors work synchronously. Another kind of parallel locality model, the Reconfigurable Ring of Processors (RRP), was considered by Rosenberg, Scarano and Sitaraman [15]. An RRP is a ring network with N processors and L parallel communication lines. The communication cost is logarithmic in the distance of two processors. They proved an asymptotically optimal bound $\Theta(\log N \cdot \log\log N)$ for the case $L = N$. This resembles the results in this paper for the latency function $l(k) = \log k$. We extend this result by proving analogous optimal lower bounds for a whole range of latency functions. Furthermore, the lower bound techniques used in [15] cannot be used for the H-PRAM since, though restricted, the H-PRAM is a shared memory model.

A lot of work has been done on communication primitives like broadcast, gossip or summation on explicit topologies (hypercube, mesh, etc.). A survey was written by Hromkovic et al. [9].

1.2 New Results

An upper bound $T(p) = \sum_{i=0}^{\log\log p} 2^i \cdot l(p^{1/2^i})$ on the runtime of our algorithm ROOT for broadcast on the EREW H-PRAM is given. Basically the same technique was used for the fat-trees [2] and for the RRP [15]. The upper bound $T(p)$ turns out to be, e.g., $O(\log p)$ for $l(k) = O(\log k / \log^{1+\epsilon} \log(k))$ with arbitrary $\epsilon > 0$, $O(\log p \cdot \log\log\log p)$ for $l(k) = \log k / \log\log k$, $O(\log p \cdot \log\log p)$ for $l(k) = \log k$, and $O(l(p))$ for $l(k) = \Theta(\log^{1+\epsilon}(k))$ and $l(k) = \Theta(k^\epsilon)$ with arbitrary $\epsilon > 0$. The algorithm ROOT works on other locality models, like D-BSP, E-BSP or Y-PRAM, too.

The main result of this paper is a matching lower bound $\Omega(\sum_{i=0}^{\log\log p} 2^i \cdot l(p^{1/2^i}))$ given for the H-PRAM. This lower bound holds for all latency functions in the range $l(k) = \Omega(\log k / \log\log k)$ and $l(k) = O(\log^2 k)$. This is not a severe restriction since for latency functions $l(k) = O(\log k / \log^{1+\epsilon} \log(k))$ with arbitrary $\epsilon > 0$, the runtime of algorithm ROOT matches the trivial lower bound $\Omega(\log p)$ and for $l(k) = \Theta(\log^{1+\epsilon} k)$ or $l(k) = \Theta(k^\epsilon)$, the runtime matches the other trivial lower bound $\Omega(l(p))$. The lower bound holds even if synchronization on the H-PRAM is for free or if the H-PRAM is extended with a "pipelining" facility - a shared memory write takes, for the writing processor, only one time unit instead of $l(k)$ units (the accessed shared memory cell is updated in $l(k)$ time steps).

For the proof a broadcast tree is defined, a general parallel locality model suitable for proving lower bounds. It enables to transfer the results to other parallel locality models like D-BSP, E-BSP and Y-PRAM, too. A worth mentioning corollary of the lower bound is that the shared memory of the H-PRAM does not represent any advantage over the models with distributed memory, for the broadcast problem.

Among other things, the lower bound $\Omega(\log p \cdot \log\log p)$ for the latency function $l(k) = \log k$ answers a question posted by Bilardi et al. [2]: Are ideal fat-trees with large capacities that are extended with pipelining facilities more powerful than those without pipelining? The answer is "No" since our lower bound matches their upper bound.

The lower bound $\Omega(\log p \cdot \log\log p)$ for the latency function $l(k) = \log k$, which reflects the communication properties of the hypercube, has one more corollary: though the locality models reflect the topological properties of the underlying networks better than the non-local models (e.g., on BSP with hypercube-derived parameters, $\Omega(\log^2 p \,/ \log\log p)$ time is needed for broadcast) they still do not fully capture them. The lower bound still differs by a factor $\Omega(\log\log p)$ from the optimal $O(\log p)$ broadcast algorithm implemented directly on the hypercube, even on the weak one-port model.

1.3 Organization

In Section 2, the H-PRAM and some other parallel locality models are described. Section 3 deals with the algorithm ROOT for broadcast. Finally, in Section 4 the main lower bound for broadcast algorithms is proven. A preliminary version of this paper appeared in the proceedings of the 7th International Colloquium on Structural Information & Communication Complexity (SIROCCO 2000) [10].

2 Locality Models

Parallel locality models considered in this paper were introduced as models that try to capture both the general purpose properties of models like the PRAM or the BSP on the one hand, and to exploit locality of special purpose models like specific networks (meshes, hypercubes) on the other hand. They aim at enhancing general purpose models by introducing some weak form of locality. To do so, they define communication costs as a function in the "distance" between the source and destination of the communication processors. In this section, a variant of locality models is defined, the H-PRAM, and a few other models are briefly sketched.

2.1 The H-PRAM Model

A simplified version of the model is given that is sufficient for our purposes. The *Hierarchical PRAM* or *H-PRAM* [7] is a PRAM that accounts for communication. We deal with the EREW PRAM variant where only exclusive memory accesses are allowed. The cost for accessing the shared memory within an H-PRAM with p processors is $l(p)$ where $l(k)$ is a given non-decreasing latency function with $l(1) \geq 1$. The instruction set is extended with a `partition` instruction. This instruction partitions the H-PRAM into two or more independent H-PRAMs of smaller sizes and with smaller but "cheaper" shared memory. When all sub-HPRAMs finish their work they are merged back. Each sub-HPRAM consists of consecutive processors in the original H-PRAM and the shared memory is partitioned proportionally to the sizes of sub-HPRAMs (that is, if the original H-PRAM with processors $1, 2, \cdots, p$ has m memory cells $1, 2, \cdots, m$ then, after the `partition` instruction, the ith sub-HPRAM consisting of k_i processors $p_i+1, \cdots, p_i+k_i$ will have a contiguous block of $k_i \frac{m}{p}$ memory cells $p_i \frac{m}{p}+1, \cdots, (p_i+k_i)\frac{m}{p}$ of the original shared memory, accessible now to the ith sub-HPRAM processors only). Each of the new sub-HPRAMs can use the `partition` instruction again which gives rise to a hierarchy of sub-HPRAMs.

Let T denote the number of computational steps and C the number of shared memory accesses in an algorithm without `partition` instructions. Then the runtime of the algorithm is $T + C \cdot l(p)$. The cost of a `partition` instruction is the maximum time taken by the created sub-HPRAMs.

In order to fully support the H-PRAM, an "architecture" (network) must be capable of recursively partitioning itself into independent subsystems with the same structure as the whole network. The H-PRAM allows a lot of partitioning flexibility, in contrast to other models that restrict the size and number of submachines. In order to show that a network G can fully support the H-PRAM, a constant c and an ordering of the network processors must be provided such that for any two processors whose indices differ by d in the ordering, their distance is at most $c \cdot diam(d)$ in the network G, where $diam(d)$ is a diameter of a network of size d and the same structure as G. Two examples of networks with this property are the mesh and the hypercube. It was shown [4] that for the mesh, the Peano indexing scheme has the required property. The Gray code numbering of the nodes does the job for the hypercube. Figure 1 shows the numbering for a 16-processor mesh and a hypercube.

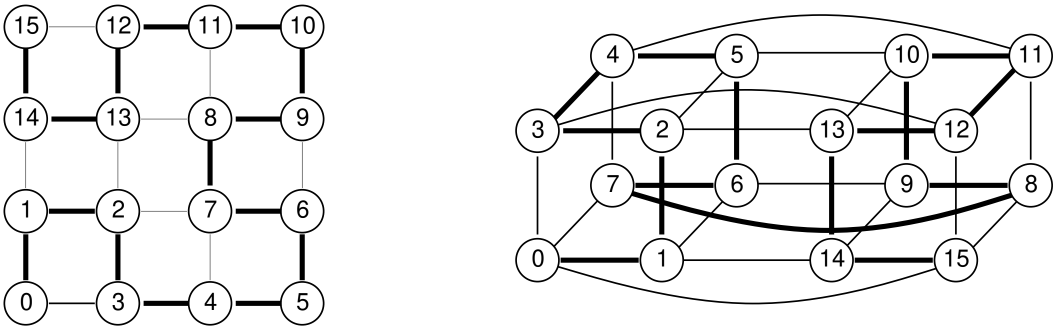

Figure 1: H-PRAM numbering for a mesh and a hypercube.

2.2 Other Models

Besides the H-PRAM, a few other general models were introduced that account for communication and try to exploit locality. In all these models the machine is somehow decomposable into submachines, and the communication cost is defined by one or two functions in the submachine size.

The D-BSP model [17], a refinement of the BSP model [18], is a distributed memory model that consists of a set of processors with local memory, a router for delivering messages in a point-to-point fashion, and facilities for global synchronization. Moreover, it has the ability to partition itself into submachines within which the communication is less expensive. The recursive D-BSP model can be partitioned at each level into 2 equally sized submachines. The machine operates in supersteps consisting of local computations, send and receive operations within the current submachine, and partition instructions. The communication operations are completed only at the end of the superstep and therefore the transmitted data can be used only in the next superstep. The cost of computation on the D-BSP model depends on a bandwidth-inefficiency function $g(k)$ and a latency function $l(k)$. The cost of a superstep without partitioning and with at most T local computation steps and at most h communication steps per processor, is $T + \max\{h \cdot g(k), l(k)\}$ where k is the size of the current submachine. The cost of the partition is the maximal cost of the created submachines. The total cost is the sum of the costs of the individual supersteps. A realization of this BSP model can be found in the Paderborn University BSP (PUB) library [3], which allows partitioning as an additional feature.

The Y-PRAM [16] is, like the H-PRAM, a shared memory model. The p-processor machine is recursively partitionable into two sub-PRAMs of size $p/2$. There are two parameters of the machine: a latency $l(k)$ and a bandwidth inefficiency $\beta(k)$ (not exactly matching the bandwidth inefficiency $g(k)$ of the D-BSP). Each processor owns an equal amount of the "shared" memory and within a submachine, memory accesses are possible to memory cells owned by the involved processors only. Each memory cell can be accessed exclusively only (EREW) and the time for a total cost of M memory accesses in a submachine of size k is $l(k) + m + M\beta(k)/k$, where m is the maximum of memory accesses per processor.

The E-BSP is another extension of the BSP model [18]. It is a distributed memory model that takes unbalanced communication as well as locality into account. Communication in a superstep is considered as an (M, h_1, h_2)-relation with locality L, where M is the total amount of communicated data, h_1 is the maximum of data sent and h_2 is the maximum of data received per processor. Locality L means that the maximum distance of two processors P_i and P_j communicating with each other is $|j - i| \leq L$. For each topology the cost of an (M, h_1, h_2)-relation has to be determined.

In the Fat-Tree model [2] the p processors are connected by a complete binary tree which serves as a router. Each routing module v in the tree has a capacity $w(k)$ that depends on the number of processors k in the subtree with root v. In each step a processor may send and receive a message and execute one local operation. A message arrives $2 \cdot h$ steps after it has been sent, where h is the height of the lowest common ancestor of the communicating processors. Only those algorithms are allowed that do not exceed the capacity of the routing modules.

3 The Upper Bound

In this section a simple H-PRAM algorithm for broadcast is given. This algorithm is very similar to the algorithm of Heywood and Ranka [8]. First, our algorithm ROOT is executed recursively on a k-processor sub-PRAM. After that, the k processors write the broadcast item to k equally interspaced memory locations. Finally, the algorithm is executed recursively in parallel by k sub-PRAMs with p/k processors, each. The difference between the algorithm of Heywood and Ranka and ours is that in the presented version the parameter k is varied. This leads to a better performance. For simplicity of the formal description of our algorithm ROOT, we assume that there are only p shared memory cells and the broadcast item is in memory cell 1, initially.

ROOT(p)

if $p = 2$ **then**

- Processor 2 reads the broadcast item from memory location 1,

else /* i.e. $p > 2$ */

- Partition the processors into two groups of sizes $\lceil\sqrt{p}\rceil$ and $p - \lceil\sqrt{p}\rceil$, and execute ROOT($\lceil\sqrt{p}\rceil$) on the first group,
- The processors $P_i, i = 1, \cdots, \lceil\sqrt{p}\rceil - 1$ write the broadcast item to memory locations $i \cdot \lceil\sqrt{p}\rceil + 1$,
- Partition the processors into $\lceil\sqrt{p}\rceil$ groups of sizes $\lceil\sqrt{p}\rceil$ each, except for the last group of size $p - (\lceil\sqrt{p}\rceil - 1) \cdot \lceil\sqrt{p}\rceil$ (it contains all remaining processors) and execute the algorithm ROOT recursively on each but the first group.

Theorem 3.1 *A broadcast can be performed on the H-PRAM in runtime*

$$T(p) = \sum_{i=0}^{\log\log p} 2^i \cdot l(p^{1/2^i}),\ \textit{for}\ p = 2^{2^k}\ \textit{for some integer}\ k,$$

and in runtime

$$T(p) \leq l(p) + 2 \cdot \sum_{i=0}^{\lceil \log\log p \rceil} 2^i \cdot l(\lceil p^{1/2^i} \rceil)\ \textit{otherwise}\ .$$

Proof. The broadcast algorithm consists of two recursive broadcast calls on sub-PRAMs of size $\lceil \sqrt{p} \rceil$ and one memory access. Therefore, the runtime of the algorithm is given by the recurrence: $T(p) = 2 \cdot T(\lceil \sqrt{p} \rceil) + l(p)$ for $p > 2$ and $T(1) = l(1)$.

In case of $p = 2^{2^k}$ all ceilings used in the algorithm are superfluous, and we get the recurrence $T(p) = 2 \cdot T(\sqrt{p}) + l(p)$, which is easily solved to

$$T(p) = \sum_{i=0}^{\log\log p} 2^i \cdot l(p^{1/2^i}). \tag{1}$$

For other p let $q > p$ be the minimal number such that $q = 2^{2^k}$ for some integer k. Then, since $q \leq p^2$, $\lceil \sqrt{p} \rceil \leq \sqrt{q}$, and l is non-decreasing, we may conclude from equation 3.1. :

$$T(\lceil \sqrt{p} \rceil) \leq T(\sqrt{q}) \leq \sum_{i=0}^{\log\log \sqrt{q}} 2^i \cdot l(q^{1/2^{i+1}}) \leq \sum_{i=0}^{\lceil \log\log p \rceil} 2^i \cdot l(\lceil p^{1/2^i} \rceil)$$

Plugging in this bound on $T(\lceil \sqrt{p} \rceil)$ in the above recursion for $T(p)$ yields the second time bound stated in Theorem 3.2. ∎

For ease of description, we will suppress the effects of ceilings in the rest of the paper. Variants of the algorithm ROOT can be designed for the other locality models, too, yielding analogous time bounds. For the D-BSP model, we get the following result:

Theorem 3.2 *A broadcast can be performed on the recursive D-BSP model in runtime* $T(p) = O(\sum_{i=0}^{\log\log p} 2^i \cdot \max\{l(p^{1/2^i}), g(p^{1/2^i})\})$.

Proof. A minor problem of the recursive D-BSP model is that not any consecutive subset of processors can be used as a submachine but the only allowed partitioning is into two submachines of equal size. However, this can be easily overcome by replacing a partitioning instruction into groups of size $\sqrt{p}$ by a sequence of $\frac{\log p}{2}$ partitioning instructions into halves. The non-recursive communication costs on a submachine of size p are $\max\{l(p), g(p)\}$. This leads to a runtime recurrence $T(p) \leq 2 \cdot T(\sqrt{p}) + \max\{l(p), g(p)\}$ for $p > 2$, and $T(1) = \max\{l(1), g(1)\}$. Solving this recurrence gives the desired runtime $T(p) = O(\sum_{i=0}^{\log\log p} 2^i \cdot \max\{l(p^{1/2^i}), g(p^{1/2^i})\})$. ∎

4 The H-PRAM Lower Bound

For the purpose of the lower bound proof we introduce the following simple computation model. It consists of p processors, each of them with a memory of size M. In each step each processor i can initiate one of the following operations

1. write the value x from its memory cell 0 to another of its local memory cells,
2. read the value x from any of its local cells and store it in its memory cell 0,
3. write the value x from its cell 0 to any cell of any other processor j,
4. read the value x from any cell of any other processor j and store it in its cell 0.

Each memory cell can be accessed by at most one processor at a time. Operations 1 and 2 need one time step, operations 3 and 4 need $l(|i-j|)$ time steps, that is the read or the written value x reaches its destination in $l(|i-j|)$ steps. The processors are allowed to start a new operation in each step, that is pipelining is allowed. Algorithms for this model are called *generic* algorithms.

It is easily seen that a "general" read or write operation (that is, for example, processor i reads the value x from any cell of processor j and stores it in any of its local cells) can be simulated by two of the above instructions. Thus, also any broadcast algorithm for the H-PRAM and other locality models can be transformed into a generic algorithm by increasing the runtime by a constant factor only. The reason for not allowing the "general" read and write operations on this simple model is a technical one and will be explain later on, after the introduction of the broadcast tree.

Further, the generic algorithms can be even made *oblivious*, i.e., they can be changed in such a way that the communication pattern is the same for all values of the broadcast item. To see this consider any generic broadcast algorithm and its computation on two different broadcast values, say on 0 and 1. We say that a memory cell m of processor i is *affected* by the broadcast value either if $i=1$ and $m=0$, or if processor i stores into its memory cell m value from an already affected memory cell. At the end of a single computation not all the processors need to have an affected memory cell, e.g. processor i may conclude from getting no message that the broadcast item is 1. However, to be able to distinguish between 0 and 1, each processor has to have at least one memory cell affected at the end of computation for at least one of these two items. We get an oblivious protocol from the arbitrary algorithm by merging the communication schemes for broadcast items 0 and 1. That is, we run first the algorithm on broadcast item 0 and we use each send operation from an affected memory cell to spread the real broadcast item x, and then we use the algorithm on broadcast item 1 in the same way. The runtime is at most twice as large.

Thus, we may deal with oblivious generic algorithms only, for the lower bounds. Such algorithms can be described by a static structure called *broadcast tree.* This is a binary tree with nodes labeled with a (processor, memory-cell) pair (i,k). The root is labeled (1,0) denoting the fact that the broadcast item is initially stored in cell 0 of processor 1. An edge between nodes labeled (i,k) and (j,k') is weighted with $l(|j-i|)$, if $j \neq i$, and is weighted with 1 otherwise. The left child of a node v always has the same label as v, the right child of a node labeled (i,k) has a label (j,k'), with $k'=0$ or $k=0$. In the case $k=0$ the edge represents 'processor i copies its local cell 0 to cell k' of processor j', in the other case the edge represents 'processor j copies the cell k of processor i into its cell 0.' For each i, all nodes with the label $(i, 0)$ must form a path. This condition reflects the fact that each processor can initiate only one operation per time step. An example of a broadcast tree is given in Figure 2.

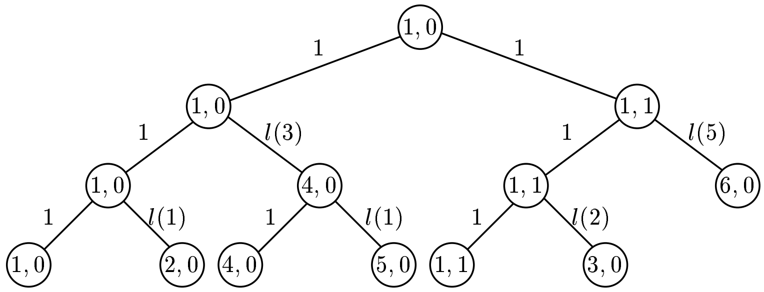

Figure 2: A broadcast tree of size 6, cost $\max\{1 + l(5), 1 + l(1) + l(3)\}$, and height 3.

The cost of a path is the sum of its edge-weights, the *cost* of the tree is its maximum root-leaf path cost. The *size* $\|T\|$ of the tree T is the number of different first components of the labels. It is easy to see that each generic algorithm for broadcast on p processors can be represented as a broadcast tree of size p in such a way that the cost of the tree corresponds to the runtime of the algorithm. Thus, in order to prove lower bounds for broadcast on the locality models, it suffices to prove lower bounds for the cost of broadcast trees of a given size. In the following the *height* of the tree will denote the length (in number of edges) of the longest root-leaf path.

Note that without the restriction about communication between processors of the simple model through memory cells 0 only, and without corresponding conditions about $(i, 0)$ nodes of the broadcast tree, this approach would not yield any nontrivial lower bound. Consider the following tree which would be a legal one in that case: root $(1, 0)$, the right son of a node with label $(1, i)$ on level $l, 0 \leq l < \log p$ has label $(1, i + 2^{\log p - 1 - l})$, the right son of a node $(1, i)$ on level $\log p$ has label $(i, 0)$. The size of this tree is p and its cost is $\log p + l(p)$ only.

Another note is appropriate at this point. The reason for having more nodes in the broadcast tree corresponding to a single processor is the shared memory of the H-PRAM: it makes possible that the broadcast item can spread from a single processor (given there are many copies of it in the local memory) to many other processors in parallel. The above definition of the broadcast tree captures this ability. In the case of parallel locality models without shared memory the definition of the broadcast tree and thus the whole lower bound proof can be slightly simplified (just one component in the labels - the processor number - and corresponding furhter simplifications).

We conclude the introduction of the broadcast tree by an observation concerning the relation of size, height and number of leaves in such a tree.

Lemma 4.1 *The number of leaves b of a broadcast tree with size s and height h is at most $b \leq 2 \cdot s \cdot h$.*

Proof. In parallel models, the number of memory cells that have been used by an algorithm, can be bounded by a product of the runtime and the number of processors. This can be reformulated for the broadcast tree, too. For each branching in the tree, one of the nodes with $(i, 0)$ labels is responsible. Moreover, all but the first and the last nodes on any of the path with the same $(i,0)$ label are responsible for at most one branching, the first node for at most two and the last one for none. Since the number of leaves is by one larger than the number of branchings, the number of leaves is at most $b \leq s \cdot h + 1 \leq 2 \cdot s \cdot h$. ∎

4.1 The Lower Bound Proof

Main Lemma *Let $C(p)$ denote the cost of the cheapest broadcast tree with a cost function $l(k), l(k) \leq \log^2 k$, and size p. Then*

$$C(p) = \Omega\left(\sum_{i=0}^{\log \frac{\log p}{\log\log p}} 2^i \cdot l(p^{1/2^i})\right) \tag{2}$$

Proof. Let S be a cheapest broadcast tree of size p. The idea of the proof is to find, in a recursive manner, an expensive path in the broadcast tree S. First, it is shown that S has an 'expensive edge', i.e. an edge with weight at least $\sqrt{p}$ (Lemma 4.2). Removing the first expensive edge from each root-leaf path that contains one, divides the tree S into a head H above these edges and a forest F below these edges. The forest F is proven to contain 'sufficiently many sufficiently large subtrees' (Lemma 4.4), and the subtree above these 'large subtrees' is shown to be expensive enough. Combining these facts yields, after some technical steps, the desired lower bound.

We start with a simple upper bound. The cost of a complete binary tree with p nodes, when used as a broadcast tree, is at most $l(p) \cdot \log p$. This implies $C(p) \leq l(p) \cdot \log p$. Since the cost of a tree is also an upper bound on its height, we have immediately an upper bound on the longest root-leaf path in S (*height bound*).

Lemma 4.2 *A tree S of size m and cost at most $l(m) \cdot \log m$, contains at least one edge e with cost $l(e) \geq l(m^{1/2})$.*

Proof. Let (i,k) denote the root of S. Since the size of S is m, there exists a node (v,n), for some n, such that $|v - i| \geq m/2$. If S consists of edges cheaper than $l(m^{1/2})$ only, then for each node $(j,l) \in S$ we have, with the help of the height bound, $|j - i| \leq \mathit{height}(S) \cdot m^{1/2}$ $\leq l(m) \cdot \log m \cdot m^{1/2} < m/2$, a contradiction. ∎

The edges with cost larger than $l(m^{1/2})$ will be called *expensive* edges in the rest of the section.

Consider all root-leaf paths in S and from each of them remove the first expensive edge, if there is any on it. At least one edge from S is removed, according to the Lemma 4.2. What remains is the upper tree *head* H, that does not contain any expensive edge, and the lower trees *forest* F, that may contain such edges (Figure 3).

Lemma 4.3 *The size of the subtree* head H *is bounded by* $\|H\| \leq m/2$.

Proof. If the size of H would be larger than $m/2$ then, by the same argument as in the proof of Lemma 4.2, there must be an expensive edge e in H. But H was chosen in such a way that there is no such edge. ∎

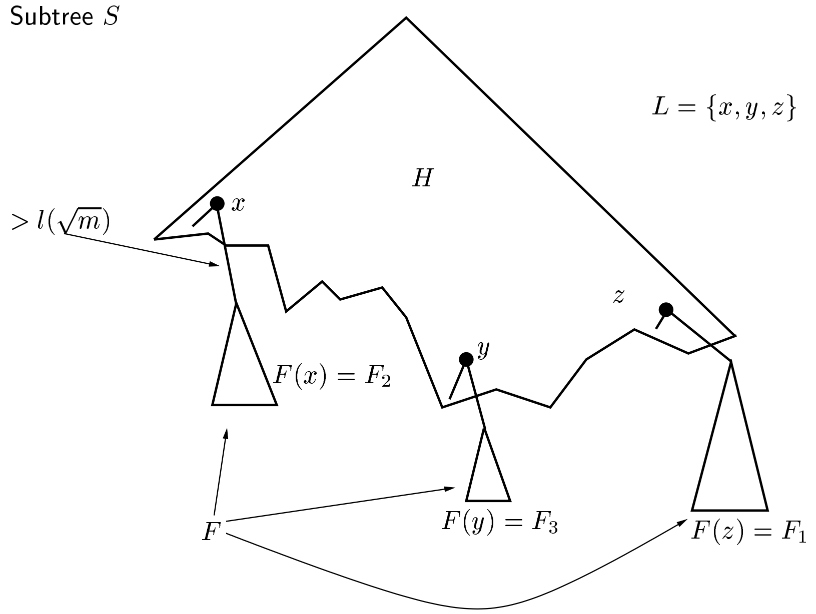

Figure 3: Subtree S and its partitioning into head H and forest F.

Let $L \subseteq H$ be the set of nodes of H that are incident to expensive edges (Figure 3). Using the fact that the size of L is at most $\|L\| \leq \|H\| \leq m/2$ and combining Lemma 4.1 with the height bound, we get a bound on the total number of nodes in L, namely $|L| \leq m/2 \cdot l(m) \cdot \log m$ (recall that the size $\|L\|$ is the number of nodes with different first component in the label only). Since the edge to the left child of every node in L has cost 1, each node in L has exactly one expensive edge and $|L|$ is also a bound on the number of trees in F. For $v \in L$, let $F(v)$ be the subtree of S that is connected to v by its expensive edge. Let $\|F(v)\|$ denote the size of $F(v)$. Recall that the size of a broadcast subtree is the number of its nodes with different first coordinate in the label.

Let $F_1, F_2, ..., F_{|L|}$ be the subtrees $F(v)$ for all $v \in L$, sorted by their size, starting from the largest. That is $\|F_i\| \geq \|F_j\|$ for $i < j$.

Lemma 4.4 *There exists a number $b \in \{1, \ldots, m\}$ such that for all $i \leq b$:*

$$\|F_i\| \geq \frac{m}{2 \cdot b \cdot \log m}$$

Proof. The size of all F_i's is at least:

$$\sum_{i=1}^{|L|} \|F_i\| = \|S\| - \|H\| \geq m/2 \tag{3}$$

Assume that Lemma 4.4 does not hold. Since $\|F_1\| \geq \|F_2\| \geq \ldots \geq \|F_{|L|}\|$, we have

$$\begin{aligned}
\|F_1\| &< \frac{m}{2 \cdot \log m} \text{ and} \\
\|F_2\| &< \frac{m}{2 \cdot 2 \cdot \log m} \text{ and } \ldots \\
\|F_i\| &< \frac{m}{2 \cdot i \cdot \log m} \text{ for all } i \leq |L|
\end{aligned}$$

This gives an upper bound on the sum of the sizes

$$\sum_{i=1}^{|L|} \|F_i\| < \sum_{i=1}^{|L|} \frac{m}{2 \cdot i \cdot \log m} = \frac{m}{2 \cdot \log m} \sum_{i=1}^{|L|} \frac{1}{i} < \frac{m}{2}$$

which is a contradiction to (3). ■

Lemma 4.5 *The minimum cost $C(m)$ of a tree of size m is*

$$C(m) \geq l(m^{1/2}) + 2 \cdot C\left(\frac{\sqrt{m}}{\sqrt{10} \cdot l(m) \cdot \log m}\right) .$$

Proof. We start with a proof of a slightly different recurrence: There exists a number $b \in \{1, \ldots, m\}$ such that

$$C(m) \geq l(m^{1/2}) + C\left(\frac{b}{2 \cdot \log b \cdot l(b)}\right) + C\left(\frac{m}{2 \cdot b \cdot \log m}\right) .$$

Consider the partitioning of the tree along the expensive edges, as described earlier. According to Lemma 4.4, there are b leaves in the head H such that each of them is connected by an expensive edge to a subtree with cost at least $C(\frac{m}{2 \cdot b \cdot \log m})$. Let H' denote the subtree of H that consists of these b leaves and all their predecessors. According to the height bound, H' has a height less than $l(b) \cdot \log b$. Lemma 4.1 implies that the size of H' is at least $\frac{b}{2 \cdot \log b \cdot l(b)}$ which gives a lower bound $C(\frac{b}{2 \cdot \log b \cdot l(b)})$ on the cost of H'.

To complete the proof we make use of a property of the cost function $C(m)$, stated in the next lemma:

Lemma 4.6

$$C(a) \geq 2 \cdot C\left(\sqrt{\frac{a}{5 \cdot l(a) \cdot \log a}}\right) .$$

Proof. Let E be the optimal broadcast tree of size a and let $a' = \sqrt{a/(5 \cdot l(a) \cdot \log a)}$. We want to show that E can be divided into a head V_{up} and a forest of subtrees V_{down} of this head such that both the cost of V_{up} and the cost of the maximal subtree in V_{down}, are at least $C(a')$.

For a node $(i, k) \in E$, let $T(i, k)$ denote the subtree of (i, k) and let left(i, k) and right(i, k) denote the subtrees of the children of (i, k). Let

$$V = \left\{(i, k) \mid \|T(i, k)\| \geq a' \text{ and } \|\text{left}(i, k))\| < a' \text{ and } \|\text{right}(i, k)\| < a'\right\} .$$

Let V_{up} denote the subtree of E that consists of the nodes in V and all their predecessors, and let $|V_{up}|$ denote the number of all nodes in V_{up}. Let V_{down} be the nodes in V and their subtrees. The size of V_{down} is at most $\|V_{down}\| \leq 2 \cdot |V| \cdot a'$. Let $\bar{V}$ be the set of nodes of E that are neither in V_{down} nor in V_{up}. To each node in V_{up} can be connected at most one subtree consisting of nodes from $\bar{V}$, and each such subtree has size at most a' (otherwise this subtree would be a subtree of V). We can bound the size of E by $\|E\| = a \leq \|V_{up}\| + \|\bar{V}\| + \|V_{down}\|$ $\leq \|V_{up}\| + 2 \cdot |V_{up}| \cdot a' + 2 \cdot |V_{up}| \cdot a' \leq \|V_{up}\| \cdot (1 + 4 \cdot l(a) \cdot \log a \cdot a' \leq \|V_{up}\| \cdot (5 \cdot l(a) \cdot \log a \cdot a')$, which implies true if $\|V_{up}\| \geq a'$. By the definition of V, any subtree attached to a node of V_{up} has size at least a', too. ■

Realizing that $C(x)+C(y) \geq C(x+y)$, we are ready to finish the proof of Lemma 4.5:

$$
\begin{aligned}
&C\left(\frac{b}{2\cdot\log b\cdot l(b)}\right)+C\left(\frac{m}{2\cdot b\cdot\log m}\right) \geq \\
&2\cdot C\left(\sqrt{\frac{b}{10\cdot\log^2 m\cdot l^2(m)}}\right)+2\cdot C\left(\sqrt{\frac{m}{10\cdot b\cdot\log^2 m\cdot l^2(m)}}\right) \geq \\
&2\cdot C\left(\frac{\sqrt{b}+\sqrt{\frac{m}{b}}}{\sqrt{10}\cdot l(m)\cdot\log m}\right) \geq 2\cdot C\left(\frac{\sqrt{m}}{\sqrt{10}\cdot l(m)\cdot\log m}\right).
\end{aligned}
$$

■

It remains to transform the recurrence of Lemma 4.5 into the desired sum. Consider a sequence $a_0 = p$, $a_{i+1} = \frac{\sqrt{a_i}}{\sqrt{10}\cdot\log^3 a_i}$. By a simple induction argument, it can be bounded from below by $a_i \geq \frac{p^{1/2^i}}{10\cdot log^6 p} \geq p^{\frac{1}{2^{i+1}}}$, for $i \leq \log\frac{\log p}{7\cdot\log\log p}$. This yields a lower bound

$$
C(p) \geq \sum_{i=0}^{\log\frac{\log p}{7\cdot\log\log p}} 2^i\cdot l(p^{1/2^{i+1}})
$$

and completes the proof of the Main Lemma. ■

The sum in equation (2) of the lower bound is almost exactly the sum in the equation (1) of the upper bound, except for the missing last $\Theta(\log\log\log p)$ elements in the summation. However, for latency functions $l(k) \geq \log k$, the sizes of the elements in the desired sum are non-increasing. That is, the missing elements do not contribute significantly to the sum and the upper and lower bounds match asymptotically each other.

For latency functions $l(k)$ with $\log k \geq l(k) \geq \log k/\log\log k$, the lower bound above sums up to $\Omega(\log p\cdot\log\log\log p)$. For these latency functions, the contribution of the missing $\Theta(\log\log\log p)$ elements is at most $O(\log p\cdot\log\log\log p)$. Therefore, the upper and the lower bound match each other for all latency functions $l(k) \geq \log k/\log\log k$.

For latency functions $l(k) \leq \log k/\log^{1+\epsilon}\log(k)$ with arbitrary $\epsilon > 0$, the upper bound is $O(\log p)$ which matches the trivial lower bound.

The Main Lemma and the remarks at the beginning of this section yield the following theorem.

Theorem 4.7 *The broadcast problem on the p-processor H-PRAM with a latency function* $l(k)$, $\frac{\log k}{\log\log k} \leq l(k) \leq \log^2 k$ *for all* $k \in \{1,\ldots,p\}$, *requires runtime*

$$
T(p) = \Omega\left(\sum_{i=0}^{\log\log p} 2^i\cdot l(p^{1/2^i})\right)
$$

By setting the costs of edges in the broadcast tree in a way that reflects the cost definition in the other parallel locality models, the lower bound can be applied to them as well.

Theorem 4.8 *The broadcast problem on the p-processor recursive D-BSP with a latency function $l(k)$ and bandwidth inefficiency function $g(k)$, $\frac{\log k}{\log\log k} \leq l(k) \leq \log^2 k$ and $\frac{\log k}{\log\log k} \leq g(k) \leq \log^2 k$ for all $k \in \{1, \ldots, p\}$, requires runtime*

$$T(p) = \Omega\left(\sum_{i=0}^{\log\log p} 2^i \cdot \max\{l(p^{1/2^i}), g(p^{1/2^i})\}\right)$$

5 Conclusion

In this paper the broadcast problem on general purpose parallel computation models that exploit locality is considered. The focus of the paper is on the proof of the optimality of our algorithm ROOT, that is on proving a lower bound that matches the runtime of this algorithm. Most of the results are presented for the H-PRAM model only but they can be easily modified for the other general locality models too. Moreover, it is straightforward that both the upper and lower bounds apply to prefix operations as well.

It was already know earlier that the models that do exploit locality are stronger than those that do not (e.g., consider the lower bound $\Omega(\log p \cdot \sqrt{p})$ for the broadcast on the BSP model with two-dimensional mesh parameters and the upper bound $O(\sqrt{p})$ for it on the corresponding H-PRAM). One of the implications of the presented lower bound is that, nevertheless, even the general models exploiting locality are weaker than the specific network models (consider the $\Omega(\log p \cdot \log\log p)$ broadcast lower bound for latency function l and the runtime $\log n$ of the simple broadcast algorithm on the hypercube). On the other hand, algorithms for our locality models are portable, i.e., our models are still "bridging models" as BSP or LogP.

Let us now have a different look on the latency function and the numbering of the nodes. The numbering can be understood as an embedding of the given graph G onto a line. When dealing with embeddings, usually the objective is to minimize the edge dilation. In contrast to this, our objective is to minimize the dilation of *paths* from G with respect to a metric on the line defined by the latency function l. This is exactly the meaning of the condition $\forall u, v \in G$ $\text{dist}(u, v) \leq l(|\pi(u) - \pi(v)|)$. We have seen that when a *line* is chosen as the target topology, the model is not able to capture all the communication properties of, e.g., the hypercube. Would choosing a different topology yield a more accurate model? If so, would not be the cost for the accuracy be too high from the programmers' point of view, that is, would not be the resulting model be too difficult to use?

For the following problems upper bounds have been already given on parallel locality models: broadcast and prefix operations (the optimal algorithms in the present paper), FFT graph, matrix multiplication, sorting and list ranking. For graph problems that do not have linear sequential runtime, it seems difficult to develop efficient algorithms that exploit locality. Is it possible to prove a lower bound for some of these that would imply a gap between the power of the d-dimensional mesh and the H-PRAM with d-dimensional mesh derived parameters? The broadcast problem was not able to do this.

References

[1] M. Adler, P.B. Gibbons, Y. Matias, and V. Ramachandran. Modeling Parallel Bandwidth: Local vs. Global Restrictions. In *Annual ACM Symposium on Parallel Algorithms*

and Architecture, 1997.

[2] G. Bilardi, B. Codenotti, G. Del Corso, C. Pinotti, and G. Resta. Broadcast and Other Primitive Operations on Fat-Trees. In *Euro-Par'97, LNCS 1300*, August 1997.

[3] O. Bonorden, B. Juurlink, I. von Otte, and I. Rieping. The Paderborn University BSP (PUB) Library - Design, Implementation and Performance. In *International Parallel Processing Symposium, IEEE Computer Society Press*, 1999. Full version as Technical report tr-rsfb-98-063, 1998, University Paderborn.

[4] George Chochia, Murray Cole, and T. Heywood. Implementing the Hierarchical PRAM on the 2D Mesh: Analyses and Experiments. Technical Report ECS-CSG-10-95, Department of Computer Science, The University of Edingburgh, Scotland, 1995. Also in SPDP 1995.

[5] D. Culler, R. Karp, D. Patterson, A. Sahay, K. Schauser an d E. Santos, R. Subramonian, and T. von Eicken. LogP: Towards a Realistic Model of Parallel Computation. In *Proc. 4th Symp. on Principles and Practice of Parallel Programming*, pages 1–12. ACM SIGPLAN, May 1993.

[6] T. Heywood and S. Ranka. Sorting and List Ranking on the Hierarchical PRAM Model. Technical report, School of Computer and Information Science, Syracuse University, December 1991.

[7] T. Heywood and S. Ranka. A Practical Hierarchical Model of Parallel Computation, I. The Model. *Journal of Parallel and Distributed Computing*, 16:212–232, 1992.

[8] T. Heywood and S. Ranka. A Practical Hierarchical Model of Parallel Computation, II. Binary Tree and FFT Algorithms. *Journal of Parallel and Distributed Computing*, 16:233–249, 1992.

[9] J. Hromkovic, R. Klasing, B. Monien, and R. Peine. Dissemination of Information in Interconnection Networks (Broadcast and Gossiping). In F. Hsu and Ding-Zhu Du, editors, *Combinatorial Network Theory*, pages 125–212. Kluwer Academic Publishers, 1995.

[10] B.H.H. Juurlink, P. Kolman, F. Meyer auf der Heide, and I. Rieping. Optimal broadcast on parallel locality models. In M. Flammini, E. Nardelli, G. Proietti, and P. Spirakis, editors, *Proc. 7th Int. Coll. Structural Information and Communication Complexity, SIROCCO*, pages 211–225. Carleton Scientific, 2000.

[11] B.H.H. Juurlink and H.A.G. Wijshoff. The E-BSP Model: Incorporating Unbalanced Communication and General Locality into the BSP Model. In *Proc. of the Annual International EURO-PAR Conference, LNCS*, pages 339–347. Springer LNCS 1124, 1996.

[12] B.H.H. Juurlink and Harry A.G. Wijshoff. A Quantitative Comparison of Parallel Computation Models. In *Annual ACM Symposium on Parallel Algorithms and Architecture*, pages 13–24, 1996.

[13] F. T. Leighton. *Introduction to Parallel Algorithms and Architectures.* Morgan Kaufman, San Mateo, 1992.

[14] B.M. Maggs and E.J. Schwabe. Real-time Emulations of Bounded-degree Networks. *Information Processing Letters*, 66(5):269–276, 16 June 1998.

[15] A.L. Rosenberg, V. Scarano, and R.K. Sitaraman. The Reconfigurable Ring of Processors: Fine-Grain Tree-Structured Computations. *IEEE Transactions on Computers*, 46(10), Oct 1997.

[16] P. de la Torre and C.P. Kruskal. Towards a Single Model of Efficient Computation in Real Parallel Machines. In *Parallel Architectures and Languages Europe*, pages 6–24. LNCS, 1991.

[17] P. de la Torre and C.P. Kruskal. Submachine Locality in the Bulk Synchronous Setting. In *EUROPAR: Parallel Processing, 2nd International EURO-PAR Conference*. LNCS, 1996.

[18] Leslie Valiant. A Bridging Model for Parallel Computation. *Communications of the ACM*, 33(8), August 1990.