The Reconfigurable Ring of Processors: Fine-Grain Tree-Structured Computations*

Arnold L. Rosenberg Vittorio Scarano† Ramesh K. Sitaraman

Department of Computer Science
University of Massachusetts at Amherst
Amherst, MA 01003, USA
E-mail: {rsnbrg,vittorio,ramesh}@cs.umass.edu

*A portion of this paper was presented at the *6th IEEE Symp. on Parallel and Distributed Processing*, Dallas, Tex., October 26-29, 1994.

†Current affiliation: Dipartimento di Informatica ed Applicazioni "R.M. Capocelli", Universitá di Salerno, 84081 Baronissi (SA), Italy. E-mail: vitsca@udsab.dia.unisa.it.

Abstract

We study fine-grain computation on the *Reconfigurable Ring of Processors* ($\mathcal{RRP}$), a parallel architecture whose processing elements (PEs) are interconnected via a multiline reconfigurable bus, each of whose lines has one-packet width and can be configured, independently of other lines, to establish an arbitrary PE-to-PE connection. We present a message-passing protocol, COMET, that will, in the presence of suitable implementation technology, endow an $\mathcal{RRP}$ with message latency that is *logarithmic* in the number of PEs a message passes over in transit. Our study focusses on the computational consequences of such latency in such an architecture. Denoting by (N, L)-$\mathcal{RRP}$ an N-PE ring whose bus has L lines, we establish the following computational properties of logarithmic-latency $\mathcal{RRP}$s.

1. A *leveled tree-structured algorithm* (LTS algorithm) consists of some fixed number of complete up- and/or down-sweeps on a complete binary tree, performing a unit-time computation at each node; (one-to-all) broadcast and accumulation (reduction) are 1-sweep LTS algorithms; parallel-prefix (scan) is a 2-sweep LTS algorithm.
   (a) An (N, L)-$\mathcal{RRP}$ can execute each sweep of an N-argument LTS algorithm within time
   $$\frac{\log^2 N}{\log L} + \log N \log\log L. \tag{1}$$
   Hence, when L is as large as $N^{1/\log\log N}$, the $\mathcal{RRP}$ performs a sweep within time $2 \log N \log\log N + 1$.o.t.
   (b) The performance of $\mathcal{RRP}$s on LTS algorithms can be improved by at most a constant factor, both when the buswidth L is "small," so that the first term of bound (1) dominates, and when L is "large," so that the second term dominates. Our lower bound extends to a much broader range of architectures than $\mathcal{RRP}$s: all of these architectures require time proportional to $\log N \log\log N$ to perform any N-argument 1-sweep LTS algorithm, such as broadcast.
2. We expose an architecture-independent limitation imposed by the logarithmic communication latency of $\mathcal{RRP}$s: for a broad range of N-PE parallel architectures, including N-$\mathcal{RRP}$s, any operation that requires one PE to receive and/or send information—directly or indirectly—from and/or to all other PEs, requires time proportional to $\log N \log\log N$.

1 Introduction

1.1 Overview

We study fine-grain computation on the *Reconfigurable Ring of Processors* ($\mathcal{RRP}$), a parallel architecture whose processing elements (PEs) are interconnected via a multiline reconfigurable bus: each line of the bus has one-packet width and can be configured, independently of other lines, to establish an arbitrary PE-to-PE connection.

Our study is inspired by a novel strategy for (a) designing the bus of an $\mathcal{RRP}$ and for (b) passing one-word messages along the lines of the bus in a way that yields message latency that is *logarithmic* in the number of PEs the message traverses—at least for (MOS) wafer-scale implementations of $\mathcal{RRP}$s. We are currently working on estimating the technological parameters that would enable our COoperative MEssage Transmission (COMET) strategy to be realized. Our goal in the current paper, as in its companion paper [7], is to understand the computational consequences of a fine-grain logarithmic delay model for $\mathcal{RRP}$s: sufficiently good news would provide powerful motivation for paying the technological cost that would enable the model to be realized. Henceforth, we focus only on $\mathcal{RRP}$sthat have been implemented so as to achieve logarithmic message latency—perhaps, but not necessarily, via COMET. The interested reader should see [6] for another theoretical study of reconfigurable architectures with logarithmic message latency.

In the remainder of this section, we describe, in turn, our main results, the detailed architecture of $\mathcal{RRP}$s, and the COMET message-passing protocol. The subsequent sections of the paper describe our results and their proofs in detail.

1.2 Our Results

Our results are parameterized by the number of PEs in an $\mathcal{RRP}$ and the number of lines in its bus. We denote an N-PE ring whose bus has L lines an (N, L)-$\mathcal{RRP}$ when we want to make all parameters explicit, an N-$\mathcal{RRP}$ when we want to concentrate only on the number of PEs, and an $\mathcal{RRP}$ when the specific parameter values are not consequential.

In [7], we showed that $\mathcal{RRP}$s can efficiently execute any *normal hypercube algorithm*[1], a class which contains efficient algorithms for such diverse problems as sorting, matrix multiplication, and the Fast Fourier Transform. The upper-bound results of the current paper, which appear in Section 2, show that $\mathcal{RRP}$s can perform, with close to optimal efficiency, any *leveled tree-structured algorithm* (LTS algorithm), i.e., any algorithm that

[1] See, e.g., [4] for definitions.

can be specified in terms of a fixed number of complete up- and/or down-sweeps on complete binary trees, performing a unit-time task each time a tree-node is encountered. Problems that can be solved by such algorithms include (segmented versions of) broadcast and accumulation (a/k/a reduction), which are 1-sweep algorithms, and parallel-prefix (a/k/a scan), which is a 2-sweep algorithm. When executed on an (N, L)-$\mathcal{RRP}$, our algorithm performs each sweep in time bounded above by expression (1). Thus, for values of L smaller than $N^{1/\log\log N}$, each sweep of the algorithm runs within time

$$\mathsf{Time}_{\mathsf{Sweep}} \leq \frac{2\log^2 N}{\log L},$$

while for larger values of L, each sweep runs within time

$$\mathsf{Time}_{\mathsf{Sweep}} \leq 2\log N \log\log N.$$

(Since we envision only chip- or wafer-scale $\mathcal{RRP}$s, the values of N will always be moderate, no more than, say, a few hundred; hence, a bus with $N^{1/\log\log N}$ lines is quite within the realm of feasibility.) It follows that an N-$\mathcal{RRP}$, endowed with sufficient communication bandwidth (as measured by the number of buslines), can execute LTS algorithms almost as fast as can an N-node hypercube of commensurate-power PES: the slowdown incurred is only roughly $2\log\log N$.

In Section 3, we show that the algorithms of Section 2 are within constant factors of optimal, in a variety of senses. Throughout this section, we use (one-to-all) broadcast as the prototypical 1-sweep LTS algorithm, since it is—in a sense made precise in Section 4—the simplest such algorithm. In Section 3.1, we show that the time for performing a one-to-all broadcast on an (N, L)-$\mathcal{RRP}$ is bounded below by

$$\mathsf{T}_{\mathsf{Broadcast}} \geq \frac{1}{3}\frac{\log^2 N}{\log L}.$$

Since this bound trivializes when L is as large as N^{ϵ} (because even communicating between PEs 0 and $N/2$ takes time proportional to $\log N$), we were motivated to find a nontrivial lower bound that holds for *all* $\mathcal{RRP}$s, no matter how many buslines they have. In Section 3.2 we prove such a bound: the time required for *any* N-$\mathcal{RRP}$—no matter how many buslines it has—to perform a one-to-all broadcast is no smaller than

$$\mathsf{T}_{\mathsf{Broadcast}} \geq \frac{1}{15}\log N \log\log N.$$

This lower bound is surprisingly general: not only does it apply to all N-$\mathcal{RRP}$s, it applies also, with minor adaptation, to a much broader class of logarithmic-latency architectures. One can view this fact as suggesting that, at least for LTS algorithms, one could not gain

appreciably—i.e., by more than a constant factor—if one replaced the ring-structured topology of $\mathcal{RRP}$s by a more highly connected topology such as the mesh.

The extendibility of the lower bound in Section 3.2 both inspires and lays the technical groundwork for our very general lower bound in Section 4. We focus in that section on the following broad class of algorithms for *any parallel architecture that uses point-to-point communication*. We term an algorithm *nontrivial* if it requires some one PE of the architecture to receive and/or send information—either directly or indirectly—to and/or from all other PEs. We prove that, to within constant factors, no nontrivial algorithm can be performed more efficiently than the (one-to-all) broadcast operation. As a consequence, any N-PE architecture of the sort discussed in Section 3.2 requires time proportional to $\log N \log\log N$ to execute any nontrivial algorithm. We stress that this lower bound is a fundamental limitation imposed by the logarithmic-latency model and is independent of specific architectural implementations![2]

1.3 The Abstract $\mathcal{RRP}$ Architecture

A static view. An (N, L)-$\mathcal{RRP}$ is a SIMD architecture that comprises N identical PEs,[3] $\mathcal{P}_0, \mathcal{P}_1, \ldots, \mathcal{P}_{N-1}$, which we view as placed with equal spacing around a circle (see Figure 1). There is a "bundle" of L lines, each having one-packet-width, passing outside the circle "over" the PEs and "through" the switches that provide the reconfiguration capability. Each PE of an (N, L)-$\mathcal{RRP}$ has L associated *communication sub-PEs* (CPEs) to help it manage the message traffic on its bus; specifically, CPE $\mathcal{CP}_{i,j}$ controls switches on line j that allow PE $\mathcal{P}_i$ to participate (either as a source, a destination, or an intermediate node) in a dedicated point-to-point path along that line. We impose certain limitations on $\mathcal{RRP}$s, in order to minimize the technological resources required to implement them. First, as is implicit in the fact that buslines form *point-to-point* paths between pairs of communicating PEs, we assume that each message has exactly one sending PE and exactly one receiving PE; in particular, we do not support any "wired-or" or multireader bus capability. Second, we have our $\mathcal{RRP}$s observe a *single-port* communication regimen: in a single step a PE can send at most one message and receive at most one (possibly, though not necessarily, involving distinct buslines). A word about these restrictions is in order. The constraint of single-port communication may well decrease the efficiency of $\mathcal{RRP}$s on a broad-range of computational problems; we have yet to study the multiport version of our model. However, the absence of a multireader capability has less overall impact than one might initially expect. Clearly, a multireader bus capability would enable an N-$\mathcal{RRP}$ to perform one-to-all broadcasts in time proportional to $\log N$; however, this

[2] It is instructive to see how our model and results compare with those of [6].

[3] Throughout, when we discuss the PEs of an N-$\mathcal{RRP}$, all PE-indices are computed modulo N.

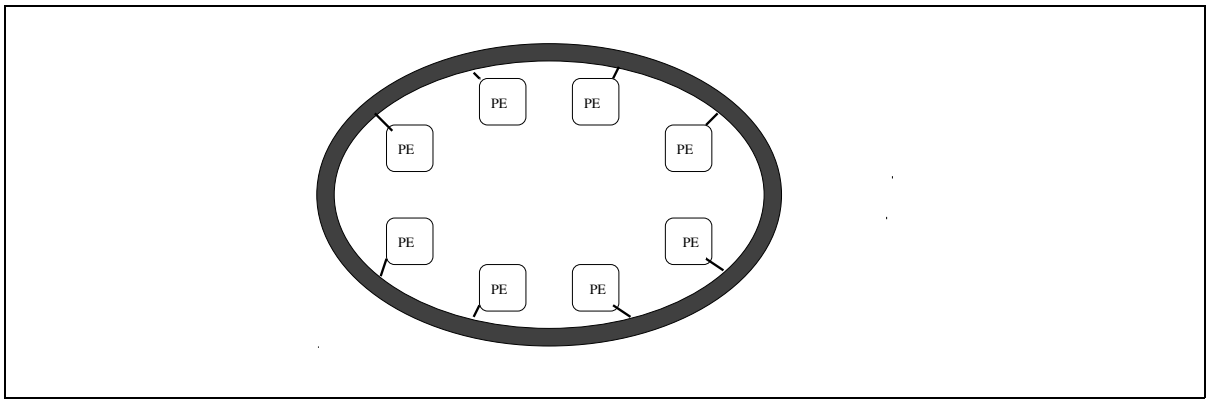


Figure 1: *An abstract view of an $\mathcal{RRP}$.*

speedup would likely incur nonnegligible costs, as the multiple voltage drains would require either larger drivers, hence, a smaller $\mathcal{RRP}$ for a fixed amount of real estate, or slower CPEs, hence, a larger constant of proportionality on the time-function. Moreover, building on the techniques of Sections 3 and 4, one can show that a multireader bus capability would *not* accelerate operations such as accumulation, hence would have only small impact on operations such as parallel-prefix.

The SIMD regimen observed by our $\mathcal{RRP}$s allows switch settings to be computed centrally and downloaded to the CPEs; hence, we shall not comment further on this aspect of $\mathcal{RRP}$ operation.

For reasons that will become clear when we describe the COMET message-passing protocol (in Section 1.4), we study only "fine-grain" computations by $\mathcal{RRP}$s i.e., ones in which every inter-PE communication consists of a single one-packet message which is sent along a single busline.

Finally, to simplify algorithm specification and analysis, we assume henceforth that our $\mathcal{RRP}$s have numbers of PEs N and numbers of buslines L that are powers of 2; in particular, we shall write (when convenient) $N = 2^n$ and $L = 2^\ell$. These assumptions, which can be avoided by clerical modifications to our development, will affect only small additive terms in our bounds.

> **Notation.** For any pair of PE-indices i and $j \neq i$, we denote by $B(i,j)$ the *block* of PEs $\{\mathcal{P}_i, \mathcal{P}_{i+1}, \ldots, \mathcal{P}_j\}$.

Path formation and communication. The first step in performing a communication between PEs $\mathcal{P}_i$ and $\mathcal{P}_j$ (in either direction) is to establish a dedicated path that connects

these two PEs. This is achieved as follows. The SIMD controller appropriates a segment of some busline which runs above either block $B(i,j)$ or block $B(j,i)$ and which is not currently used by any other communication. (Presumably, but not necessarily, if segments above both blocks are available, then the controller will choose the shorter one.) Say, for definiteness, that the available segment is a portion of busline k that runs above block $B(i,j)$. The controller then establishes the following connections, using the switches in the CPEs; see Figure 2.

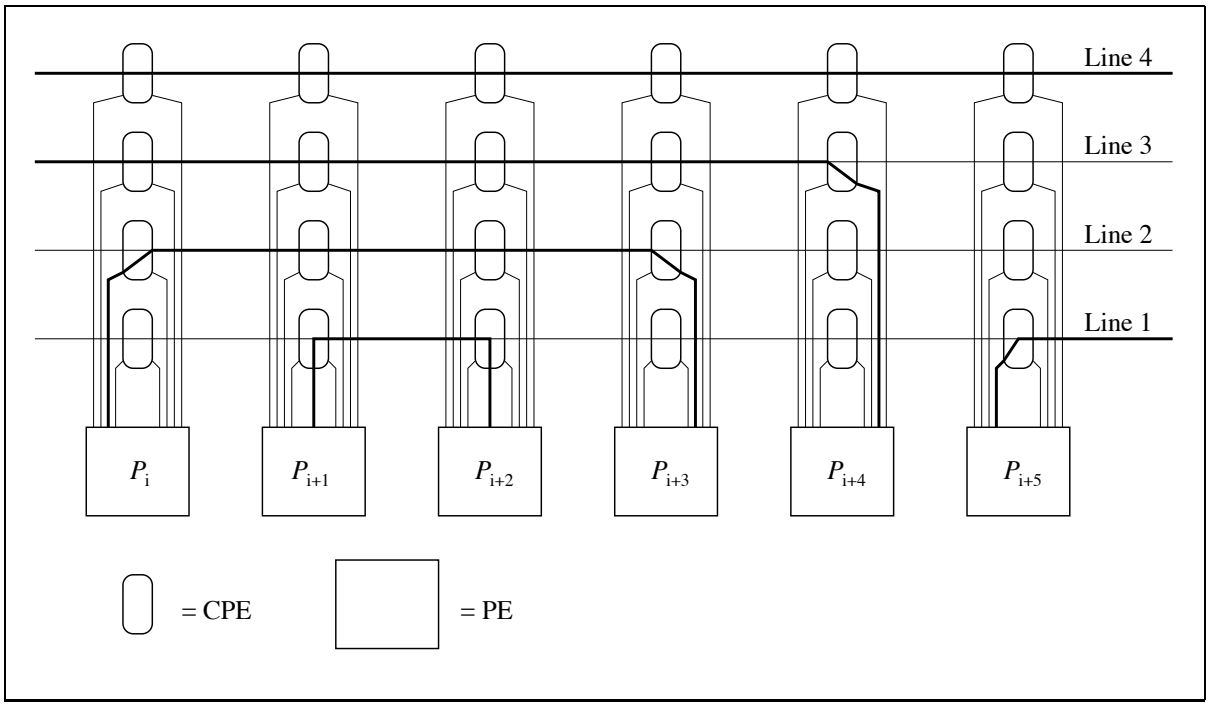


Figure 2: *A section of a bus of a* $(N,5)$*-*$\mathcal{RRP}$.

- PE $\mathcal{P}_i$ is connected to line k, via CPE $\mathcal{CP}_{i,k}$;
- PE $\mathcal{P}_j$ is connected to line k, via CPE $\mathcal{CP}_{j,k}$;
- each CPE $\mathcal{CP}_{h,k}$, where $i \leq h < j$ is connected to CPE $\mathcal{CP}_{h+1,k}$.

(If no busline is free above either block $B(i,j)$ or $B(j,i)$, then the message transmission must be deferred until a later time. The algorithms we present are carefully designed to avoid such an event.) After forming the dedicated path, the desired (one-packet) message is transmitted from $\mathcal{P}_i$ to $\mathcal{P}_j$ along line k.

More notation. For any block of PEs $B(i,j)$, a communication between PEs $\mathcal{P}_i$ and $\mathcal{P}_j$ that is transmitted above block $B(i,j)$—in either direction—is termed a $(i \leftrightarrow j)$ communication. Of course, every communication between $\mathcal{P}_i$ and $\mathcal{P}_j$ is either a $(i \leftrightarrow j)$ communication or a $(j \leftrightarrow i)$ communication.

Message latency. We assume that the latency (or, duration) of a communication depends only on the distance the message travels (not the direction) and is *logarithmic* in this distance. Specifically, we assume that every $(i \leftrightarrow j)$ communication takes[4]

$$\lceil \log((j-i) \bmod N) \rceil$$

steps. In the next subsection, we sketch a possible implementation of $\mathcal{RRP}$s, which has the promise of implementing this delay model.

1.4 An Implementation Proposal

Our aim in this study is not to propose yet another abstract model, but rather to explore the consequences of stretching existing VLSI technology in certain directions. Therefore, we view as an integral part of our research the following outline of a strategy for achieving logarithmic (fine-grain) message latency in $\mathcal{RRP}$s. This strategy has been a major motivating factor for our work; its technological feasibility and costs are currently under study.

Remark. The results in this paper, as those in [7], rely only on the abstract properties of $\mathcal{RRP}$s (such as logarithmic message latency) that are described in Section 1.3. Having said this, we would be very excited if the following design strategy could be developed to yield a "real" instantiation of the results.

The hardware-design portion of our strategy is specified only implicitly: we would like to design the CPEs of $\mathcal{RRP}$s so that they can support the COMET "cooperative" message transmission protocol, which (on paper, at least) allows the PEs of $\mathcal{RRP}$s to exchange one-word messages with only logarithmic communication latency.

The COMET protocol builds on the assumption that there is a fixed *transit time* τ such that a one-packet message can be transmitted in τ (machine) cycles between:

- any PE $\mathcal{P}_i$ and any one of its CPEs $\mathcal{CP}_{i,k}$ (in either direction);

[4] All logarithms are to the base 2 unless otherwise specified.

- any CPE $\mathcal{CP}_{i,k}$ and a neighboring CPE $\mathcal{CP}_{i\pm1,k}$.

As our description of COMET proceeds, the reader should note two defining properties of the protocol.

- COMET builds on specific characteristics of MOS VLSI technology, particularly its being a capacitive, voltage-driven technology, rather than a current-driven one). Therefore, COMET will accelerate message transmission only with chip- or wafer-scale implementations of $\mathcal{RRP}$s, not with implementations that leave an MOS environment. Consequently, we envisage the $\mathcal{RRP}$s we study as comprising at most a few hundred PEs. This worldview makes it imperative that we always seek explicit analyses of bounds, rather than asymptotic ones.
- COMET depends on having neighboring CPEs "cooperate" to accelerate the progress of a message in transit. This "cooperation" precludes pipelining messages, so we must restrict attention to routing small packets rather than, say, potentially long worms. This limitation explains our focussing here only on *fine-grain* computations and messages.

The Logic of Comet. We define the cooperative message transmission protocol that COMET uses to accelerate message transmission, via the following generic example. Let us focus on an arbitrary single-packet ($i \leftrightarrow j$) message M that PE $\mathcal{P}_i$ wishes to transmit to PE $\mathcal{P}_j$ on busline k. Assume that $\mathcal{P}_i$ has already inserted message M onto busline k via CPE $\mathcal{CP}_{i,k}$.

Step 0. $\mathcal{CP}_{i,k}$ sends message M to $\mathcal{CP}_{i+1,k}$. Pictorially:

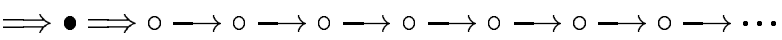
$$\bullet \Longrightarrow \circ \longrightarrow \circ \longrightarrow \circ \longrightarrow \circ \longrightarrow \circ \longrightarrow \circ \longrightarrow \circ \longrightarrow \circ \longrightarrow \cdots$$

In this illustration:

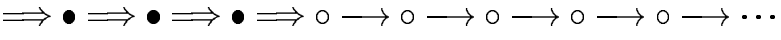
- a filled circle denotes a CPE that "knows" message M
- an empty circle denotes a CPE that does not "know" message M
- a single arrow denotes an "empty" link of the dedicated path $P(i,j)$
- a double arrow denotes a link of the dedicated path $P(i,j)$ that contains message M.

Thus, the above diagram is intended to illustrate that, after one step (of duration τ cycles), both CPEs $\mathcal{CP}_{i,k}$ and $\mathcal{CP}_{i+1,k}$ "know" message M.

Step 1. $\mathcal{CP}_{i+1,k}$ starts helping $\mathcal{CP}_{i,k}$ send message M along line k.

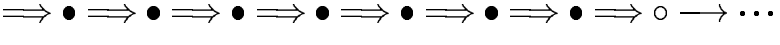


Step 2. $\mathcal{CP}_{i+2,k}$ and $\mathcal{CP}_{i+3,k}$ start helping $\mathcal{CP}_{i,k}$ and $\mathcal{CP}_{i+1,k}$ send message M along line k.



$\star\star\star$

Step m. $\mathcal{CP}_{i+2^{m-1},k}$, $\mathcal{CP}_{i+2^{m-1}+1,k}$, ..., $\mathcal{CP}_{i+2^m-1,k}$ start helping $\mathcal{CP}_{i,k}$, $\mathcal{CP}_{i+1,k}$, ..., $\mathcal{CP}_{i+2^{m-1}-1,k}$ send message M along line k.



The Technology and the timing. Each step (of duration τ cycles) of a COMET message transmission doubles both the number of CPEs transmitting message M and the number receiving the message (the latter number ignores a possible discrepancy in the very last step when $j-i$ mod N is not a power of 2). The feature of MOS technology that allows a transmission to speed up in this way (traveling twice as far at step t as at step $t-1$) is that 2^t CPEs can "pump" voltage twice as hard as can 2^{t-1} CPEs. In a voltage-driven technology, where delays are caused by capacitive loads, this harder "pumping" allows successively longer line segments to fill to threshold at successive steps of the transmission. This scheme leads to the logarithmic latency model described earlier.

Of course, the trick in making COMET work in a real technology is to build wires that will carry the pumped charge without melting. This limitation explains why we must model and simulate electrical overhead of the COMET protocol, in order to determine how large and fast an $\mathcal{RRP}$ the protocol will permit. Specifically, we wish to determine what clock speed (which is embodied in the transit time τ) can be supported with various numbers of PEs. As we stated in Section 1.1, our goal here is to determine whether or not logarithmic message latency leads to computational efficiency that would induce one to pursue vigorously a switch design that will efficiently support the COMET protocol. We believe that our upper bounds, here and in [7], supply an affirmative answer to this question.

> **Remark.** We stress that the COMET protocol is intended to diminish only the *capacitive* delays of message transmission in MOS technologies; it does not affect transmission-line limitations. Therefore, our speedup scheme does not run into any conflicts with speed-of-light limitations [1, 8].

2 LTS Algorithms for $\mathcal{RRP}$s

This section is devoted to our upper-bound results. We begin, in Section 2.1, with an algorithm that allows an (N, L)-$\mathcal{RRP}$ to perform a single sweep of an LTS Algorithm in time

$$T(N, L) \leq \frac{\log^2 N}{\log L} + \log N \log\log L.$$

We then describe briefly, in Section 2.2, how to perform the operations of broadcast and accumulation using 1-sweep LTS algorithms and how to compute the parallel-prefix using a 2-sweep LTS algorithm; our descriptions are brief because these LTS algorithms are well known.

2.1 Generic Single-Sweep LTS Algorithms

In this subsection, we describe two intimately related parameterized families of 1-LTS algorithms. Each algorithm $\mathcal{D}own_{(N,L)}$ (resp., $\mathcal{U}p_{(N,L)}$) simulates, on an (N, L)-$\mathcal{RRP}$, a single *downward* (resp., *upward*) sweep on an N-leaf complete binary tree. For the sake of simplicity, we describe only algorithms that involve the entire $\mathcal{RRP}$. However, because our algorithms actually work on a *path* of PEs, rather than on a ring, it should be clear that they are easily modified to compute *segmented* versions of the same operations.[5] (This fact is essential for the correctness of our recursive simulations.) In fact, we exploit the wraparound structure of our rings only in our allowing *any* PE to play the role of the root of the tree in a simulated sweep.

2.1.1 Specification of Algorithms $\mathcal{D}own_{(N,L)}$ and $\mathcal{U}p_{(N,L)}$

Fix the parameters N and L, and focus on an (N, L)-$\mathcal{RRP}$ $\mathcal{R}_{N,L}$. Both our downward-sweep algorithm $\mathcal{D}own_{(N,L)}$ and our upward-sweep algorithm $\mathcal{U}p_{(N,L)}$ have $\mathcal{R}_{N,L}$ simulate the N-leaf complete binary tree $\mathcal{T}_N$ via a recursive sequence of remappings of the nodes of $\mathcal{T}_N$ onto the PEs of $\mathcal{R}_{N,L}$. These mappings and remappings are specified for $\mathcal{D}own_{(N,L)}$ implicitly via the following recursive three-phase process.

Informal Specification of $\mathcal{D}own_{(N,L)}$

Phase 1: Sweeping down the top of $\mathcal{T}_N$. $\mathcal{D}own_{(N,L)}$ recursively uses the block of PEs $B(0, 2^k - 1)$ to execute the top $k + 1$ levels of $\mathcal{T}_N$. (We choose the parameter

[5] Our *lower bounds* hold for arbitrary algorithms on $\mathcal{RRP}$s, not just those that operate as ours do.

k later.) We assume that this execution ends with the 2^k level-k nodes of $\mathcal{T}_N$ distributed, in left-to-right order, in the PEs of block $B(0, 2^k - 1)$.

Phase 2: Remapping. $\mathcal{D}own_{(N,L)}$ remaps level-k of $\mathcal{T}_N$ by having each PE $\mathcal{P}_i$ of block $B(0, 2^k - 1)$ send its current "state" to PE $\mathcal{P}_{i2^{n-k}}$.
(We assume here that fine-grain LTS algorithms produce very small "states." This is true for the motivating examples in the literature.)

Phase 3: Sweeping down the bottom of $\mathcal{T}_N$. $\mathcal{D}own_{(N,L)}$ recursively uses all blocks of PES $B(i2^{n-k}, (i+1)2^{n-k} - 1)$, in parallel, to execute the bottom $n - k$ levels of $\mathcal{T}_N$. We assume that this execution ends with the leaves of $\mathcal{T}_N$ (which are its level-n nodes) distributed, in left-to-right order, in the PEs of $\mathcal{R}_{N,L}$.

Choosing the parameter k. We have two goals that jointly determine our choice of the parameter k.

1. In order to minimize the communication overhead of our tree-sweep, we wish to be able to perform the remapping (Phase 2) at each level of the recursion via a single global communication. To accomplish this, we choose k so that $2^k \leq L + 1$. We reason that we can remap up to L level-k nodes of $\mathcal{T}_N$ via a single global communication, since $\mathcal{R}_{N,L}$ has precisely this many buslines. Since the leftmost level-k node does not move in the remapping (staying in PE $\mathcal{P}_0$), we arrive at the indicated inequality; since L is a power of 2, our inequality mandates making $k \leq \ell$.

2. Letting $T(M; L)$ denote the time that $\mathcal{D}own_{(N,L)}$ takes to sweep down an M-leaf subtree of $\mathcal{T}_N$, we expect the recursive initiation (Phase 1) of the algorithm to take time $T(2^k; L)$ and the recursive extension (Phase 3) to take time $T(2^{n-k}; L)$. Heuristic arguments suggest that $T(M; L)$ is a convex function of M, whence the time for $\mathcal{D}own_{(N,L)}$ (which is dominated by the sum $T(2^k; L) + T(2^{n-k}; L)$) will be minimized if we balance the times for Phases 2 and 3 by choosing k so that $2^k \approx 2^{n-k}$.

These two arguments lead us to choose the parameter k to be as large as possible, subject to the restriction that $2^k \leq \min(L, \lceil\sqrt{N}\rceil)$.

We now present a detailed specification of Algorithm $\mathcal{D}own_{(N,L)}$. Each recursive invocation of the algorithm requires two parameters: the index `root.index` of the PE of $\mathcal{R}_{N,L}$ that plays the role of the root of the current tree at that point in the recursion, and the number `num.leaves` of leaves of that tree. With no loss of generality, PE $\mathcal{P}_0$ of $\mathcal{R}_N$ will play the role of the root of the initial tree $\mathcal{T}_N$; of course, the initial number of leaves is N. The detailed specification appears in Figure 3. We simplify the specification

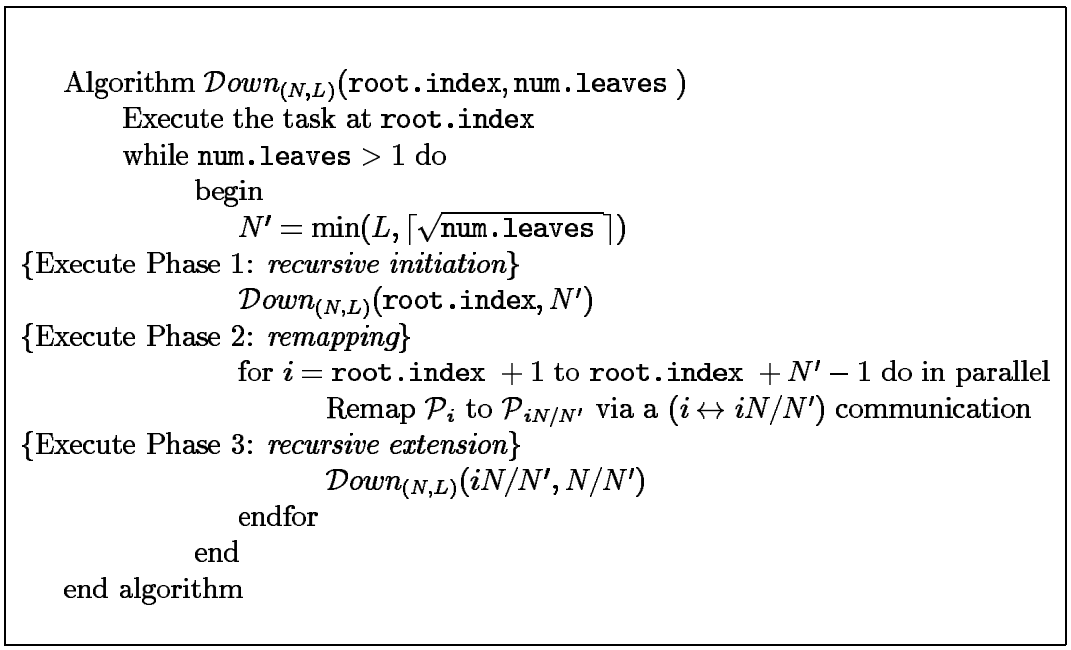


Figure 3: The broadcasting algorithm.

by assuming that num.leaves is always divisible by $\min(L, \lceil\sqrt{\texttt{num.leaves}}\rceil)$; only small additive errors result from this assumption.

Algorithm $\mathcal{U}p_{(N,L)}$ is just the dual of Algorithm $\mathcal{D}own_{(N,L)}$, with Phases 1 and 3 temporally interchanged. We leave both the informal and formal specification of $\mathcal{U}p_{(N,L)}$ to the reader.

2.1.2 Analysis of Algorithms $\mathcal{D}own_{(N,L)}$ and $\mathcal{U}p_{(N,L)}$

We now assess the time complexity of Algorithms $\mathcal{D}own_{(N,L)}$ and $\mathcal{U}p_{(N,L)}$. For simplicity, we analyze only Algorithm $\mathcal{D}own_{(N,L)}(0, N)$. The reader can easily adapt the analysis to both upward and/or segmented sweeps.

Theorem 2.1 *For $N \geq 4$ and $L \geq 2$, Algorithm $\mathcal{D}own_{(N,L)}(0, N)$ operates in time*

$$T(N, L) \leq \log N \log\log L + \frac{\log^2 N}{\log L}. \tag{2}$$

Proof. The following recurrence for $T(N, L)$ is clear from the description and specification of $\mathcal{D}own_{(N,L)}$.

$$T(N, L) \leq T(N', N') + \log N + T\left(\frac{N}{N'}, L\right), \tag{3}$$

where, as before, $N' = \min(L, \lceil\sqrt{N}\rceil)$. The first term in recurrence (3) is the time for the Phase-1 recursive invocation of the algorithm, which sweeps down the first N' levels of $\mathcal{T}_N$; the second term is the time for the message-transmissions that effect the Phase-2 remapping of the level-N' tree-nodes; the third term is the time for the Phase-3 recursive invocation of the algorithm, which sweeps down the remaining, bottom, levels of $\mathcal{T}_N$. Using the following easily verified initial condition for the recurrence,

$$T(4, 2) = 2,$$

we now bound the recurrence term by term.

We claim first that, for all $M \geq 4$,

$$T(M, M) \leq \log M \log\log M. \tag{4}$$

We proceed by induction on M. The basis is true, since $T(4, 4) \leq T(4, 2) = 2$. By invoking the inductive hypothesis and bound (3), we bound $T(M, M)$ for arbitrary $M > 4$ as follows.

$$\begin{aligned} T(M, M) &\leq 2T(\sqrt{M}, \sqrt{M}) + \log M \\ &\leq 2\log(\sqrt{M}) \log\log(\sqrt{M}) + \log M \\ &= \log M \log\log M. \end{aligned}$$

We now use bounds (3, 4) to establish the claimed bound (2) on $T(N, L)$.

$$\begin{aligned} T(N, L) &\leq \log N' \log\log N' + \log N + T(N/N', L) \\ &\leq (\log L \log\log L + \log N) \log_L N \\ &= \log N \log\log L + \log^2 N / \log L. \end{aligned}$$

The second step here uses bound (3) $\log_L N$ times to bound the $T(N/N', L)$ term. □

2.2 Specific LTS Algorithms

We describe here simple LTS algorithms for three fundamental operations: broadcasting, accumulation, and parallel-prefix [2]-[4].

One-to-all broadcasting. In the operation of *broadcasting*, one PE—with no loss of generality, $\mathcal{P}_0$—sends a single-packet message M to all other PEs. Let the PEs of the architecture be mapped to the nodes of a complete binary tree in any way that places PE $\mathcal{P}_0$ at the root of the tree. Now perform a sweep down the tree: as each PE-node receives the message M from its parent, it relays the message to both of its children. At the end of the downward sweep, each PE "knows" the message M.

In the sequel, we denote by $\mathcal{B}r_{(N,L)}$ the algorithm for broadcasting within an (N, L)-$\mathcal{RRP}$, that is based on the downward sweep algorithm $\mathcal{D}own_{(N,L)}$.

Accumulation. The operation of *accumulation* (or, *reduction*) is defined for any binary associative operator $\otimes$. The $\otimes$-reduction of the vector $\langle x_0, x_1, \ldots, x_{N-1}\rangle$ is the product $x_0 \otimes x_1 \otimes \cdots \otimes x_{N-1}$. An architecture whose PEs are indexed from 0 to $N-1$ in such a way that each PE $\mathcal{P}_i$ initially "knows" value x_i can compute this reduction as follows. Assign the nodes of $\mathcal{T}_N$ to the PEs in any way that assigns the leaves of $\mathcal{T}_N$ to the PEs in their natural left-to-right order. Now perform a sweep up the tree: each PE computes the $\otimes$-product of the quantities it receives from its children and passes this product to its parent. At the end of this upward sweep, the PE that is assigned the root of $\mathcal{T}_N$ "knows" the accumulated $\otimes$-product.

Parallel-prefix. The *parallel-prefix* (or, *scan*) operation is also defined for any binary associative operator $\otimes$. The $\otimes$-scan of the vector $\langle x_0, x_1, \ldots, x_{N-1}\rangle$ is the vector $\langle y_0, y_1, \ldots, y_{N-1}\rangle$, where each $y_i = x_0 \otimes x_1 \otimes \cdots \otimes x_i$. An architecture whose PEs are indexed from 0 to $N-1$ in such a way that each PE $\mathcal{P}_i$ initially "knows" value x_i can compute this scan as follows. Assign the nodes of $\mathcal{T}_N$ to the PEs in any way that assigns the leaves of $\mathcal{T}_N$ to the PEs in their natural left-to-right order. Now perform a sweep up the tree, followed by a sweep down the tree. During the upward sweep, the architecture performs an $\otimes$-reduction of the vector, but each PE retains (for the downward sweep) the value computed by its left child. During the downward sweep, each PE sends its retained value to its right child, which then computes the $\otimes$-product of its parent's retained value by its retained value (in that order). At the end of the downward sweep, each PE $\mathcal{P}_i$ "knows" the quantity y_i.

3 Lower Bounds for LTS algorithm on $\mathcal{RRP}$s

In this section, we prove lower bounds on the following quantities:

- $T^{\star}(N, L) \stackrel{\text{def}}{=}$ the fastest time in which an (N, L)-$\mathcal{RRP}$ can perform a one-to-all broadcast (Section 3.1),

- $T^\star(N) \stackrel{\text{def}}{=}$ the fastest time in which an N-$\mathcal{RRP}$ can perform a one-to-all broadcast (Section 3.2).

In particular, we prove that, to within constant factors, no (N, L)-$\mathcal{RRP}$ can perform a one-to-all broadcast in fewer than $(\log^2 N)/(\log L)$ steps; and, no N-$\mathcal{RRP}$—no matter how many buslines it has—can perform a one-to-all broadcast in fewer than $\log N \log\log N$ steps. Since broadcasting is intuitively the simplest 1-sweep LTS algorithm (an intuition that is verified in Section 4), these lower bounds demonstrate that our 1-sweep LTS algorithms in Section 2 (which operate within timebound (1)) cannot be sped up by more than a constant factor: our bound on $T^\star(N, L)$ provides the demonstration when the number L of buslines is small; our bound on $T^\star(N)$ provides the demonstration when the number L of buslines is large. We turn now to the details of our bounds and their proofs.

3.1 A Lower Bound on $T^\star(N, L)$

The following lower bound on $T^\star(N, L)$ establishes the optimality (to within a constant factor) of our 1-pass LTS algorithms $\mathcal{D}own_{(N,L)}$ and $\mathcal{U}p_{(N,L)}$, for "small" values of L, i.e., values no greater than $N^{1/\log\log N}$.

Theorem 3.1 *For all integers $N > 0$ and $L \geq 9$,*

$$T^\star(N, L) \geq \frac{1}{3}\frac{\log^2 N}{\log L} + \textit{l.o.t.} \tag{5}$$

Proof. Say that the (N, L)-$\mathcal{RRP}$ $\mathcal{R}$ has a PE $\mathcal{P}_0$ which wants to broadcast a one-word message M to all other PEs. We demonstrate that, no matter how $\mathcal{R}$ disseminates M to its PEs (subject, of course, to the limitations of $\mathcal{RRP}$s), it can decrease only slowly the size of the largest remaining block of PEs that are "ignorant" of message M. To the end of verifying this, let us consider the execution of an arbitrary broadcast algorithm $\mathcal{A}$ on $\mathcal{R}$.

At the beginning of the broadcast, only PE $\mathcal{P}_0$ "knows" message M, so the initial block of "ignorant" PEs is precisely the block $B_0 \stackrel{\text{def}}{=} B(1, N-1)$.

Let T_0 be the first instant in the execution of Algorithm $\mathcal{A}$ in which a message crosses either PE $\mathcal{P}_{\lceil N/3 \rceil}$ or PE $\mathcal{P}_{\lceil 2N/3 \rceil}$. By our delay model, $T_0 \geq \log N/3$. Because our $\mathcal{RRP}$ has L buslines, at time T_0, no more than $2L$ messages are "traveling" along the bus and directed to the block $B(\lceil N/3 \rceil, \lceil 2N/3 \rceil)$. Now, even if we assume that all of these $2L$

messages arrive at their destinations instantaneously at time T_0 (which can only speed up Algorithm $\mathcal{A}$), there must remain at time T_0 a block of "ignorant" PEs B_1 of size

$$|B_1| \geq \left\lceil \frac{N}{3(2L+1)} \right\rceil .$$

Let us now apply the above reasoning to the block B_1. Let T_1 be the first instant *after time* T_0 in the execution of Algorithm $\mathcal{A}$ in which a message crosses one of the two PEs located at $|B_1|/3$ positions to the right of the beginning of B_1 and $|B_1|/3$ positions to the left of the end of B_1. (In other words, we imagine that we "reset the clock" after time T_0 and begin looking at the indicated PEs.) Since we allowed all messages that were in transit at time T_0 to get to their destinations instantaneously, two things are clear. First,

$$T_1 \geq \log \frac{1}{3}|B_1| \geq \log \frac{N}{9(2L+1)}.$$

This follows just from the fact that no messages were in transit at time T_0 (because we let all of them arrive instantaneously), so the message that determines time T_1 must have traversed at least one-third of block B_1 since time T_0. Second, even if we assume that all of the $2L$ messages that could be in transit at time T_1 arrive at their destinations instantaneously, and that all of them are destined for PEs in the center of block B_1, there must remain at time T_1 a block of "ignorant" PEs B_2 of size

$$|B_2| \geq \left\lceil \frac{N}{3^2(2L+1)^2} \right\rceil .$$

We continue this reasoning with block B_2, which yields a block B_3 of "ignorant" PEs, then with block B_3, and so on. Eventually, we infer the existence of a sequence of time instants, $T_0, T_0 + T_1, T_0 + T_1 + T_2 \ldots$ in the execution of Algorithm $\mathcal{A}$, with associated blocks $B_0, B_1, B_2, \ldots$ of "ignorant" PEs, such that, for each i:

$$T_i \geq \log \frac{1}{3}|B_i| \text{ and } |B_i| \geq \frac{N}{3^i(2L+1)^i}.$$

Now, since Algorithm $\mathcal{A}$ must eventually get the broadcast message to every PE, the total time taken by $\mathcal{A}$ can be bounded as follows. Let

$$\lambda = \log_{3(2L+1)} N = \frac{\log N}{\log(3(2L+1))}.$$

Then, we have

$$\begin{aligned} T^{\star}(N,L) &\geq \sum_{i=0}^{\lambda-1} T_i + 1 \\ &\geq \sum_{i=0}^{\lambda-1} \log \frac{N}{3[3(2L+1)]^i} + 1 \\ &= \lambda \log N - \lambda \log 3 - \binom{\lambda}{2} \log(3(2L+1)) + 1. \end{aligned}$$

If we now substitute for λ in this bound and simplify, we obtain the desired bound (5). Since our reasoning holds for any broadcast algorithm on an (N,L)-$\mathcal{RRP}$, the theorem follows. □

By substituting L in the upper and lower bounds provided by Theorems 2.1 and 3.1, respectively, one establishes the following result.

Corollary 3.1 *If $L \leq N^{1/\log\log N}$, then Algorithm $\mathcal{B}r_{(N,L)}$ is within a constant factor of optimal on an (N,L)-$\mathcal{RRP}$.*

We close this section by noting that the bound of Theorem 3.1 trivializes for large L (e.g., $L \geq N^{1/\log\log N}$) to the observation that $T^{\star}(N,L)$ must be proportional to $\log N$. This bound tells us nothing about the complexity of the broadcast operation on $\mathcal{RRP}$s, because simply getting message M from PE $\mathcal{P}_0$ to PE $\mathcal{P}_{N/2}$ must take time proportional to $\log N$ (because of the subadditivity of logarithms). In the next subsection we derive a lower bound on $T^{\star}(N,L)$ that is nontrivial no matter how large L is, i.e., no matter how many buslines the $\mathcal{RRP}$ in question has. Moreover, this bound establishes the optimality (to within a constant factor) of $\mathcal{B}r_{(N,L)}$ (hence, of $\mathcal{D}own_{(N,L)}$ and $\mathcal{U}p_{(N,L)}$), even for large values of L.

3.2 A Limitation for All $\mathcal{RRP}$s

In this section, we prove that the time taken by an N-$\mathcal{RRP}$ $\mathcal{R}$ to broadcast must be proportional to $\log N \log\log N$, no matter how many buslines the $\mathcal{RRP}$ has. As usual, we assume, with no loss of generality, that PE $\mathcal{P}_0$ is broadcasting a message M to all other PEs. The reader should note that we could rephrase the proof of this bound to hold for any 1-sweep LTS algorithm. The rephrasing for downsweeps is little more than a change in terminology; the rephrasing for upsweeps is a bit more complicated, as one has to "run the proof backwards," which requires a bit of reformulation.

Theorem 3.2 *For all* $N > 4$, $T^\star(N) \geq \frac{1}{15} \log N \log\log N$.

Proof. We proceed by induction on N, using as a base all $N \leq 1024$. Our lower bound is trivial for N in this range, since it asserts only that an $\mathcal{RRP}$ requires more than two steps to broadcast when $N > 4$. Let us, therefore, focus on an arbitrary fixed $N > 1024$, and let us assume inductively that the theorem holds for all smaller N. Say that we have an optimal algorithm $\mathcal{A}$ that broadcasts on an N-$\mathcal{RRP}$ $\mathcal{R}$ within time $T^\star(N)$. By Theorem 2.1, we know that $\mathcal{A}$ operates within time $2 \log N \log\log N$. We can derive from $\mathcal{A}$ a *broadcast tree* $BT(\mathcal{A})$ whose structure exposes how $\mathcal{A}$ disseminates the broadcast message. The tree $BT(\mathcal{A})$ has node-set $\{0, 1, \ldots, N-1\}$; $BT(\mathcal{A})$ has an edge from node i to node j precisely if, in Algorithm $\mathcal{A}$, PE $\mathcal{P}_j$ of $\mathcal{R}$ receives the broadcast message *for the first time* via a direct communication from PE $\mathcal{P}_i$. Note that node 0 is the root of $BT(\mathcal{A})$.

Henceforth, let us focus on the broadcast of a message M by Algorithm $\mathcal{A}$.

The intuitive flow of our proof is as follows. From the structure of $BT(\mathcal{A})$ we find in $\mathcal{R}$ a block of adjacent PEs, called a *barrier*, that partitions a portion of the broadcast operation into three disjoint phases:

Phase 1. The root-PE $\mathcal{P}_0$ supplies the PEs in the barrier with message M.

Phase 2. The PEs in the barrier supply a new set of "subroot" PEs with message M.

Phase 3. The new subroot-PEs broadcast message M to the PEs in their broadcast subtrees.

The sum of the times for these three phases is clearly a lower bound on the total time for Algorithm $\mathcal{A}$'s broadcast. The strategy of our proof is to show that there must exist a barrier for which each of these three operations takes a rather long time. We achieve this by showing that there must exist a barrier whose PEs must supply message M to "many" subroots, all of which are "far" from the barrier and each of which must broadcast message M to PEs of $\mathcal{R}$ which belongs to a "big" subtree of $BT(\mathcal{A})$. One can view the rest of the proof as quantifying the quoted words in the preceding sentence.

We turn now to the details of the argument. Given a barrier B, we call any PE that receives the broadcast message—for the first time—directly from some PE of B a *subroot induced by* B. We say that a subroot $\mathcal{P}_i$ *covers* a PE $\mathcal{P}_j$ precisely when node j is in the subtree of $BT(\mathcal{A})$ rooted at node i; this is equivalent to saying that $\mathcal{P}_j$ receives the broadcast message for the first time from $\mathcal{P}_i$, either directly or indirectly. For any set of PEs P, we denote by $\text{COV}(P)$ the set of PEs $P \cup \{\text{PEs covered by PEs in } P\}$.

In order to define the barriers of interest, we first partition the (universal) block of PEs $B(0, N-1) = \{\mathcal{P}_0, \mathcal{P}_1, \ldots, \mathcal{P}_{N-1}\}$ into $\sqrt{N}-1$ subsets, each subset being the union of two blocks of PEs. For each $i \in \{1, 2, \ldots, \sqrt{N}-1\}$ the ith subset B_i is the set[6]
$B_i \stackrel{\text{def}}{=} B\left(-\lfloor \frac{1}{2}(\sqrt{N}-1)i \rfloor,\ -\lfloor \frac{1}{2}(\sqrt{N}-1)(i-1) \rfloor\right) \cup B\left(\lceil \frac{1}{2}(\sqrt{N}-1)(i-1) \rceil,\ \lceil \frac{1}{2}(\sqrt{N})i \rceil\right)$.
The *kth barrier* of $\mathcal{R}$ comprises the PEs whose indices are in the set $\mathcal{B}_k \stackrel{\text{def}}{=} \bigcup_{i=1}^{k} B_i$.

Notation. We denote the set of subroots induced by the kth barrier by R_k (see Figure 4). Further, for notational clarity, we henceforth abbreviate the quantity $\frac{1}{15}\log N \log\log N$ by λ_N.

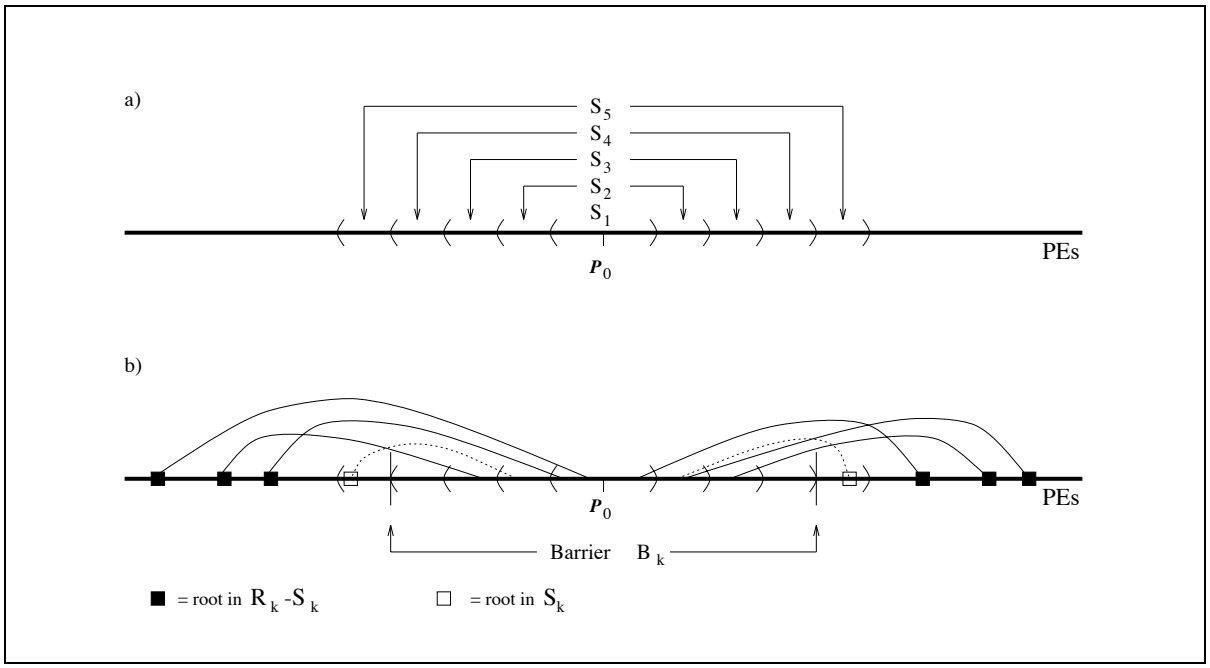


FIGURE 4: *An example of how we define (a) the subset B_i and (b) the subroots induced by a barrier.*

Our first lemma shows that $\mathcal{R}$ must have a barrier which induces subroots that are located "far" from the barrier and that cover "many" PEs. Specifically, the lemma is our first step in quantifying the qualifiers we have been putting in quotes.

Lemma 3.1 *There exists a $k \in \{1, 2, \ldots, \lambda_N^2\}$ such that*

$$|\text{Cov}(R_k - B_{k+1})| \geq \frac{N}{\lambda_N^2} - k\sqrt{N}. \tag{6}$$

[6]Recall that all arithmetic on the indices of PEs is modulo N, so that $\mathcal{P}_{-i} = \mathcal{P}_{N-i}$.

Proof. Let the parameter i range over the set $\{1, 2, \ldots, \lambda_N^2\}$. Note first that, for all i, the set of subroot PEs R_i covers at least $N - i\sqrt{N}$ PEs. Let us now consider the λ_N^2 sets of subroots $R_i \cap B_{i+1}$, for varying i (see Figure 4). We claim that at least one of these sets covers fewer than $N(1 - 1/\lambda_N^2)$ PEs. Were this not the case, each of the sets $\text{COV}(R_i \cap B_{i+1})$ would contain more than $N(1-1/\lambda_N^2)$ PEs. By the pigeon-hole principle, then, some PE would be covered by all λ_N^2 sets $R_i \cap B_{i+1}$. This, however, would imply that there is a "chain" of subroots,

$$r_1 \in R_1 \cap B_2, r_2 \in R_2 \cap B_3, \ldots, r_{\lambda_N^2} \in R_{\lambda_N^2} \cap B_{\lambda_N^2+1},$$

such that each subroot r_j covers subroot r_{j+1}. Such a chain of subroot dependencies is equivalent to a path in $BT(\mathcal{A})$ of length $\geq \lambda_N^2$, hence would imply that Algorithm $\mathcal{A}$ takes time $\geq \lambda_N^2$. This, however, contradicts our having chosen Algorithm $\mathcal{A}$ from among those broadcast algorithms that operate within time proportional to λ_N. This contradiction proves the lemma. □

Now let k_0 be an index for which the set $\mathcal{R}'_{k_0} \stackrel{\text{def}}{=} R_{k_0} - B_{k_0+1}$ has large coverage in the sense of inequality (6). Our second lemma shows that there are "many" remote subroots in $\mathcal{R}'_{k_0}$, each of which covers a "big" subtree of PEs.

Lemma 3.2 *There exists a nonempty set of subroots $\mathcal{R}''_{k_0} \subseteq \mathcal{R}'_{k_0}$ such that:*

(a) *The number of PEs covered by the subroots in $\mathcal{R}''_{k_0}$ exceeds*

$$\frac{1}{2}\left(\frac{N}{\lambda_N^2} - k_0\sqrt{N}\right).$$

(b) *For each subroot $r \in \mathcal{R}''_{k_0}$, the number of PEs covered by r exceeds*

$$\frac{1}{4\lambda_N|\mathcal{R}''_{k_0}|}\left(\frac{N}{\lambda_N^2} - k_0\sqrt{N}\right).$$

Proof. Let us index the elements of $\mathcal{R}'_{k_0}$ in nondecreasing order of coverage, that is, so that

$$|\text{COV}(\{r_1\})| \geq |\text{COV}(\{r_2\})| \geq \cdots \geq |\text{COV}(\{r_{|\mathcal{R}'_{k_0}|}\})|.$$

For each $i \in \{1, 2, \ldots, |\mathcal{R}'_{k_0}|\}$, let us denote by n_i and N_i, respectively, the cardinalities $n_i \stackrel{\text{def}}{=} |\text{COV}(\{r_i\})|$ and $N_i \stackrel{\text{def}}{=} |\text{COV}(\{r_1, r_2, \ldots, r_i\}|$. We now show that there is a nonempty set of subroots in $\mathcal{R}'_{k_0}$ which comprises the desired set $\mathcal{R}''_{k_0}$.

Let k_1 be the largest index i such that $N_i \leq 2i\lambda_N n_i$. Note that, since $N_1 < 2\lambda_N N_1 = 2\lambda_N n_1$, we are sure that k_1 "exists," i.e., that $k_1 \geq 1$. Let us abbreviate the quantity $|\mathcal{R}'_{k_0}|$ by S. Now, by definition of k_1, we know that, for each subroot r_i, where $k_1+1 \leq i \leq |\mathcal{R}'_{k_0}|$,

$$n_i < \frac{N_i}{2i\lambda_N} < \frac{N_S}{2i\lambda_N}.$$

It follows that[7]

$$\sum_{j=k_1+1}^{S} n_j \;<\; \frac{N_S}{2\lambda_N}\sum_{j=1}^{S}\frac{1}{j} \;\leq\; \frac{N_S}{2\lambda_N}(1+\ln S) \;\leq\; \frac{1}{2}N_S.$$

We now claim that the set $\{r_1, r_2, \ldots, r_{k_1}\}$ can serve as the sought set of subroots $\mathcal{R}''_{k_0}$. To wit, Lemma 3.1 assures us that

$$|\mathcal{R}''_{k_0}| = N_{k_1} \geq \frac{1}{2}N_S \geq \frac{N}{2\lambda_N^2} - \frac{k_0\sqrt{N}}{2}.$$

Moreover, by definition, each subroot $r_i \in \mathcal{R}''_{k_0}$ covers at least

$$n_i \geq n_{k_1} \geq \frac{N_{k_1}}{2k_1\lambda_N} > \frac{1}{4k_1\lambda_N}\left(\frac{N}{\lambda_N^2} - k_0\sqrt{N}\right)$$

PEs. $\mathcal{R}''_{k_0}$ thus yields the desired set of k_1 roots, which completes the proof. □

We are now ready to prove Theorem 3.2. Let k_0 and k_1 be the integers produced in the proofs of Lemmas 3.1 and 3.2, respectively. We bound the time that the optimal algorithm $\mathcal{A}$ takes for each of the three phases of the broadcast defined by the barriers we have selected.

The time for Phase 1. We claim that Algorithm $\mathcal{A}$ (indeed, any algorithm) needs time

$$T_1 \geq T^\star\left(\frac{k_1}{\log k_1 \log\log k_1}\right)$$

in order to generate the k_1 instances of message M that PEs within barrier $\mathcal{B}_{k_0} = \bigcup_{i=1}^{k_0} B_i$ must transmit (in Phase 2) to the k_1 "remote" subroot PEs of $\mathcal{R}''_{k_0}$.

In order to see this, say that, at the instant when *the last* copy of these k_1 instances of message M leaves barrier $\mathcal{B}_{k_0}$, there are p PEs in $\mathcal{B}_{k_0}$ that "know" message M. Since only such knowledgeable PEs can transmit the message instances, some one of these p PEs

[7] $\ln x$ denotes the natural logarithm of x.

must have transmitted at least k_1/p instances of M to the remote subroots. Furthermore, Algorithm $\mathcal{A}$ must have taken time at least $T^\star(p)$ to create p knowledgeable PEs. Thus, we have

$$T_1 \geq \min_p \max(T^\star(p),\ k_1/p). \tag{7}$$

Since $T^\star(x) \geq \log x$ for all x (see the closing paragraph of Section 3.1), the value of p that minimizes maximization (7) can be determined to have the form $p_0 = k_1/\log k_1 + \text{l.o.t.}$ Therefore, since $T^\star$ is (clearly) monotonically increasing as a function of N, we have

$$T_1 \ \geq \ T^\star\left(\frac{k_1}{\log k_1}\right) \ \geq \ T^\star\left(\frac{k_1}{\log k_1 \log\log k_1}\right). \tag{8}$$

The time for Phase 2. The k_1 instances of message M that are sent from within barrier $\mathcal{B}_{k_0}$ to the "remote" subroot PEs must pass over block B_{k_0+1} in transit. The time T_2 necessary for the last message instance to make this trip can be no smaller than

$$T_2 \geq \frac{1}{2}\log N, \tag{9}$$

because of the distance traveled.

The time for Phase 3. Finally, we focus on the last subroot to receive the message—call it r_0—that receives message M in Phase 2, and on the time it takes r_0 to relay message M to all the PEs in its subtree. By Lemma 3.2, r_0 covers no fewer than

$$\frac{N}{4k_1\lambda_N^3} - \frac{k_0\sqrt{N}}{4k_1\lambda_N} \ \geq \ \frac{N}{8k_1\lambda_N^3}$$

PEs; consequently, r_0 must take time at least

$$T_3 = T^\star\left(\frac{N}{8k_1\lambda_N^3}\right) \tag{10}$$

to transmit message M to these PEs.

Adding bounds (8), (9) and (10), we find that the optimal time to perform a one-to-all broadcast in a N-$\mathcal{RRP}$ is no smaller than

$$\begin{aligned} T^\star(N) \ &\geq \ T^\star\left(\frac{k_1}{\log k_1 \log\log k_1}\right) + \frac{1}{2}\log N + T^\star\left(\frac{N}{8k_1\lambda_N^3}\right) \\ &\geq \ \min_x \left\{ T^\star\left(\frac{x}{\lambda_N}\right) + \frac{1}{2}\log N + T^\star\left(\frac{N}{8x\lambda_N^3}\right)\right\} \end{aligned} \tag{11}$$

Now, since $T^\star(N) \leq 2 \log N \log\log N$, we know that the function $T^\star$ is subadditive; hence, the value of x that minimizes the final expression in (11) must satisfy

$$\frac{x}{\lambda_N} = \frac{N}{x\lambda_N^3},$$

so that

$$x = \frac{1}{2\sqrt{2}} \frac{\sqrt{N}}{\lambda_N}.$$

Thus, (11) yields the recurrence

$$T^\star(N) \geq 2T^\star\left(\frac{225}{2\sqrt{2}}\ \frac{\sqrt{N}}{(\log N \log\log N)^2}\right) + \frac{1}{2}\log N,$$

which we now set out to solve.

First, using the inductive hypothesis, we obtain

$$\begin{aligned}
T^\star(N) &\geq 2T^\star\left(\frac{225}{2\sqrt{2}} \frac{\sqrt{N}}{(\log N \log\log N)^2}\right) + \frac{1}{2}\log N \\
&\geq \frac{2}{15}\log\left(\frac{225}{2\sqrt{2}} \frac{\sqrt{N}}{(\log N \log\log N)^2}\right) \log\log\left(\frac{225}{2\sqrt{2}} \frac{\sqrt{N}}{(\log N \log\log N)^2}\right) + \frac{1}{2}\log N \\
&\geq \frac{1}{15}\log N \log\left(\log\frac{225}{2\sqrt{2}}\sqrt{N} - \log(\log N \log\log N)^2\right) \\
&\quad -\frac{2}{15}\log(\log N \log\log N)^2 \log\log\left(\frac{225}{2\sqrt{2}} \frac{\sqrt{N}}{(\log N \log\log N)^2}\right) + \frac{1}{2}\log N
\end{aligned}$$

Now, for any $N > 1024$, we have

$$\log\left(\frac{225}{2\sqrt{2}}\sqrt{N}\right) - \log(\log N \log\log N)^2 > \frac{5}{42}\log N;$$

therefore,

$$\begin{aligned}
T^\star(N) &\geq \frac{1}{15}\log N\left(\log\log N + \log\frac{5}{42}\right) \\
&\quad -\frac{2}{15}\log(\log N \log\log N)^2 \log\log\left(\frac{225}{2\sqrt{2}} \frac{\sqrt{N}}{(\log N \log\log N)^2}\right) + \frac{1}{2}\log N \\
&\geq \frac{1}{15}\log N \log\log N + \left(\frac{1}{2} - \frac{1}{15}\log\frac{42}{5}\right)\log N \\
&\quad -\frac{2}{15}\log(\log N \log\log N)^2 \log\log\left(\frac{225}{2\sqrt{2}} \frac{\sqrt{N}}{(\log N \log\log N)^2}\right)
\end{aligned}$$

Since

$$\frac{1}{2} - \frac{1}{15}\log\frac{42}{5} > 0.29,$$

and since for any $N > 1024$,

$$0.29 \log N > \frac{2}{15}\log(\log N \log\log N)^2 \log\log\left(\frac{225}{2\sqrt{2}}\frac{\sqrt{N}}{(\log N \log\log N)^2}\right),$$

we have finally shown that

$$T^\star(N) \geq \frac{1}{15}\log N \log\log N,$$

as was claimed. □

4 Nontrivial Algorithms on Arbitrary Architectures

This section is devoted to proving two rather surprising generalizations of Theorem 3.2. First, the lower bound of the Theorem holds not only for any single-sweep LTS algorithm, as shown in Section 3.2, but also for any algorithm that requires *nontrivial communication*. We say that an algorithm $\mathcal{A}$ *requires nontrivial communication* (for short, *is nontrivial*) if it requires some PE $\mathcal{P}$ to receive/send information—directly or indirectly—from/to *all* other PEs during the course of the computation.

The second generalization of Theorem 3.2 is perhaps even more surprising. The lower bound of the Theorem—even when generalized to all nontrivial algorithms—holds for a much broader class of parallel architectures than just $\mathcal{RRP}$s. In fact, the bound holds for parallel machines that communicate via *arbitrary* point-to-point fixed interconnection networks, provided only that the machines have been implemented in a way that satisfies the following conditions (which are quite consistent with current technology).

1. The machines must be laid out in some fixed number of spatial dimensions. For simplicity, we assume two-dimensional layouts in what follows, but our argument can be adapted easily to three dimensions.

2. Each PE of a machine must occupy a "unit-square" and can send at most one message per time step. (Of course, this condition really just defines our units.)

3. The message latency of the machine must be *at least logarithmic* in the Manhattan (rectilinear) distance in the layout, between the sending and receiving PEs, and it must be unaffected by the direction of the communication.

Let $\mathcal{M}$ be any N-PE parallel machine having point-to-point connections between PEs, which is implemented in such a way that these three conditions hold.

Theorem 4.1 *The N-PE machine $\mathcal{M}$ requires time proportional to $\log N \log\log N$ to execute any nontrivial algorithm $\mathcal{A}$.*

Proof. When machine $\mathcal{M}$ executes algorithm $\mathcal{A}$, there is at least one PE whose final state is affected by the initial state of every other PE; let $\mathcal{P}_0$ denote one such "sink" PE. We prove the theorem in two steps. First, we prove that the time that $\mathcal{M}$ takes to execute Algorithm $\mathcal{A}$ can be no smaller than the time that $\mathcal{M}$ takes to perform a one-to-all broadcast from PE $\mathcal{P}_0$. Second, we bound from below the time that $\mathcal{M}$ must take to perform this broadcast. The latter proof evolves in a manner similar to the proof of Theorem 3.2.

Let us attack the first portion of our proof by considering the *dependency tree* $DT(\mathcal{A})$ of Algorithm $\mathcal{A}$, that is constructed as follows. We assign PE $\mathcal{P}_0$ to be the root of $DT(\mathcal{A})$. We assign as the children of $\mathcal{P}_0$ those PEs—call them $\mathcal{P}_1, \mathcal{P}_2, \ldots, \mathcal{P}_k$—that communicated *directly* to $\mathcal{P}_0$ during the execution of $\mathcal{A}$. We assign as the children of $\mathcal{P}_0$'s child $\mathcal{P}_1$ those PEs that:

- communicated directly with $\mathcal{P}_1$ *before* its last communication with $\mathcal{P}_0$ *and*
- have not yet been assigned within the tree;

name these grandchildren of the root $\mathcal{P}_j$, for $j = k{+}1, k{+}2, \ldots$. We repeat the assignment process for PEs $\mathcal{P}_2, \mathcal{P}_3, \ldots$, in order of the indices we are assigning during this process, until all PEs of $\mathcal{M}$ are assigned to tree-nodes. The essential features of the assignment are that the children of PE $\mathcal{P}_i$ must have communicated directly with $\mathcal{P}_1$ *before* its last communication with its parent in the tree, *and* they must not yet have been assigned within the tree; we then index these new tree-nodes using the smallest as-yet unassigned indices. Now, it is obvious that $DT(\mathcal{A})$ is a spanning tree of the computation graph of Algorithm $\mathcal{A}$. Given that communication delay is not affected by the direction of the communication (by condition 3), we can conclude that the time taken by $\mathcal{M}$ to execute Algorithm $\mathcal{A}$ is no smaller than the time that $\mathcal{M}$ would take to perform a one-to-all broadcast from $\mathcal{P}_0$, using $DT(\mathcal{A})$ as the broadcast tree (cf. the proof of Theorem 3.2).

We next bound from below the time taken to perform the broadcast. To this end, we formally define the two-dimensional analog of the blocks B_i of Theorem 3.2, for our proof follows the logical flow of the proof of that Theorem.

Given any two-dimensional layout of machine $\mathcal{M}$, let us use the position of PE $\mathcal{P}_0$ as a reference point, in order to partition the PEs of $\mathcal{M}$ into two-dimensional blocks; we

call these blocks *squares* as an aid to the intuition. We effect the partition as follows: square SQ_i comprises those PEs whose distance from $\mathcal{P}_0$ in the tree $DT(\mathcal{A})$ is greater than $\frac{1}{2}N^{1/4}\sqrt{i-1}$ but less than $\frac{1}{2}N^{1/4}\sqrt{i}$ (see Figure 5). Let us now define the kth

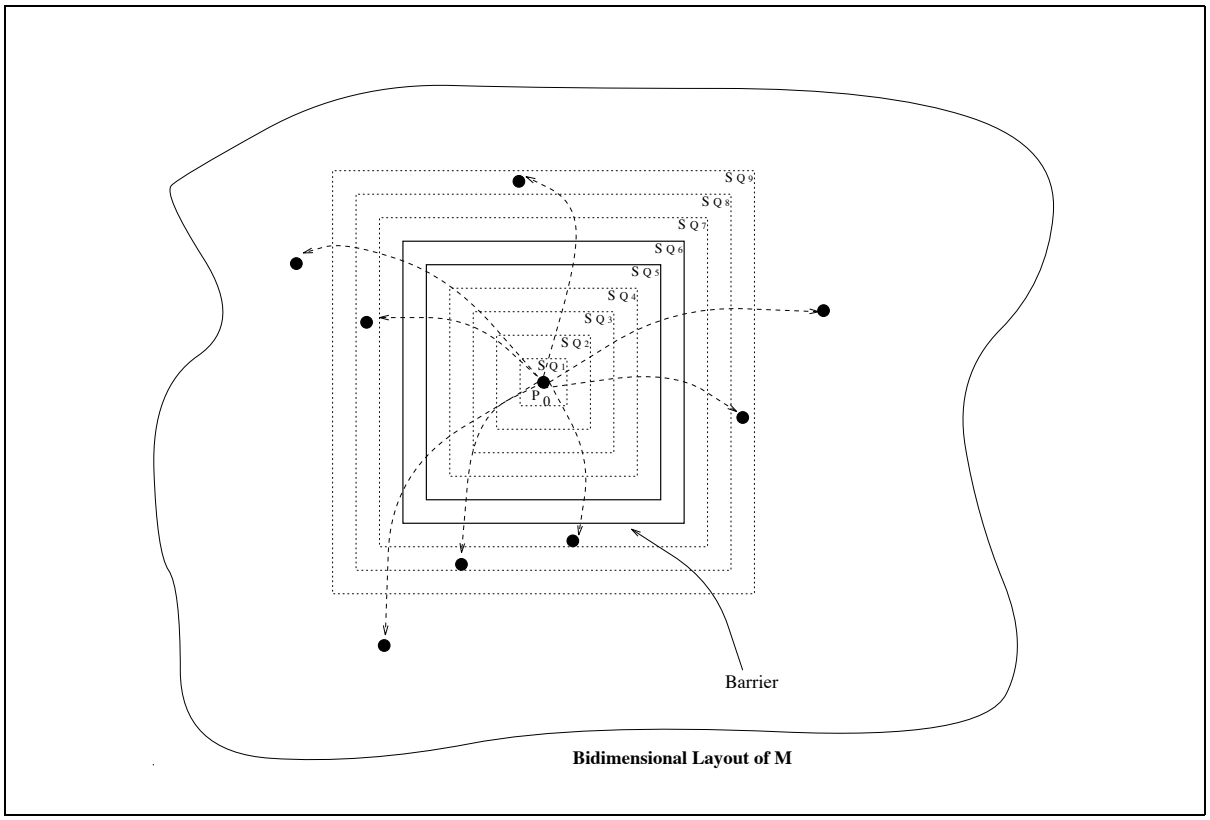


FIGURE 5: *An example of how we define the squares and the barrier in the bidimensional layout of M.*

barrier (of the layout) as the set of PEs $\bigcup_{i=1}^{k} \text{SQ}_i$. Finally, let us denote by R_k the set of subroots of the broadcast tree $DT(\mathcal{A})$ that learn the broadcast message directly from PEs in the kth barrier.

The following lemmas are the two-dimensional analogs of Lemmas 3.1 and 3.2 and are proved in much the same way.

Let T be the optimal time to broadcast using broadcast tree $DT(\mathcal{A})$.

Lemma 4.1 *There exists a $k \in \{1, 2, \ldots, T^2\}$, such that*

$$|\text{COV}(R_k - \text{SQ}_{k+1})| \geq \frac{N}{T^2} - k\sqrt{N}.$$

The interested reader can adapt the proof of Lemma 3.1 to prove Lemma 4.1. Let k_0 be an integer for which $\mathcal{R}'_{k_0} \stackrel{\text{def}}{=} R_{k_0} - \text{SQ}_{k_0+1}$ is large as in Lemma 4.1.

Lemma 4.2 *There exists a nonempty set of subroots $\mathcal{R}''_{k_0} \subseteq \mathcal{R}'_{k_0}$ such that:*

(a) *The number of PEs covered by the set $\mathcal{R}''_{k_0}$ exceeds*

$$\frac{1}{2}\left(\frac{N}{T^2} - k_0\sqrt{N}\right).$$

(b) *For each subroot $r \in \mathcal{R}''_{k_0}$, the number of PEs covered by r exceeds*

$$\frac{1}{8T|\mathcal{R}''_{k_0}|}\left(\frac{N}{T^2} - k_0\sqrt{N}\right).$$

Proof Sketch. Define N_i, n_i and S as in Lemma 3.2. Select x to be the largest i such that

$$n_i > \frac{N_i}{4iT}.$$

For each subroot r_i, where $x+1 \leq i \leq S$,

$$n_i < \frac{N_i}{4iT} < \frac{N_S}{4iT}.$$

Now,

$$\sum_{j=x+1}^{S} n_j \;\leq\; \sum_{j=1}^{S} n_j \;<\; \frac{N_S}{4T}\sum_{j=1}^{S}\frac{1}{j} \;\leq\; \frac{N_S}{4T}(\ln S + 1) \;\leq\; \frac{1}{2}N_S$$

where the last inequality comes from the fact that for any nontrivial computation, $T \geq \frac{1}{2}\log N$. The rest of the proof is similar to Lemma 3.2 and is left to the reader. □

The proof of the theorem now proceeds by induction on N and is similar to that of Theorem 3.2. □

Acknowledgments. It is a pleasure to acknowledge numerous helpful conversations with Marc Picquendar.

The research of the first two authors was supported in part by NSF Grant CCR-92-21785; the research of the third author was supported in part by NSF Grant CCR-94-10077. A portion of this research was done while the first author was visiting the Department of Computer Science, The Technion, Haifa, Israel.

References

[1] G. Bilardi and F.P. Preparata (1992): Horizons of parallel computation. In *Symp. for the 25th Anniversary of INRIA. Lecture Notes in Computer Science 653*, Springer-Verlag, Berlin, pp. 155-174.

[2] G.E. Blelloch (1989): Scans as primitive parallel operations. *IEEE Trans. Comp. 38*, 1526-1538.

[3] S.L. Johnsson (1993): Massively parallel computing: data distribution and communication. In *Parallel Architectures and Their Efficient Use: Proceedings of the 1st Heinz-Nixdorf Symposium*, Paderborn, Germany (F. Meyer auf der Heide, B. Monien, A.L. Rosenberg, eds.) *Lecture Notes in Computer Science 678*, Springer-Verlag, Berlin, pp. 68-92.

[4] F.T. Leighton (1992): *Introduction to Parallel Algorithms and Architectures: Arrays, Trees, Hypercubes.* Morgan Kaufmann, San Mateo, Cal.

[5] C. Mead and L. Conway (1980): *Introduction to VLSI Systems.* Addison-Wesley, Reading, Mass.

[6] R. Miller, V.K. Prasanna-Kumar, D. Reisis, Q.F. Stout (1993): Parallel computations on reconfigurable meshes. *IEEE Trans. Comp. 42*, 678-692.

[7] A.L. Rosenberg, V. Scarano, R.K. Sitaraman (1994): The Reconfigurable Ring of Processors: efficient algorithms via hypercube simulation. *Parallel Proc. Let.* (Special Issue on Dynamically Reconfigurable Architectures).

[8] P.M.B. Vitanyi (1988): A modest proposal for communication costs in multicomputers. In *Concurrent Computations: Algorithms, Architecture, and Technology* (S.K. Tewksbury, B.W. Dickinson, S.C. Schwartz, eds.) Plenum Press, N.Y., 203-216.