# Assigning labels in unknown anonymous networks

## [Extended Abstract]

Pierre Fraigniaud [*] Andrzej Pelc [†] David Peleg [‡] Stéphane Pérennes [§]

## ABSTRACT

We consider the task of distributedly assigning distinct labels to nodes of an unknown anonymous network. *A priori*, nodes do not have any identities (anonymous network) and do not know the topology or the size of the network (unknown network). They execute identical algorithms, apart from a distinguished node, called the source, which starts the labeling process. Our goal is to assign short labels, as fast as possible. The *quality* of a labeling algorithm is measured by the range from which the algorithm picks the labels, or alternatively, the length of the assigned labels. Natural *efficiency* measures are the time, i.e., the number of rounds required for the label assignment, and the message and bit complexities of the label assignment protocol, i.e., the total number of messages (resp., bits) circulating in the network. We present label assignment algorithms whose time and message complexity are asymptotically optimal and which assign short labels. On the other hand, we establish inherent trade-offs between quality and efficiency for labeling algorithms.

## 1. INTRODUCTION

### 1.1 The problem

Designing network algorithms without complete information about the network is an important problem whose many variations have been extensively studied. Of particular interest are computations in anonymous networks (cf. [2, 3, 4, 15, 9, 12, 13, 7, 16]), in which processors do not have distinct identities and execute identical algorithms. The impossibility of distinguishing processors yields symmetry in computations and restricts the computational power of the network.

The situation is even more drastic if the topology of the network and even its size are unknown, and the only knowledge available to a node is its own degree. Such is, e.g., the situation when a robot has to explore an unknown graph, in order to draw a map of it (cf. [1, 8, 14]), or when a node has to broadcast information to all nodes of an unknown network [6, 11].

Known characterizations of the attainable and unattainable tasks in unknown anonymous networks reveal that assuming no means of symmetry breaking, little can be done. In particular, it is impossible to select a unique leader, or to assign unique labels (or ID's) to the network nodes. On the other hand, it is equally well known (see, for example, [15] and the references therein) that even in an unknown anonymous network, the existence of a unique leader in the network makes it computationally equivalent to a full-knowledge environment, in which unique node labels exist, and every node in the network knows the entire topology. From a complexity-oriented point of view, however, there is a rather significant difference between the two environments, in the sense that the cost of transforming from the former environment to the latter may be expensive.

In this paper we consider the cost of the first step along this transformation, namely, the task of assigning distinct labels to nodes of an unknown anonymous network. Thus the initial state is that of complete ignorance; nodes do not know the topology or even the size of the network, and they do not have any labels. The only knowledge available to a node is its own degree. Asymmetry is provided by postulating the existence of a unique node $s$, called the *source*, which effectively acts as the leader and initiates the labeling process. All other nodes execute identical algorithms.

We assume that communication between nodes is executed in *rounds* controlled by a global clock. A message sent by a node to its neighbor in a given round becomes available at the neighbor in the next round. The goal of a labeling protocol is to assign distinct labels to all nodes of the network.

[*]Laboratoire de Recherche en Informatique - CNRS, Université Paris-Sud, 91405 Orsay, France. (pierre@lri.fr)

[†]Département d'Informatique, Université du Québec à Hull, Hull, Québec J8X 3X7, Canada. (pelc@uqah.uquebec.ca)
Supported in part by NSERC grant OGP 0008136. Research done during this author's visit at INRIA Sophia Antipolis.

[‡]Department of Computer Science and Applied Mathematics, The Weizmann Institute, Rehovot 76100, Israel. (peleg@wisdom.weizmann.ac.il)
Supported in part by a grant from the Israel Ministry of Science and Art.

[§]SLOOP I3S-CNRS/INRIA, Université de Nice-Sophia Antipolis, 2004 Route des Lucioles, BP 93, F-06902 Sophia Antipolis Cedex, France. (Stephane.Perennes@sophia.inria.fr)


PODC 2000 Portland Oregon

## 1.2 Models and complexity measures

The natural measure for the *quality* of the solution is the range from which the algorithm picks the labels, or alternatively, the length of the assigned labels. The significance of this parameter is that in any subsequent use of the labels within any distributed algorithm, the label length will directly affect the size of the messages sent by the algorithm. Hence it is desirable to assign labels which are as short as possible.

On the other hand, the natural *efficiency* measures are the time, i.e., the number of rounds required for the label assignment, and the message and bit complexities of the label assignment protocol, i.e., the total number of messages (resp., bits) circulating in the network. As one might expect, our results imply that there are inherent tradeoffs between the quality and efficiency parameters for the label assignment problem.

It turns out that the precise communication model used significantly impacts the results. We consider three natural models. In the *all-port* model, every node can send a message to each of its neighbors, as well as receive a message from each neighbor, in each round. At the other extreme, the *one-port* model allows each node, in each round, to send only one message to a single neighbor, or to receive a single message from a neighbor, but not both. Finally, in the intermediate (or *mixed* model, every node is allowed to send only a single message (to one neighbor) per round, but it may receive messages from many neighbors simultaneously.

Since nodes are ignorant of network topology, they cannot distinguish between yet unused adjacent links. Consequently, the local decision made by a node, regarding which of its outgoing links it should use to send a message in a particular round, can be thought of as fixed by an *adversary*, as in [1, 8, 11, 14]. We adopt the worst-case approach, i.e., we are interested in the performance of the protocol resulting from the most detrimental behavior of the adversary. In particular, we are interested in the time requirement of the algorithm on the *worst-case network* of a given size, as well as in the *optimality* of the algorithm on *every* network. We say that an algorithm $A$ has *asymptotically optimal time* if its time complexity is bounded by a constant times the optimal time, for every network $G$ and source $s$.

To illustrate the difficulties of the problem, let us first present two naive algorithms for it on a tree. The first algorithm is based on a single message (the "token"), that traverses the graph in depth-first fashion, starting at the source $s$ and visiting one vertex at a time, assigning a distinct label to each visited node. (See Fig. 1.) While this algorithm assigns the smallest possible labels (from the range $[1, n]$), its time requirements are $2n$ *on any* $n$-node tree, which is about twice the lower bound for the worst case, and more importantly, very far from optimal on many trees (e.g., complete binary trees).

A second naive algorithm, named `0-1-Split`, is best illustrated on a binary tree. The source labels itself by the empty sequence $\varepsilon$, and every non-leaf node labeled $L$ assigns the labels $L0$ and $L1$ to its left and right children, respectively. This algorithm clearly achieves optimal time in the all-port

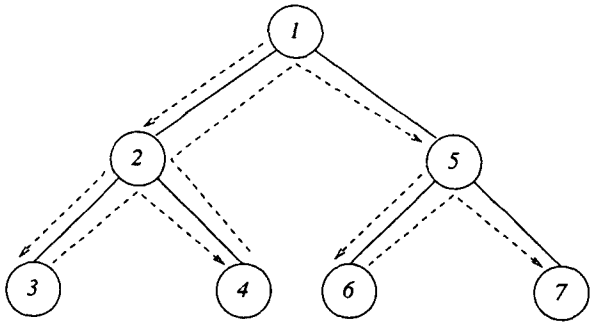


**Figure 1:** Labeling using a DFS based algorithm.

model on any tree, and in particular, its worst-case time is $n-1$, but on the other hand it generates very long (potentially $\Omega(n)$-bit) labels (see Fig. 2).

## 1.3 Results

While we have developed a number of algorithms for the labeling problem, our main positive results revolve around variants of one main algorithm, named `Wake & Label`, which seems to fare rather well (in fact, near-optimally) w.r.t. all of our quality and efficiency measures.

For presenting our results in more detail, it is convenient to start with the intermediate mixed model, and then describe the way the results change when we deviate to either of the two extreme models.

We first consider the situation when the network is an unknown tree. The tree is *a priori* non-rooted, but for our purposes it can be viewed as rooted at the source $s$, as the source originates the labeling process. The main variant of Alg. `Wake & Label` operates in this setting. For $n$-node trees, it assigns labels of size $O(\log n)$ and its worst-case time complexity is $n+1$ (while the worst-case lower bound on time is obviously $n-1$, e.g., for the path). Its message and bit complexities are also asymptotically optimal: $O(n)$ and $O(n \log n)$, respectively.

Another significant advantage of this algorithm reveals itself when one considers the question of time optimality. Let us first introduce an important parameter affecting the time required for label assignment. For any network $G$, define $bb(G, s)$ to be the worst-case time of broadcasting a message from a source $s$ to all nodes of $G$. (The notation $bb$ stands for

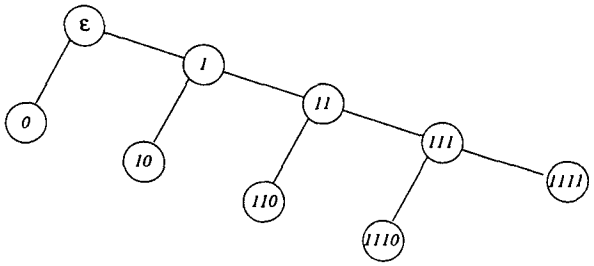


**Figure 2:** Labeling using the 0-1-Split algorithm.

"blind broadcast," as nodes do not know to whom they send messages.) Alternatively, for a tree $T$ rooted at $s$, $bb(T,s)$ is the maximum sum of node out-degrees, taken over all branches of $T$. A central observation for the purposes of the current paper is that for any graph $G$ and source $s$, the time required for label assignment is lower bounded by $bb(G,s)$, namely, $\tau(G,s) \geq bb(G,s)$.

Returning to our Alg. `Wake & Label`, we note that it runs in asymptotically optimal time, namely, it uses $3 \cdot bb(T,s)$ time, which is at most 3 times larger than worst-case optimal for any given tree.

Two other protocols presented for the problem are also fast in the worst case, but have contrasting qualities and drawbacks. In particular, our second algorithm (`Label-Passing`) assigns the shortest possible labels to all nodes, i.e., labels $1,\ldots,n$, but on the other hand, it has large message and bit complexities: $O(n^2)$ and $O(n^2 \log n)$, respectively, and its time is always $n$ (regardless of the tree), hence it is far from being time-optimal. Our third algorithm (`Fastest`) achieves the optimal time $\tau(T,s)$ for any given tree $T$ and source $s$, and uses $O(n)$ messages, but its only guarantee on label size is $O(\sqrt{n})$.

To complete the picture, we show two lower bounds for labeling trees. One says that any algorithm working in optimal time $\tau(T,s)$ for any given tree $T$ and source $s$, requires labels of size $\Omega(\sqrt{n})$, hence Alg. `Fastest` is the best possible in this sense. The second is a lower bound on the size of labels produced by protocols whose message complexity is strictly optimal, i.e., those using exactly $n-1$ messages. We prove that such protocols must produce labels of length $\Omega(n)$, on some $n$-node trees.

For general unknown $n$-node networks, a variant of Algorithm `Wake & Label` named $\texttt{Wake \& Label}_B^G$ assigns labels from the range $[1,n]$ in time $3n$. We also present a slightly faster protocol (with worst-case time $2n$) but this protocol assigns labels of size up to $\Omega(n)$ in some cases. This is complemented by observing that there are networks for which labeling in $[1,n]$ cannot be done faster than in time $2n-o(n)$. We then present a variant of Alg. $\texttt{Wake \& Label}_B^G$ assigning shortest possible labels, and working in asymptotically optimal time for any given network.

Finally, we discuss the changes in our results when we consider the problem in the all-port or the one-port models.

In the all-port model we give another variant of Algorithm `Wake & Label` (named Alg. `All-port 3-Phase`) which assigns optimal labels, and works in asymptotically optimal time $3 \cdot ecc(G,s)$, where $ecc(G,s)$ denotes the *eccentricity* of the source $s$ in the network $G$, namely, the maximum distance between $s$ and any other node of the network. This algorithm is proved to be strictly optimal for trees: We show that any algorithm assigning labels from the range $[1,n]$ to all $n$-node trees must use time at least $3 \cdot ecc(T,s) - 2$ for some tree $T$ and some source $s$. On the other hand, any algorithm working in optimal time (namely, time equal to the eccentricity of the tree w.r.t. the source) must assign labels of size $\Omega(n)$. We also show another (DFS-based) algorithm which also assigns labels from the range $[1,n]$. This algorithm works in time $2n$ on any $n$-node network, hence it is slightly better in the worst case (the worst-case lower bound on time is $n-1$), but is not asymptotically optimal.

In the one-port model the DFS-based algorithm works in asymptotically optimal time $2m$ for any $m$-edge network, as it can be proven that for general networks, $m$ is a lower bound on time. On the other hand, if we restrict attention to trees, another algorithm works in asymptotically optimal time, namely, it uses time at most 3 times larger than worst-case optimal, for any given tree. (It should be noted that if the network is known to be a tree, $m = n-1$ is not a lower bound on time any more: as in the mixed model, the lower bound becomes $bb(T,s)$.)

Our results are summarized in the table of Fig. 3.

In what follows, Sections 2 and 3 deal with trees and general networks in the intermediate mixed model. Section 4 addresses the all-port and one-port models. Table 3 summarizes our results.

## 2. LABELING TREES

In this section we study the label assignment problem for trees, in the mixed model. We consider an $n$-node tree $T$. Our algorithms view the tree as rooted at the source $s$, and use the terms *parent*, *child*, *ancestor*, *descendant* and *leaf* with respect to this rooted tree. The tree $T$ and the integer $n$ are unknown to the nodes.

A natural first question to be considered regarding the tree labeling problem is whether quality and efficiency can both be optimized in the worst-case, i.e., whether it is possible to label any $n$-node tree in $n-1$ or fewer rounds using the minimal possible labels, $[1,n]$. Curiously, it turns out that the answer to this question is positive. In the full paper (see [10]) we present a simple algorithm named `Label-Passing`, analyze its performance and establish the following theorem.

THEOREM 2.1. *Given an $n$-node tree and a source $s$, Algorithm* `Label-Passing` *assigns distinct labels from the range $[1,n]$ to all the nodes. The label assignment completes in $n-1$ rounds. The time, message and bit complexities of the algorithm are $n$, $O(n^2)$ and $O(n^2 \log n)$, respectively.*

### 2.1 Algorithm `Wake & Label`

Our main algorithm, Algorithm `Wake & Label`, is slightly inferior to Algorithm `Label-Passing` with respect to worst-case time complexity and label size. Specifically, it assigns labels of asymptotically optimal length $O(\log n)$, and works in worst-case time at most $n+1$. On the positive side, its message and bit complexities are far better, namely, $O(n)$ and $O(n \log n)$, respectively. But most importantly, Alg. `Wake & Label` works in asymptotically optimal time. This overcomes the main disadvantage of Alg. `Label-Passing`, which is that its execution time is *always* at least $n-1$, regardless of the tree structure.

#### 2.1.1 *Three simplified variants*

The idea of Alg. `Wake & Label` significantly differs from that of Alg. `Label-Passing`.

| Model | Networks | Algorithm | Time-worst | Time-per graph | Label size | Messages |
|---|---|---|---|---|---|---|
| Mixed | Trees | `Label-Passing` | $n$ | $n$ | $\lceil \log n \rceil$ | $O(n^2)$ |
| | | `Wake & Label` | $n+1$ | $3 \cdot bb(T,s)$ | $O(\log n)$ | $O(n)$ |
| | | `Fastest` | | $bb(T,s)$ | $O(\sqrt{n})$ | $O(n)$ |
| | | Lower bounds | $n-1$ | $bb(T,s)$ | $\lceil \log n \rceil$ | $n-1$ |
| | | Tradeoff | | | $\Omega(n) \Leftarrow$ | $n-1$ |
| | | Tradeoff | | $bb(T,s)$ | $\Rightarrow \Omega(\sqrt{n})$ | |
| | Graphs | `0-1-Split`$^G$ | $2n$ | | $n$ | $O(nm)$ |
| | | `Wake & Label`$_B^G$ | $3n$ | | $\lceil \log n \rceil$ | $O(m)$ |
| | | `Wake & Label`$_A^G$ | | $4 \cdot bb(G,s)+1$ | $\lceil \log n \rceil$ | $O(m)$ |
| | | Lower bounds | $2n-o(n)$ | $bb(G,s)$ | $\lceil \log n \rceil$ | $m$ |
| All-Port | Trees | Mixed Alg's | same | same | same | same |
| | | `All-port 3-Phase` | $3n$ | $3 \cdot ecc(T,s)-2$ | $\lceil \log n \rceil$ | $O(n)$ |
| | | Lower bounds | $n-1$ | $ecc(T,s)$ | $\lceil \log n \rceil$ | $n-1$ |
| | | Tradeoff | | $ecc(T,s)$ | $\Rightarrow \Omega(n)$ | |
| | | Tradeoff | | $3 \cdot ecc(T,s)-2 \Leftarrow$ | $\lceil \log n \rceil$ | |
| | Graphs | Mixed Alg's | same | same | same | same |
| | | `DFS`$^+$ | $2n$ | $2n$ | $\lceil \log n \rceil$ | $O(m)$ |
| | | `All-port 3-Phase` | $3n$ | $3 \cdot ecc(G,s)+1$ | $\lceil \log n \rceil$ | $O(m)$ |
| | | Lower bounds | $n-1$ | $ecc(G,s)$ | $\lceil \log n \rceil$ | $m$ |
| One-Port | Trees | `DFS` | $2n-3$ | $2n-3$ | $\lceil \log n \rceil$ | $2n-3$ |
| | | `Time-Slots` | $3n$ | $3 \cdot bb(T,s)$ | $\lceil \log n \rceil$ | $O(n)$ |
| | | Lower bounds | $n-1$ | $bb(T,s)$ | $\lceil \log n \rceil$ | $n-1$ |
| | Graphs | `DFS` | $2m$ | $2m$ | $\lceil \log n \rceil$ | $O(m)$ |
| | | Lower bounds | $m$ | $m$ | $\lceil \log n \rceil$ | $m$ |

**Figure 3:** Summary of results: Algorithms, lower bounds and tradeoffs.

As the algorithm and its analysis are rather complex, it is convenient to first present a succession of three simplified variants of the algorithm, of increasing complexity. These variants are slightly weaker, but somewhat more intuitive.

All variants of Alg. `Wake & Label` operate in three phases and use three types of messages: *wakeup*, *count* and *allocation*. Moreover, in the first three variants we describe, the first two phases are similar.

In the first (*wakeup*) phase, the source wakes up all the nodes by broadcasting a wakeup message. The second (*counting*) phase is started by the leaves of the tree $T$, and provides every node $v$ with a count of the number of nodes in its subtree $T_v$. This is done in a bottom-up convergecast fashion, using count messages of size at most $\log n$ which travel bottom-up in $T$.

The differences between the first three variants of Algorithm `Wake & Label` are manifested in the third (*allocation*) phase, whose task is to distribute the labels via allocation messages. In our first variant, named Alg. `Wake & Label`$_A$, this phase is straightforward. It is started by the source immediately upon receiving count messages from all its children. The source assigns itself label 1, and sends to each child a disjoint integer interval corresponding to the size of the subtree rooted at this child. Likewise, any node that got interval $[a,b]$ assigns itself label $a$ and sends to each child a disjoint integer interval corresponding to the size of the subtree rooted at this child. Hence this algorithm assigns labels from the optimal range $[1,n]$, and has asymptotically optimal time, message and bit complexities, but its worst-case time complexity is three times greater than the optimal, namely, $3(n-1)$. This worst case is realized on the $n$-node path with the source residing at one of the endpoints. Hence our efforts focus on reducing the worst-case time complexity to $n+1$ while preserving the above qualities as much as possible.

The second variant, named `Wake & Label`$_B$, is based on observing that the source does not necessarily need to wait until receiving a count from *all* of its children. Rather, it can start the label distribution once it knows the counts of all its children *but one*. In this case, it assigns consecutive blocks of labels to the children for which it knows the count, and then passes the remaining infinite (semi-open) block of labels to the remaining child. This child, in turn, behaves the same. It can be shown that this idea is sufficient for reducing the worst-case time complexity of the algorithm to $3n/2$. The worst example is, e.g., the $n$-node path where the source resides at the middle node.

*Algorithm* `Wake & Label`$_C$
The third variant, `Wake & Label`$_C$, assigns labels of length $O(\log n \log d)$, where $d$ is the maximum degree of the tree, and has time complexity $n+d$ and bit complexity $O(n \log n \log d)$. The labels of all nodes are of the form $(p;y)$, where $p$ is a sequence of integers smaller than $d$, called the *prefix*, and $y$ is a natural number smaller than $n$, called the *value*.

The first two phases of the algorithm remain as before. However, each node waits for the count messages only for a prescribed amount of time, which still guarantees that the deadline $n+d$ for the whole process is met in the worst case. If

by the end of the waiting period only one child has not reported, labeling subtrees of all other children can be done efficiently, using consecutive integers as values and keeping the prefix unchanged. If, on the other hand, at least two children did not report (such a situation is called *special*), labels of these children and all their descendants get a prefix which is an extension of that of the current node. It turns out that the length of label given to any node $v$ is $O(\log n + \sigma)$, where $\sigma$ is the number of special situations occurring on the path from $s$ to $v$. The crucial issue is to show that waiting periods are sufficiently long to guarantee that $\sigma$ is not too large.

Allocation messages travel top-down and their size is $O(\log n \log d)$. In order to describe them, we introduce the crucial notions of *initial* and *final capital* at node $v$. These are natural numbers denoted by $c'(v)$ and $c''(v)$, respectively. Initially $c'(s) = 1$. For $v \neq s$, $c'(v)$ will be set to $c''(u)$, where $u$ is $v$'s parent ($c''(v)$ is set by the protocol as the execution develops). Intuitively, capital is defined in such a way that node $v$ can wait $c'(v)$ rounds between sending the last wakeup message and the first allocation message, without violating the overall deadline for algorithm completion in the subtree rooted at $v$.

After sending all wakeup messages, node $v$ starts a waiting period which continues (at least) until it receives an allocation message from its parent $u$. The allocation message contains $c''(u)$, and $v$ sets $c'(v) \leftarrow c''(u)$. If $v$ received the allocation message while sending wakeup messages, then $v$ will wait $c'(v)$ rounds after it will have finished sending wakeup messages. If, on the other hand, $v$ received the allocation message after it has already finished to send its wakeup messages (say, $t_0$ rounds later), then $v$ acts as follows. If $c'(v) > t_0$, then $v$ continues its waiting period for $c'(v) - t_0$ additional rounds. Otherwise, the waiting period is over. The source behaves as if receiving "from its parent" an allocation message with capital 1 at time 0, that is the source will wait 1 rounds after having sent all its wakeup messages.

At the end of this waiting period, there are two possibilities:

1. $v$ got count messages from all of its children except possibly one,
2. $v$ did not get count messages from at least two children.

The first situation is called *normal*, the second is called *special*. We first describe the actions for each situation when $v = s$ is the source. Note that $s$ has waited $c'(s) = 1$ time unit after having sent the last wakeup message. It now assigns itself the label $(\varepsilon; 0)$, where $\varepsilon$ denotes the empty sequence.

In the normal situation, $c''(s) \leftarrow 1$ $(= c'(s))$ and $s$ sends allocation messages to its children, in consecutive time units, in the same order in which it sent wakeup messages. Children from which $s$ got a count message get a *closed* allocation message, and the (at most one) child from which no count message arrived gets an *open* allocation message. Suppose, without loss of generality, that $v_i$, $i = 1, \ldots, r$, are those children of $s$ from which count messages arrived saying that the subtree rooted at $v_i$ has size $x_i$, and no count message arrived from child $v_{r+1}$. Then the closed messages sent to $v_i$, $i = 1, \ldots, r$, consist of the prefix $\varepsilon$ (the empty sequence) and of intervals $I_i$, $i = 1, \ldots, r$, of natural numbers, such that the left end of $I_1$ is 1, the left end of $I_{i+1}$ is the successor of the right end of $I_i$, and $|I_i| = x_i$. The open message sent to $v_{r+1}$ consists of the final capital $c''(s)$, the prefix $\varepsilon$ and of the half-line $[t, \infty)$, where $t$ is the successor of the right end of $I_r$. Integers from these intervals will be used to construct labels of nodes in respective subtrees. (Intervals are coded by pairs of their ends and the half-line $[t, \infty)$ is coded by $t$.)

In the special situation, $c''(s) \leftarrow 2$ $(= c'(s) + \lceil c'(s)/2 \rceil)$, and the source sends *special* allocation messages to its children, in the same order in which it sent wakeup messages. A special message sent to child $v_i$ consists of a marker *special*, of the final capital $c''(s)$ and of the prefix which is the one-term sequence $(i)$.

Let $v \neq s$ and let $u$ be $v$'s parent. At the end of the waiting period, there are three possible cases.

**(1)** The received allocation message was closed, consisting of the prefix $p$ and the interval $[a, b]$. In this case, node $v$ assigns itself the label $(p; a)$ and sends to its children $v_i$, $i = 1, \ldots, r$, a closed allocation message consisting of the prefix $p$ and of intervals $I_i$, $i = 1, \ldots, r$ of natural numbers, such that the left end of $I_1$ is $a + 1$, the left end of $I_{i+1}$ is the successor of the right end of $I_i$, and $|I_i| = x_i$, where $x_i$ is the size of the subtree rooted at $v_i$ (reported to $v$ by a count message).

**(2)** The received allocation message was open, consisting of the final capital $c''(u)$, prefix $p$, and half-line $[t, \infty)$. In this case, node $v$ assigns itself the label $(p; t)$. It sets $c'(v) \leftarrow c''(u)$. At the end of the waiting period there are two possibilities, as described above for the source $s$: normal and special. In the normal situation, $c''(v) \leftarrow c'(v)$, and node $v$ acts as did the source, except that the prefix sent is $p$ rather than $\varepsilon$ (the empty sequence), and the left end of interval $I_1$ is $t + 1$ rather than 1. In the special situation, $c''(v) \leftarrow c'(v) + \lceil c'(v)/2 \rceil$, and $v$ sends special allocation messages to its children, in the same order in which it sent wakeup messages. (As will be clarified later, this formula is motivated by the fact that in the special situation, a child $w$ of $v$ has at least $c'(v)$ nodes in its subtree $T_w$.) A special message sent to child $v_i$ consists of a marker *special*, of the final capital $c''(v)$ and of the prefix $p'$ which is the concatenation of the sequence $p$ and the one-term sequence $(i)$.

**(3)** The obtained allocation message was special, consisting of a marker *special*, final capital $c''(u)$, and prefix $p$. In this case node $v$ assigns itself the label $(p; 0)$ and acts as in the previous case, with $t = 0$.

This completes the description of Alg. `Wake & Label`$_C$, except for handling "late" count messages. A node $v$ may get count messages from its children after it started sending them allocation messages. In this case incoming count messages are ignored and $v$ does not send any count message to its parent.

The proof of correctness and the performance analysis of Alg. `Wake & Label`$_C$ can be sketched along the following lines. We first observe that Alg. `Wake & Label`$_C$ assigns

distinct labels to all nodes. For any node $v$, define $\hat{d}(v)$ as the maximum degree of all nodes on the path between $v$ and the source (including both ends). Then it can be shown that for any node $v$ and any of its children $w$, the delay between the time when $v$ sends a wakeup message to $w$ and the time when it sends an allocation message to $w$, is at most $c'(v) + \hat{d}(v)$.

For any node $v$, a node $w$ is called *foreign* to $v$, if it is neither $v$, nor an ancestor of $v$, nor a child of an ancestor of $v$, nor a descendant of $v$. Our next observation is that for any node $v$, there exist at least $c'(v) - 1$ nodes foreign to $v$. It follows that the execution time of Alg. `Wake & Label`$_C$, working in an $n$-node tree of maximum degree $d$, is at most $n + d$.

In order to prove that all assigned labels are of size $O(\log n \log d)$, it suffices to show that, for any branch of the tree, the number of nodes in this branch which are in a special situation during the execution of the algorithm, is $O(\log n)$. Take any branch of the tree and let $v_1, \ldots, v_k$, be nodes in a special situation in it, listed top-down. Thus $c'(v_{i+1}) \geq \frac{3}{2}c'(v_i)$ and $c'(v_1) = 1$. Hence $c'(v_k) \geq (\frac{3}{2})^{k-1}$. Hence the number of nodes foreign to $v_k$ is at least $(\frac{3}{2})^{k-1}$. It is also at most $n - 1$. Thus $k$ is $O(\log n)$.

Note that Alg. `Wake & Label`$_C$ is no longer asymptotically optimal. However, the following trivial modification of Alg. `Wake & Label`$_C$ keeps all its worst-case qualities, while enjoying also asymptotic optimality. Alg. `Wake & Label`$_C^+$ behaves like `Wake & Label`$_C$, except in the situation when a node got count messages from all of its children. Then, even if the waiting period is not yet over, the node starts sending allocation messages, as in `Wake & Label`$_A$. All previous features are preserved. On the other hand, the same argument as for Alg. `Wake & Label`$_A$ shows that Alg. `Wake & Label`$_C^+$ works in time at most $3bb(T, s)$, for any tree $T$ and source $s$.

We get the following:

LEMMA 2.1. *Algorithm* `Wake & Label`$_C^+$ *working in an $n$-node tree of maximum degree $d$, assigns distinct labels of size $O(\log n \log d)$ to all nodes of the tree. Its worst-case execution time in such a tree is $\min\{n+d, 3bb(T, s)\}$, and its message and bit complexities are $O(n)$ and $O(n \log n \log d)$, respectively.*

### 2.1.2 *Algorithm* `Wake & Label`

The improvements introduced to Alg. `Wake & Label` aim at decreasing time from $n + d$ to $n + 1$ and label sizes from $O(\log n \log d)$ to $O(\log n)$. The first is achieved by carrying out the processes of waking up and of labeling partly concurrently. The second is achieved by introducing another waiting period for each node, during which the number of children that have not reported is reduced to at most 3. This permits to use only integers 0, 1, 2, instead of $0, 1, \ldots, d$ when forming prefixes of labels. In the full paper [10] we present the full algorithm and establish the following theorem.

THEOREM 2.2. *Given an $n$-node tree $T$ and a source $s$, Algorithm* `Wake & Label` *assigns distinct labels of size $O(\log n)$ to all the nodes. Its time complexity is at most $\min\{n + 1, 3bb(T, s)\}$, hence it is asymptotically optimal. Its message and bit complexities are $O(n)$ and $O(n \log n)$, respectively.*

## 2.2 A lower bound for message-optimal algorithms

While Alg. `Wake & Label` uses $O(n)$ messages, up to 4 messages can travel along every link, and hence the total number of messages can be as much as 4 times larger than the lower bound of $n - 1$. In the full paper we prove the following result, which shows that any labeling protocol that meets this lower bound on the number of messages must in fact produce very large labels.

THEOREM 2.3. *Any labeling protocol using at most $n - 1$ messages must assign labels of $\Omega(n)$ bits to label some $n$-node tree. This property is true even if one restricts the protocol to run on binary trees.*

## 2.3 Optimal time algorithms for trees

In what follows we present an algorithm named `Fastest` working in optimal time $bb(T, s)$ on any $n$-node tree $T$ and source $s$. It uses $O(n)$ messages but assigns labels of size $O(\sqrt{n})$. In Section 2.4 we prove that this is the smallest possible size of labels which can be assigned by optimal time algorithms.

Denote the degree of a vertex $x$ by $d(x)$ and the subtree rooted at $x$ by $T_x$. Given a node $x$, let $t(x)$ denote the round when $x$ gets a message from its parent (setting $t(s) = 0$), and define the *capital* $c(x)$ as the maximum number of rounds that the node $x$ is allowed to wait between time $t(x)$ and starting broadcasting to its children, and still ensure completing broadcasting in $T_x$ by round $bb(T, s)$. (Hence for the source, $c(s) = 0$.) We have

$$c(x) = bb(T, s) - bb(T_x, x) - t(x). \tag{1}$$

Any execution of a broadcast can be thought of as if an adversary chooses the order in which a node $x$ calls its children. Given any such order, we rank the children of each node accordingly, as follows: if $v$ is the $i$th child called by $x$, then the *rank* of $v$ is set to $\phi(v) = d(x) - i$. In particular, the child ranked 0 is called at the latest possible time. We refer to this child as the *latest* child.

For a non-root node $x$, let $p(x)$ denote the path from the source $s$ to $x$ in $T$, not including $s$ itself. (For $s$, let $p(s) = \emptyset$.) We now define the *total rank* of a node $x$, denoted $\Phi(x)$, as the sum of the ranks over $p(x)$, i.e., $\Phi(x) = \sum_{v \in p(x)} \phi(v)$. (Thus for $s$, $\Phi(s) = 0$.)

LEMMA 2.2. *For any node $v$, $c(v) \geq \Phi(v)$.*

PROOF. The claim holds trivially for the source $s$, so it suffices to prove it for a non-root $v$. Consider a non-leaf node $x$. Since the adversary may fix the order in which $x$ calls its children arbitrarily, necessarily

$$bb(T_x, x) \geq bb(T_z, z) + d(x) \tag{2}$$

for any child $z$ of $x$. If $z$ is the $i$th child informed by $x$, then $t(z) = t(x) + i$, and thus by (1) and (2),

$$\begin{aligned} c(z) &= bb(T,s) - bb(T_z,z) - t(z) \\ &\geq bb(T,s) - bb(T_x,x) + d(x) - t(x) - i \;=\; c(x) + \phi(z) \end{aligned}.$$

Hence for a non-root $v$, by repeatedly applying the last inequality from $v$ upwards on the path $p(v)$, it follows that $c(v) \geq c(s) + \sum_{z \in p(v)} \phi(z) = c(s) + \Phi(v)$. Since $c(s) = 0$, the lemma follows. □

### 2.3.1 *A simplified protocol P*

We first describe a simple variant of Algorithm `Fastest`, named protocol $P$, which captures the essential idea and works in optimal time but still assigns potentially long labels. We then show how to refine it and get labels of size $O(\sqrt{n})$.

The labels assigned by protocol $P$ are strings in the language described by the following grammar:

$$S \Longrightarrow \varepsilon \;\;/\;\; \alpha\,\langle rank\rangle\,S \;\;/\;\; \beta\,\langle latest\rangle\,S$$

$$\alpha, \beta \Longrightarrow \text{coding of a positive integer in base 2}$$

where $\varepsilon$ is the empty word and $\langle rank\rangle$ and $\langle latest\rangle$ are two special symbols. The meaning of the labels is the following:

- The label $\alpha\,\langle rank\rangle\,S$ is assigned to the rank-$\alpha$ child of the node labeled $S$.
- The label $\beta\,\langle latest\rangle\,S$ is assigned to the node reached by starting from the node labeled $S$ and continuing downwards to the latest (rank-0) child, for $\beta$ times.

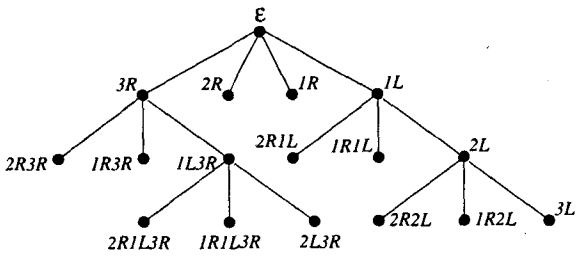


**Figure 4:** Tree labeling using protocol $P$. The symbols "$R$" and "$L$" stand for "$\langle rank\rangle$" and "$\langle latest\rangle$" respectively.

Protocol $P$ operates as follows. When a node $x$ receives message $S$ from its parent, it does the following:

- It assigns $S$ to be its own label.
- If $S = \alpha\,\langle rank\rangle\,S'$ then it sends messages $(d(x)-i)\,\langle rank\rangle\,S$ to the $i$th child it calls, for $i < d(x)$, and sends message $1\,\langle latest\rangle\,S$ to the last child it calls.
- If $S = \beta\,\langle latest\rangle\,S'$ then it sends messages $(d(x)-i)\,\langle rank\rangle\,S$ to the $i$th child it calls, for $i < d(x)$, and sends message $(\beta+1)\,\langle latest\rangle\,S'$ to the last child it calls.

To start the process, the source acts as if it has received a message $S = \varepsilon$, the empty word.

While protocol $P$ requires time $bb(T,s)$ in the worst case and assigns distinct labels to all nodes, the length of the label assigned to node $x$ can be as large as $\Omega(\Phi(x)\log n)$.

### 2.3.2 *Algorithm* `Fastest`

We now refine protocol $P$ to avoid the use of long labels. The modified algorithm, Alg. `Fastest`, is based on the observation that whenever the label assigned to $x$ by $P$ is long, it implies that $\Phi(x)$ and hence $c(x)$ is large (see Lemma 2.2). Thus $x$ is allowed to wait a long time before continuing the label assignment process. This delay can be used to explore some of the subtrees, as done in Alg. `Wake & Label`, and thus reduce the size of used labels.

In Alg. `Fastest` each node sends to its children three types of messages:

- *Allocation* messages of the form $M = ([S:i], \delta)$, consisting of a *location* part $[S:i]$, where $S$ is a string and $i$ is an integer, and a *deadline* part $\delta$ which is an integer.
- *Wakeup* messages.
- *Closure* messages of the form $([S:i],[a,b])$ where $[S:i]$ is as before, and $a, b$ are two integers.

A node $v$ other than the source also sends a *count* message to its parent informing it of the size of the subtree $T_v$, as in Alg. `Wake & Label`.

The algorithm performs the following two phases.

**Phase A**

Upon receiving a wakeup message $x$ starts sending wakeup messages to its children, in the next round. The ranking function described in Protocol $P$ is determined according to this sequence of calls.

Any leaf which got a wakeup message sends back a count message to its parent, and intermediate nodes send count messages bottom-up, as in Alg. `Wake & Label`.

**Phase B**

The source behaves like a node receiving the allocation message $([0:0], 0)$ at time 0.

Upon receiving an allocation message $([S:i], \delta)$, $x$ keeps on performing phase A for $\delta$ rounds. If phase A has not started before, $x$ starts it and performs it for $\delta$ rounds. At the end of this period, all children that have sent count messages to $x$ are called *closed* and all others are called *open*. Then $x$ chooses $([S:i])$ as its own label and immediately proceeds as follows:

**Case 1.** At most one child is open.
In this case $x$ has already called all its children. Now $x$ first sends the message $([S:i+1], d(x)-1)$ to the open child

(if any) and then sends closure messages of the form $([S : i], [a, b])$ to all closed children. The interval $[a, b]$ assigned to each closed child is computed on the basis of the count message received from that child, which reported the size of its subtree, as in Alg. `Wake & Label`.

**Case 2.** More than one child is open.
There are two subcases.

2.1. The node $x$ has already called all its children:

Then $x$ sends allocation messages to all its children in the same order as in phase A.

2.2. The node $x$ has not called all its children yet:

Then $x$ sends allocation messages first to its children which it has not called yet (implicitly serving as a "wakeup" message as well), and then to all other children, in the same order of sending the wakeup messages.

In both subcases, the children of $x$ are ranked in the order of the first messages sent to each of them. If $x$ has $k$ open children $y_1, \ldots, y_k$ ranked $\phi(y_1) > \phi(y_2) > \ldots > \phi(y_k)$, then child $y_j$ is called the *jth open child.*

Messages sent by $x$ during phase B are computed in the following way. For open children, $x$ makes use of $S$ to compute a string $S_j$ to be sent to the $j$th open child in the same way as in protocol $P$. However, protocol `Fastest` requires more information. That is, if $x$ calls an open child $j$ on round $r_j$ of phase B, then instead of sending simply the message $S_j$ as in protocol $P$, $x$ sends $M_j = ([S_j : 0], \delta)$ with $\delta = d(x) - r_j$.

Closed children of $x$ get the appropriate closure messages, computed as in Case 1.

If a node $x$ receives a closure message $([S : i], [a, b])$, then it assigns $([S : i], [a])$ as its own label. Moreover, all children of $x$ are already closed, and thus $x$ sends them the appropriate closure messages.

This completes the description of Alg. `Fastest`.

### 2.3.3 *Analysis of Algorithm* `Fastest`

In order to compute the size of labels assigned by the algorithm, we need the following lemmas.

LEMMA 2.3. *Any node $x$ involved in phase B, has performed phase A for at least $\Phi(x)$ rounds.*

PROOF. Assume that a node $x$ starts phase A (resp. B) in round $t_a$ (resp. $t_b$). Then the child $v$ of $x$ with rank $\phi(v)$ starts phase A in round $t'_a = t_a + d(x) - \phi(v)$. Moreover, it starts phase B in round $t'_b = t_b + d(x)$. It follows that $t'_b - t'_a = t_b - t_a + \phi(v)$. The conclusion now follows by the definition of $\Phi(x)$. □

LEMMA 2.4. *Let $a$ and $n$ be positive reals, let $k$ be a positive integer, and let $x = (x_1, \ldots, x_k)$ be a vector of integers greater than* 1. *Suppose that $\sum_{1 \le i \le k} i x_i \le n$. Then*

$$\sum_{i=1}^{k} (\log x_i + a) \;\le\; 2\sqrt{n} - 1 + a(\sqrt{2n} + 1) \; .$$

THEOREM 2.4. *Given an $n$-node tree $T$ and a source $s$, Algorithm* `Fastest` *assigns distinct labels of size $O(\sqrt{n})$ to all the nodes. Its time complexity is the optimal $bb(T, s)$. Its message and bit complexities are $O(n)$ and $O(n\sqrt{n})$, respectively.*

PROOF. It is easy to see that Alg. `Fastest` assigns distinct labels and works in time $bb(T, s)$. In what follows we bound the size of the labels assigned to the nodes.

Let $L(x)$ denote the label of $x$, and let $|L(x)|$ denote its length. First, we study the maximum length $|S(x)|$ of a prefix appearing in a label of $x$. As prefixes are only modified by nodes receiving an allocation message, we consider only such nodes. Suppose that $x$ receives an allocation message $[S(x) : i]$, and let $q_l(x)$ (resp. $q_r(x)$) be the number of $\langle latest \rangle$ (resp. $\langle rank \rangle$) symbols appearing in $S(x)$. Since there is at least one $\langle rank \rangle$ symbol between two successive $\langle latest \rangle$ symbols, we have $q_l(x) \le q_r(x) + 1$.

Note that $S(x)$ is made of two types of character groups : (i) $\alpha \langle rank \rangle$ which means "I am the $\alpha$th open child," and (ii) $\beta \langle latest \rangle$ which means "I am obtained by following the latest branch for $\beta$ times." Let $S_r(x)$ be the subsequence of $S(x)$ containing groups of type (i) and $S_l(x)$ be the one with groups of type $(ii)$; we have $|S(x)| = |S_r(x)| + |S_l(x)|$, and we will now bound each of the two terms separately.

*Bounding $|S_r(x)|$*

Suppose that $S_r(x) = \alpha_1 \langle rank \rangle \alpha_2 \langle rank \rangle \ldots \alpha_k \langle rank \rangle$ and consider the symbols $\langle rank \rangle$ appearing in $S$ at positions $\{1, 2, \ldots k\}$. These symbols are in a one-to-one correspondence with some vertices $v_1, v_2, \ldots v_k$ of the path from $r$ to $x$.

Since $i - 1$ $\langle rank \rangle$ symbols occur before the symbol associated to any vertex $v_i$, we have $\Phi(v_i) \ge i$. Lemma 2.3 implies that for any $v_i$, at least $i$ rounds elapse between phases $A$ and $B$. Hence, if $v_i$ is an open child, it has at least $i/2$ descendants. Moreover, for every open child $w$ of $v_i$, since $\Phi(w) \ge \Phi(v_i)$, it follows that $w$ has at least $i/2$ descendants as well.

The symbol $\alpha_i$ means that at the end of phase A, $v_i$ had at least $\alpha_i - 1$ open children different from $v_{i+1}$ (otherwise the integer $\alpha_i$ cannot appear). In the special case where $\alpha_i = 1$, $v_i$ had at least one open child different from $v_{i+1}$ (otherwise no character would be added to the prefix $S(x)$ but instead the value of the integer $j$ appearing in the location $[S : j]$ would be increased). In any case, $v_i$ has at least $\alpha_i/2$ open children, different from $v_{i+1}$. By the previous argument, each of these children has at least $i/2$ descendants, so the tree contains at least $i\alpha_i/4$ nodes which do not belong to the subtree $T_{v_{i+1}}$. Globally, we find $k$ pairwise disjoint sets

of cardinalities at least $i\alpha_i/4$. It follows that

$$\sum_{1\le i\le k} i\alpha_i \le 4n.$$

Since $|S_r(x)| \le \sum_{1\le i\le k}(\log \alpha_i + O(1))$, lemma 2.4 implies $|S_r(x)| = O(\sqrt{n})$.

*Bounding* $|S_l(x)|$
In order to bound the length of $S_l(x) = \beta_1 \langle latest\rangle \cdots \beta_k \langle latest\rangle$, note that the symbol $\beta_i$ means that there exist $\beta_i$ successive latest children each one having a total rank at least $i-1$. Similarly to the previous analysis, we get $\sum_{1\le i\le k} i\beta_i \le 2n$. Since $|S_l(x)| \le \sum_{1\le i\le k}(\log \beta_i + O(1))$, using lemma 2.4 once again we get $|S_l(x)| = O(\sqrt{n})$.

*Bounding* $|L(x)|$
In view of the above, $|S(x)| = O(\sqrt{n})$. Since $|L(x)| \le \max|S(x)| + O(\log n)$ for any vertex $x$, it follows that $|L(x)| = O(\sqrt{n})$, proving that all labels are of size $O(\sqrt{n})$.

The same arguments show that all messages are of size $O(\sqrt{n})$. Since at most 3 messages travel on any link, this shows that Alg. `Fastest` has message and bit complexities $O(n)$ and $O(n\sqrt{n})$, respectively. □

## 2.4 A tradeoff for optimal time algorithms on trees

The following lower bound shows that labels assigned by Alg. `Fastest` are of smallest possible order of magnitude, for all algorithms working in optimal time $bb(T,s)$, on any tree $T$ and source $s$.

THEOREM 2.5. *Any labeling protocol working in time at most $bb(T,s)$ on any tree $T$ and source $s$, must assign labels of size $\Omega(\sqrt{n})$ on some $n$-node tree. This property holds even if the protocol is restricted to the class of binary trees.*

PROOF. Fix a labeling protocol $P$. We study the behavior of $P$ on the class of binary trees rooted at $s$. Note that the local state of a node $v$ during the execution of $P$ depends only on the degree of $v$, on whether $v = s$, and on messages previously received by $v$.

Assume that $d = \sqrt{(n-1)/2}$ is integer, and let $B$ be the complete binary tree of depth $d$, rooted at $s$. Let $C$ be the tree obtained from $B$ by replacing every link of $B$ by a simple path of length $d$. Since $C$ has $2^d$ leaves, protocol $P$ must assign a label of size $\Omega(d) = \Omega(\sqrt{n})$ to at least one of them. Let $x$ be such a leaf of $C$.

Let $Z$ be the branch of $C$ joining $s$ and $x$. Let $z_0, z_1, \ldots, z_{d-1}$ be nodes of $Z$ with more than one child, listed top-down, with $z_0 = s$. Let $z_d = x$. Let $Y = \{y_1, \ldots, y_d\}$, where $y_i$ is the other descendant of $z_{i-1}$ at distance $d$ from it, different from $z_i$. Let $T$ be the subtree of $C$ spanned by the nodes of $Z \cup Y$ and the nodes connecting them. $T$ has $n = 2d^2 + 1$ nodes.

Let $\xi_T$ (resp., $\xi_C$) denote the execution of protocol $P$ on the tree $T$ (resp., $C$). The *deviation time* of a node $v$ of $T$, denoted by $\check{t}(v)$, is the time in which the execution $\xi_T$ deviates in $v$ from $\xi_C$, i.e., the first round in which $v$ enters a different local state in the two executions.

Note that the only nodes that start the two executions in different local states, once they are woken up, are the nodes of $Y' = Y \setminus \{y_d\}$, since each $y_i \in Y'$ has different degrees in $T$ and $C$. Since the execution is initiated by the source $s$, node $y_i$ is woken up in round $id$ at the earliest, hence $\check{t}(y_i) \ge id$.

Every other node $v$ starts the two executions with an identical local state, and therefore it can distinguish between them only after receiving a different message from some neighbor. As only the nodes of $Y'$ can originate the transmission of different messages in the executions $\xi_T$ and $\xi_C$, there must exist a path $(u_0, \ldots, u_q)$ in $T$ such that $u_0 \in Y'$, $u_q = v$, and $u_i$ sends a different message to $u_{i+1}$ in round $r_i$ of executions $\xi_T$ and $\xi_C$, where $r_0 < \cdots < r_{q-1}$. Hence denoting the distance between the nodes $v$ and $w$ in $T$ by $dist(v,w)$, we have that for any node $v \notin Y'$,

$$\check{t}(v) \ge \min_{1\le i\le d-1}\{\check{t}(y_i) + dist(y_i, v)\} \ge \min_{1\le i\le d-1}\{id + dist(y_i, v)\}$$

In particular, as $dist(y_i, x) = d(d-i+2)$, we have that

$$\check{t}(x) \ge \min_{1\le i\le d-1}\{id + (d^2 - id + 2d)\} \ge d^2 + 2d\,. \quad (3)$$

Let $t^I_C$ denote the time $x$ is assigned a label in the execution $\xi_C$. Note that $bb(T,s) = bb(C,s) = d^2 + d$, and hence $t^I_C \le d^2 + d$. By Inequality (3), $t^I_C < \check{t}(x)$, hence in round $t^I_C$ the states of $x$ in the executions $\xi_T$ and $\xi_C$ are still identical, and therefore $x$ must get the same $\Omega(\sqrt{n})$-bit label in $\xi_T$ as in $\xi_C$. □

# 3. LABELING ARBITRARY NETWORKS

In this section we study the label assignment problem for arbitrary networks in the mixed model.

Call a link *saturated* if a message has already crossed it. It is easy to see that $bb(G,s)$ equals the worst-case time required for saturating all links. Indeed, since the network is unknown, it is necessary to saturate all links to be sure that all nodes got the message, as an extra node of degree 2 could be "hidden" in the middle of an unsaturated link.

As shown in [11], for any integer $n$ there exists an $n$-node network $G_n$ with source $s_n$ such that $bb(G_n, s_n) = 2n - o(n)$. Hence $2n$ is the asymptotic lower bound on the worst-case execution time of any labeling protocol. In the full paper we describe a variant of Alg. `0-1-Split`, named `0-1-Split`$^G$, with the following properties.

THEOREM 3.1. *Given an $n$-node network and a source $s$, Algorithm* `0-1-Split`$^G$ *assigns distinct labels of length $O(n)$ to all the nodes. Its time complexity is at most $2n$.*

Our next algorithms, named Alg. `Wake & Label`$^G_X$ for $X = A, B$, are simple variants of the corresponding Algorithms `Wake & Label`$_X$. They assign labels from the range $[1, n]$, but are somewhat slower than Alg. `0-1-Split`$^G$. The first

stage of Alg. Wake & Label$_X^G$ involves constructing a spanning tree $T$ (rooted at $s$) for the network $G$, essentially by broadcasting a "wakeup" message throughout the network. In the full paper we show that the tree construction procedure terminates before time $2n-2$. Moreover, it can be shown that by round $2n-1$, each leaf $v$ of the tree $T$ knows that it is a leaf. From here on, the algorithm proceeds as the second and third phases (count and allocation) of Alg. Wake & Label$_X$. In the full paper we prove:

THEOREM 3.2. *Given an $n$-node, $m$-edge network and a source $s$, Algorithm* Wake & Label$_X^G$ *(for $X = A, B$) assigns distinct labels from the range $[1,n]$, and its message and bit complexities are $O(m)$ and $O(m + n\log n)$, respectively. The time complexity of Algorithm* Wake & Label$_B^G$ *is $3n$, and that of Algorithm* Wake & Label$_A^G$ *is $4bb(G,s)+1$ (hence it is asymptotically optimal).*

# 4. THE EXTREME MODELS

This section studies two variants of the mixed model discussed in the previous sections, namely, the *all-port* model and the *one-port* model. These variants are refered to as *extreme* because they either completely relax all the communication constraints, or maximize these constraints.

## 4.1 The all-port model

In this section we discuss label assignment in the all-port model. Clearly, all the algorithms presented for the mixed model work in the all-port model as well. Hence we focus on improved algorithms, which do not work in the mixed model, and on lower bounds (which will naturally apply to the mixed model as well).

Specifically, in the full paper we describe two labeling algorithms for arbitrary networks, with the following properties.

THEOREM 4.1. *Given an $n$-node, $m$-edge network $G$ and a source $s$ in the all-port model, Algorithms* DFS$^+$ *and* All-port 3-Phase *assign distinct labels from the range $[1,n]$ to all the nodes.*

*1. Algorithm* DFS$^+$ *has time and message complexities $2n$ and $O(m)$, respectively,*

*2. Algorithm* All-port 3-Phase *has time complexity $3\cdot ecc(G,s)+1$, hence it is asymptotically optimal.*

*Its message and bit complexities are $O(m)$ and $O(m + n\log n)$, respectively.*

*3. On trees, Algorithm* All-port 3-Phase *has time complexity $3\cdot ecc(G,s)-2$, which is optimal.*

### 4.1.1 Lower bounds for optimal labeling on trees

The optimality of Alg. All-port 3-Phase on trees is based on the following lower bound.

THEOREM 4.2. *Any labeling protocol in the all-port model that assigns labels from the range $[1,n]$ requires at least $3\cdot ecc(T,s)-2$ rounds on some tree $T$ and source $s$.*

PROOF. A $(k,k_1,k_2)$-*broom** $B$ is the graph of size $n = 2k+k_1+k_2+1$ consisting of a path of $2k+1$ nodes with extremities $v_L, v_R$, and $k_1$ (resp. $k_2$) additional nodes connected to $v_L$ (resp. $v_R$). Assume the source $s$ is the central node of the path. Then $ecc(B,s) = k+1$.

Assume for the purpose of contradiction that there exists a protocol that assigns labels from the range $[1,n]$ on every tree, and completes labeling the $(k,\hat{k},\hat{k})$-broom $B$, where $\hat{k} = 2k+2$, in $t < 3\cdot ecc(B,s) - 2 = 3k+1$ rounds.

Let $\mathcal{B}(k,\hat{k})$ denote the family of all $(k,\hat{k},k')$-brooms for any $k' \geq 0$. Consider the last round of the protocol's execution on $B$. At that time, any leaf $x$ of $v_L$ has not received yet any information from the node $v_R$, since $v_R$ is at distance $k$ from $s$, hence the earliest round in which it wakes up is $k$, and the distance from $v_R$ to $x$ is $2k+1$. Hence at time $t$, the local information maintained at the node $x$ is the same for all brooms in $\mathcal{B}(k,\hat{k})$. It follows that the label $\ell$ assigned to $x$ by the protocol is at most $2k+1+\hat{k} = 4k+3$, since otherwise, $\ell$ would be larger than the number of nodes of the $(k,\hat{k},0)$-broom, causing the protocol to fail on it.

A symmetric argument can be applied to every leaf $y$ of $v_R$, yielding that at time $t$, node $y$ cannot have receive a label larger than $4k+3$.

But $B$ has $4k+4$ leaves, and by the above argument, each of them must be assigned a distinct label from the range $[1,4k+3]$; contradiction. Hence a protocol assigning labels in the range $[1,n]$ requires at least $3\cdot ecc(B,s)-2$ rounds on the $(k,\hat{k},\hat{k})$-broom $B$. □

If the message complexity is exactly $m$ (the number of edges), then we already know (by Theorem 2.3) that labels of size $\Omega(n)$ are required in some trees. In the all-port model, one can relax the message complexity hypothesis, and show that assuming the time to be $ecc(T,s)$ is enough to force some labels to be of size $\Omega(n)$.

THEOREM 4.3. *Any labeling protocol in the all-port model that works in $ecc(T,s)$ rounds must assign labels of size $\Omega(n)$ on some tree $T$ and source $s$.*

PROOF. The proof goes along similar lines to that of Thm. 2.3. Consider the class $\mathcal{T}$ of binary trees $T$ with a source $s$ of eccentricity $d$ in which every non-leaf node $v$ has exactly two children. Given a tree $T \in \mathcal{T}$, an *extremal leaf* of $T$ is any leaf $v$ at distance $d$ from the source $s$. Associate with every extremal leaf $v$ of $T$ a $d$-bit word $w(v,T) = w_1(v,T)w_2(v,T)\ldots w_d(v,T)$, $w_i(v,T) \in$ {left,right}, such that the path from $s$ to $v$ in $T$ is obtained by starting from $s$ and going successively left or right according to the $w_i(v,T)$'s.

Consider a protocol $P$ running in $d$ rounds on any tree $T \in \mathcal{T}$. For any node $v$ of $T$, let $L(v,T)$ be the label assigned to $v$ by $P$. Such a protocol satisfies the property that for

*or really a "dual-purpose" broom, for concurrently sweeping the floor and the ceiling

every extremal leaf $v$, the label $L(v,T)$ is assigned to $v$ at time $d$, and moreover, that label is a function $L(v,T) = f(M, w(v,T))$ where $M$ is the message sent by $s$ to its child on the path to $v$ in round 1. This is because each node along the path from $s$ to $v$ had to send a message down towards $v$ in the very first round after it was woken up by $s$, and therefore it could not rely on any other information.

Let $B$ be the complete binary tree of eccentricity $d$ rooted at $s$. Since $B$ has $2^d$ leaves, protocol $P$ must assign a label of size $\Omega(d)$ to at least one of them. Let $x$ be such a leaf of $B$, and let $w = w(x, B)$.

Let $Z = \{z_0, z_1, \ldots, z_d\}$ be the branch of $B$ where $z_0 = s$ and $z_d = x$. Let $Y = \{y_1, \ldots, y_d\}$, where $y_i$ is the other child of $z_{i-1}$ in $B$, distinct from $z_i$. Let $T_0$ be the subtree of $B$ spanned by the nodes of $Z \cup Y$, with the same "left/right" orientation for the edges, namely, such that $w(x, T_0) = w(x, B)$.

Also, $B$ and $T_0$ look locally the same to $s$, hence in round 1 of the executions of protocol $P$ on $T$ and $B$, $s$ sends to $z_1$ the same message $M$.

It follows that labels assigned to $x$ in both cases are equal, i.e.,

$$L(x, T_0) = f(M, w(x, T_0)) = f(M, w(x, B)) = L(x, B).$$

As $T_0$ has $n = 2d + 1$ nodes, we conclude that the leaf $x$ of $T_0$ gets a label of size $\Omega(d) = \Omega(n)$. □

## 4.2 The one-port model

It turns out that an appropriate adaptation of Algorithm `Wake & Label`$_A$ can be made to work for trees in the one-port model as well, in asymptotically optimal time.

THEOREM 4.4. *Given an $n$-node tree $T$ and a source $s$ in the one-port model, Algorithm* `Time-Slots` *assigns distinct labels from the range $[1, n]$ to all the nodes. Its time complexity is at most $3\,bb(T, s)$, hence it is asymptotically optimal. Its message and bit complexities are $O(n)$ and $O(n \log n)$, respectively.*

For arbitrary networks, the straightforward DFS algorithm working in $2m$ rounds has asymptotically optimal time and message complexity, as implied by the following natural lower bound for the one-port model. (The proof is deferred to the full paper [10].)

THEOREM 4.5. *Any labeling algorithm working for arbitrary networks in the one-port model must take at least $m$ rounds on every $m$-edge network.*

# 5. REFERENCES


[1] S. Albers and M. R. Henzinger, Exploring unknown environments, *Proc. 29th Symp. on Theory of Computing*, 1997, 416-425.

[2] D. Angluin, Local and Global Properties in networks of processors, *Proc. 12th Symp. on Theory of Computing*, 1980, 82-93.

[3] H. Attiya, M. Snir and M. Warmuth, Computing on an Anonymous Ring, *Journal of the ACM* 35, (1988), 845-875.

[4] H. Attiya and M. Snir, Better Computing on the Anonymous Ring, *Journal of Algorithms* 12, (1991), 204-238.

[5] B. Awerbuch, A new distributed depth-first-search algorithm, *Information Processing Letters* 20, (1985), 147-150.

[6] B. Awerbuch, O. Goldreich, D. Peleg and R. Vainish, A Tradeoff Between Information and Communication in Broadcast Protocols, *Journal of the ACM* 37, (1990), 238-256.

[7] P. Boldi and S. Vigna, Computing anonymously with arbitrary knowledge, *Proc. 18th ACM Symp. on Principles of Distributed Computing*, 1999.

[8] X. Deng and C. H. Papadimitriou, Exploring an unknown graph, *Proc. 31st Symp. on Foundations of Computer Science*, 1990, 356-361.

[9] K. Diks, E. Kranakis A. Malinowski and A. Pelc, Anonymous wireless rings, *Theoretical Computer Science* 145 (1995), 95-109.

[10] Pierre Fraigniaud, Andrzej Pelc, David Peleg and Stéphane Pérennes, Assigning labels in unknown anonymous networks, Technical Report MCS00-01, the Weizmann Institute of Science, 2000.

[11] L. Gargano, A. Pelc, S. Pérennes and U. Vaccaro, *Efficient communication in unknown networks*, Technical Report RR-3609, January 1999, INRIA, France.

[12] E. Kranakis, Symmetry and Computability in Anonymous Networks: A Brief Survey, *Proc. 3rd Int. Conf. on Structural Information and Communication Complexity*, 1997, 1-16.

[13] E. Kranakis, D. Krizanc and J. van der Berg, Computing Boolean Functions on Anonymous Networks, *Information and Computation* 114, (1994), 214-236.

[14] P. Panaite and A. Pelc, Exploring unknown undirected graphs, *Proc. 9th ACM-SIAM Symposium on Discrete Algorithms*, 1998, 316-322.

[15] N. Sakamoto, Comparison of Initial Conditions for Distributed Algorithms on Anonymous Networks, *Proc. 18th ACM Symp. on Principles of Distributed Computing*, 1999.

[16] M. Yamashita and T. Kameda, Computing on anonymous networks, *Proc. 7th ACM Symp. on Principles of Distributed Computing*, 1988, 117–130.