Beeping a Deterministic Time-Optimal Leader Election

Fabien Dufoulon
LRI, Université Paris-Sud, CNRS, Université Paris-Saclay, Orsay, France
dufoulon@lri.fr
https://orcid.org/0000-0003-2977-4109

Janna Burman
LRI, Université Paris-Sud, CNRS, Université Paris-Saclay, Orsay, France
burman@lri.fr

Joffroy Beauquier
LRI, Université Paris-Sud, CNRS, Université Paris-Saclay, Orsay, France
beauquier@lri.fr

Abstract

The *beeping model* is an extremely restrictive broadcast communication model that relies only on carrier sensing. In this model, we solve the *leader election* problem with an asymptotically optimal round complexity of $O(D+\log n)$, for a network of unknown size n and unknown diameter D (but with unique identifiers). Contrary to the best previously known algorithms in the same setting, the proposed one is deterministic. The techniques we introduce give a new insight as to how local constraints on the exchangeable messages can result in efficient algorithms, when dealing with the beeping model.

Using this deterministic leader election algorithm, we obtain a randomized leader election algorithm for anonymous networks with an asymptotically optimal round complexity of $O(D + \log n)$ w.h.p. In previous works this complexity was obtained in expectation only.

Moreover, using deterministic leader election, we obtain efficient algorithms for symmetry-breaking and communication procedures: $O(\log n)$ time MIS and *5-coloring* for tree networks (which is time-optimal), as well as k-source *multi-broadcast* for general graphs in $O(min\{k, \log n\} \cdot D + k \log \frac{nM}{k})$ rounds (for messages in $\{1, \ldots, M\}$). This latter result improves on previous solutions when the number of sources k is sublogarithmic ($k = o(\log n)$).

2012 ACM Subject Classification Theory of computation → Distributed computing models, Theory of computation → Distributed algorithms, Theory of computation → Design and analysis of algorithms

Keywords and phrases distributed algorithms, leader election, beeping model, time complexity, deterministic algorithms, wireless networks

Digital Object Identifier 10.4230/LIPIcs.DISC.2018.20

Related Version A full version is available at `https://hal.archives-ouvertes.fr/hal-01794711`.

1 Introduction

The leader election (LE) problem, where a single (leader) node is given a distinguished role in the network, is a fundamental building block in algorithm design. This is because a leader can initiate and coordinate behaviors in the network, often making leader election a crucial first step in applications requiring communication and agreement on a global scale. For example, more advanced communication primitives such as broadcast, gossiping and multi-broadcast, rely on a leader to coordinate transmissions [10] (see also Sect. 4.3).

Wireless networks with severe restrictions on communication capabilities are an increasingly prevalent subject of study, e.g., [6, 20, 7, 13, 14, 8]. In order to model these networks, Cornejo and Kuhn [7] introduced a convenient formal framework: the *discrete beeping model* ($\mathcal{BEEP}$). In this model, time is divided into synchronous rounds, and in each round, a node can either listen or transmit a unary signal (beep) to all its neighbors. The possibility to directly transmit a beep to a node is defined by a *static communication graph*, and nodes have no knowledge of this graph. As a beep is merely a detectable burst of energy, a listening node does not receive the *identifiers* (ids) of its beeping neighbors. Even more critically, a beeping node receives no feedback, while a silent (listening) one can only detect that either at least one of its neighbors beeped or that all of them were silent. Although algorithms can take advantage of the synchronous nature of the rounds to transmit information using beeps, doing so impacts the time complexity in a quantifiable manner. This work studies how this impact can be minimized.
The beeping model has also been justified by its possible applications to biological networks [19], which are reliant on primitive communications. Fireflies communicate through flashes of light [2, 15] and cells through the diffusion of specific chemical markers [1, 22].

Beeps are an extremely limited form of communication, making it difficult to coordinate nodes. Being a fundamental coordination problem, leader election has received a lot of attention (see Sect. 1.1). Probabilistic and deterministic solutions were proposed for general graphs, and a time complexity lower bound of $\Omega(D+\log n)$ was established (D is the diameter of the network, and n its size). A prime concern is the design of time-efficient *uniform* solutions, that is, not requiring any knowledge on the graph topology or on the parameters n and D (or even on their upper bounds). Indeed, it is unrealistic to assume that upper bounds on these parameters are always available, especially when considering dynamic networks.
Amongst existing works on LE in $\mathcal{BEEP}$, the more difficult (for design) deterministic case has received less attention. However, this case is useful whenever random behavior is inappropriate or deterministic guarantees are required. We show in this work that an asymptotically time-optimal deterministic algorithm can be designed. This algorithm gives rise to an anonymous (not using ids) randomized algorithm that also matches the lower bound.

1.1 Related Work

Leader election (LE), being a fundamental problem in distributed computing, has been studied in various models. In each newly introduced model, an *efficient* leader election algorithm is a foremost concern, since it is frequently used as a building block in more complex algorithms. In particular, recent models designed for wireless networks assume that simultaneous communications interfere with each other. Consequently, leader coordination is even more important in these models, though LE is harder to solve efficiently.
Even though computational complexities (in particular time complexity) for LE are key aspects in the algorithmic design, additional properties are also of concern: for example, one might want nodes to detect termination, or to ensure that there is never more than one leader node during any execution (*safety property*).

Ghaffari and Haeupler [13] present the first LE algorithm for $\mathcal{BEEP}$, which elects a leader in $O(D+\log n)\cdot O(\log^2\log n)$ rounds with high probability (w.h.p.: with probability $1-n^{-\theta(1)}$). [13] also gives a lower bound of $\Omega(D+\log n)$ rounds for LE, applicable both to

Table 1 LE algorithms in the beeping model.

Reference	Round complexity	Safety	Knowledge
[13]	$O(D + \log n \log\log n) \cdot min\{\log\log n, \log \frac{n}{D}\}$ w.h.p.	w.h.p.	$N = n^c$
[12]	$O(D \cdot \log n)$ deterministic time	Deterministic	None
[9]	$O(D + \log n)$ expected time	w.h.p.	$N = n^c$
Here	$O(D + \log n)$ deterministic time	Deterministic	None
Here	$O(D + \log n)$ w.h.p.	w.h.p.	$N = n^c$

deterministic and randomized (w.h.p. time) algorithms. This bound can be compared to the $\Omega(D)$ lower bound in the $\mathcal{ECONGEST}$ model [16]. $\mathcal{ECONGEST}$ differs from $\mathcal{BEEP}$ in that any given node can send (different) messages of $O(\log n)$ bits to each of its neighbors during a round. When nodes receive messages, there are no collisions and they can distinguish from which edge they received a particular message. Intuitively, since a beep can convey at most one bit, additional $\Omega(\log n)$ rounds are necessary [18, 5, 11]. Following the result from [13], Czumaj and Davies [8, 9] presented a randomized LE algorithm with $O(D + \log n)$ expected time in $\mathcal{BEEP}$. In both randomized algorithms, the safety property is guaranteed w.h.p., but some upper bound N on the number of nodes n is required. As for deterministic LE, Förster et al. [12] give the first algorithm in $\mathcal{BEEP}$, with an $O(D \cdot \log n)$ round complexity. This algorithm is uniform in both n and D. The round complexities of different LE algorithms, including those presented in this work, are compared below (see Table 1).

It is mentioned in [13, 9] that upper bounds in $\mathcal{BEEP}$ apply to the well-known radio networks with collision detection ($\mathcal{RN}$-$\mathcal{CD}$). In $\mathcal{RN}$-$\mathcal{CD}$, nodes can send messages of $O(\log n)$ bits (instead of beeps) and listening nodes receive a "collision" message if more than two neighbors communicate at the same time. For both models, previous results are not tight, especially for deterministic leader election.

[13, 8, 9, 12] concentrate on improving the time complexity of LE in general graphs, in $\mathcal{BEEP}$. A different focus is presented in [14], where the goal is to minimize the size of the state machine representation of an algorithm solving randomized LE in single-hop networks.

Amongst the extensive leader election literature in other models, Casteigts et al. [5] is particularly relevant to this work. [5] proposes an $O(D + \log n)$ time deterministic LE algorithm in the constant-size $\mathcal{ECONGEST}$ model, where the algorithm is uniform in both the number of nodes n and the diameter D. This model is much stronger than $\mathcal{BEEP}$, in that a node can easily learn its local topology and has direct links to communicate with its neighbors, whereas the absence of such links in the beeping model causes interference and makes directed messages (with known sender and receiver) unachievable or plainly inefficient. Notice that by using a 2-hop coloring and by separating in time the transmission of messages, according to the colors of both the sender and receiver, the constant-size $\mathcal{ECONGEST}$ model can be simulated, but with a prohibitive multiplicative factor of $O(\Delta^4)$ [3] (where Δ is the maximum degree).
Nevertheless, one of the main contributions of [5] is a rooted (in the maximum id node) spanning tree construction and an *information diffusion* algorithm, designed to spread the maximum identifier efficiently, in a pipeline-like manner (rather than performing consecutive local comparisons on complete identifiers). This latter shift is crucial to the time-optimality of their algorithm, and is used here to improve on the $O(D \cdot \log n)$ result from [12].

1.2 Contributions

We propose a deterministic and completely uniform (in n and D) leader election algorithm with an $O(D + \log n)$ asymptotically optimal round complexity. By independently sampling $\theta(\log n)$ bits to create unique identifiers w.h.p. and using this algorithm, we obtain a uniform (in D only) randomized leader election algorithm which takes $O(D + \log n)$ rounds w.h.p. and works in anonymous networks. Both solutions are the first to achieve time-optimality for these guarantees in both $\mathcal{BEEP}$ and $\mathcal{RN}$-$\mathcal{CD}$, outperforming all previous deterministic and randomized results. This work closes the gap between upper and lower bounds for LE.

Furthermore, using the proposed deterministic LE algorithm, we propose the first asymptotically time-optimal (in $O(\log n)$ rounds) Maximal Independent Set (MIS) and 5-coloring algorithms for trees in $\mathcal{BEEP}$ (leveraging the fact that given a leader in a tree network, it is simple to compute a 2-coloring). The MIS and coloring algorithms can be considered as essential symmetry-breaking procedures, and designing optimal-time solutions (even limited to tree networks) might be crucial for other applications in $\mathcal{BEEP}$.
Then, we give an $O(min\{k, \log n\} \cdot D + k \log \frac{nM}{k})$ time k-source multi-broadcast (with provenance) algorithm (for messages in $\{1, \ldots, M\}$). This latter algorithm improves a previous result by Czumaj and Davies [8], when the number of sources k is sublogarithmic ($k = o(\log n)$), by executing k consecutive leader elections. Communication primitives are especially important in $\mathcal{BEEP}$, as they allow to deal with the interferences caused by simultaneous communications, and thus to design complex algorithms.

2 Model and Definitions

2.1 Preliminaries

The *communication network* is represented by a simple connected undirected graph $G = (V, E)$, where V is the node set and E the edge set. The *network size* $|V|$ is also denoted by n, and the *diameter* by D. Nodes have unique identifiers (ids). This property is essential in order to break symmetry in deterministic algorithms. The *identifier* of a node $u \in V$, $id(u)$, is an integer from $\{1, \ldots, U\}$ where U is some upper bound unknown to nodes. Then, the *maximum length* over all identifiers in G is $O(\log U)$ (also unknown). For simplicity, we make the common assumption that identifiers have logarithmic (in n) length, i.e., the id space is $\{1, \ldots, N\}$ where $N = n^c$ for some unknown constant $c > 1$. In Sect. 3.3, we explain how the results of the paper apply to an arbitrary id space setting.

Now, we give definitions pertaining to (binary) words. The empty word is denoted by ϵ. The operator $\|$ is for the *word concatenation*. The *length* of a word x is denoted by $|x|$. For any word x and integer $j \in \{1, \ldots, |x|\}$, $x[j]$ denotes the j^{th} *most significant bit* of x. Let x and y be two words (of possibly different lengths), x is said to be the *prefix* (resp., *proper prefix*) of y if there exists a word (resp., non empty word) z such that $x \| z = y$. Moreover, x is said to be *higher* than y, denoted by $x \succ y$, if y is a proper prefix of x, or if $x[j] > y[j]$ for the first differing bit j (even if $|x| < |y|$).

The α-encoding [5] of an integer $i \in \mathbb{N}^{>0}$ is a word obtained from the binary representation *bin* of i. By definition, $\alpha(i) = 1^{|bin|} \| 0 \| bin$. In the proposed LE algorithm (Sect. 3), instead of ids, nodes compare their α-encodings (α-ids). Finding the highest α-id is equivalent to finding the maximum id, and can be performed uniformly (without padding the binary representations of ids) using bit-wise comparisons. A word x is *well-formed* if there exists an integer i such that $x = \alpha(i)$. It is simple to prove that for every word x, there is at most one such integer i. Thus the α^{-1} function (α's “inverse”) is defined on well-formed words.

2.2 Model Definitions

In the *beeping model* ($\mathcal{BEEP}$), an execution proceeds in synchronous rounds, i.e., there are synchronized local clocks and all nodes start at the same time, in a *synchronous start*. In each round, nodes synchronously execute the following steps. First, each node beeps (instruction $BEEP$ in algorithms) or listens ($LISTEN$ in algorithms). Beeps are transmitted to all neighbors of the beeping node. Then, if a node beeped (in the previous step of the same round), it learns no information from its neighbors. Otherwise, it knows whether or not at least one of its neighbors beeped (during the previous step of the same round). Finally, each node performs local computations. The synchronous start assumption can be replaced by a slightly weaker variant called *wake-on-beep* [1], for a constant multiplicative overhead (and an additive factor of $O(D)$ rounds). In this variant, a node starts spontaneously either at an arbitrary time, or at one of its neighbors' beep, whichever happens first. Upon waking up, a node beeps immediately, to wake up its neighbors. As a result, the local clocks of two neighboring nodes differ by at most 1. Therefore, nodes can use phases of 3 rounds [1] (in which the node can beep or listen in the central round, and listens in both other rounds) to simulate rounds of a synchronous start execution.

We adopt the usual definitions for the system/algorithm: *state* of a node (values of its variables), *configuration* (a vector of all the nodes' states), *execution* (a sequence of configurations at consecutive rounds' ends), *terminal configuration* (a configuration repeated indefinitely), *termination* (when a terminal configuration has been reached), *round complexity* (number of rounds needed until a terminal configuration satisfying the problem conditions is reached, in the worst case). A variable var of a node v is explicitly associated to v using a subscript var_v. An algorithm is said to be *uniform*[1] in a parameter p if the algorithm is not given p (and is unable to infer it from the information it receives). For example, in a uniform (in n) algorithm, nodes do not know the size n of the network, neither can they deduce it from their identifier. Notice that the variable size of the identifiers gives an unusable upper bound on the network size, as it leads to an excessive and unrealistic time complexity.

2.3 Leader Election

In the *leader election* (LE) problem, each node has a boolean variable, indicating a *leader* or a *non-leader* state. During an execution, there is never more than one leader (*safety* property). Initially, all nodes are non-leaders. Every execution terminates, and at the termination there is exactly one leader.
Now we give auxiliary definitions. First, we define *eventual leader election*, where the algorithm terminates but no node can detect this. Then, we define *terminating leader election*, where the algorithm terminates and all nodes detect when there remains a single candidate node (the leader). We solve *explicit* leader election (when nodes have unique identifiers): a terminating leader election in which all nodes know the elected leader's identifier at the termination.

[1] It is known that termination detection is easy in a synchronous setting whenever particular parameters related to the size of the communication graph are known, i.e., non-uniform terminating algorithms are easier to construct than the uniform ones.

Algorithm 1 Uniform Eventual Leader Election Algorithm.

```
1: IN: $id$: identifier ; OUT: $leader$: boolean, $leaderId$: identifier
2: $candidate := true$, $prefix := \epsilon$, $suspicious := false$        ▷ $\epsilon$ is the empty word
3: $leaderId := 0$, $leader := false$        ▷ $id$ and $leaderId$ are ids, from $\{1, \ldots, N\}$
4: for diffusion phase $p := 1$ ; $p{+}{+}$ do
5:     // First, a communication phase with $c$ rounds.
6:     Communicate $(prefix, suspicious)$ to all neighboring nodes.
7:     // Then, apply predicates of rules 1 to 5 on received $(prefix, suspicious)$ pairs.
8:     Use received $(prefix, suspicious)$ pairs to update $prefix$, $candidate$ and $suspicious$
9:     if not $candidate$ then $leader := false$
10:    else if $prefix = \alpha(id)$ then $leader := true$
11:    if $prefix$ is well-formed then $leaderId := \alpha^{-1}(prefix)$
```

3 Leader Election Algorithms

Classical approaches used to solve leader election in $\mathcal{CONGEST}$ models do not directly apply to $\mathcal{BEEP}$. Although they can be adapted using a transformer, doing so is too costly in most communication graph topologies (see discussion in the related work section: Sect. 1.1).
To solve the strongest version of LE, the explicit leader election, we proceed in two main steps. First, we design a uniform algorithm for eventual leader election, in Sect. 3.1. Then, in Sect. 3.2, we combine this algorithm with a specially designed uniform termination detection component to obtain a uniform explicit leader election algorithm. Finally, in Sect. 3.3, we discuss how the presented algorithm can be applied to other settings (e.g., arbitrary id range).

3.1 Uniform Eventual Leader Election

The algorithm (Algorithm 1) is described first (Sect. 3.1.1). Then, in Sect. 3.1.2, k-balanced messages are presented. They are used to allow constant-size *communication phases* composed of rounds and dedicated to the communication of (large) messages respecting local constraints. Using the k-balanced message technique, a detailed description of the communication phases (appearing in Algorithm 1) is given in Sect. 3.1.3. Finally, in Sect. 3.1.4, we relate the presented techniques to existing works in $\mathcal{CONGEST}$ models.

3.1.1 Description

All nodes aim to spread their α-identifiers ($\alpha(id)$ in Alg. 1) to the whole network (*information diffusion algorithm*). They execute loosely synchronized bit-wise comparisons and propagate the bits of the highest detected prefix (of α-id). All nodes start out as *candidates*, with two variables: $prefix$ and *suspicious*. The binary word $prefix$ is initialized to the empty word ϵ and represents the prefix of an α-id. Most of the time, it represents the highest prefix of which the node is aware. Each node adapts its $prefix$ by adding or removing the less significant bits, depending on the information gathered. The boolean *suspicious* is initialized to $false$ and indicates whether the node removed bits from $prefix$ in the last phase.

Nodes execute *diffusion phases* (of c rounds each) synchronously. A diffusion phase consists of one *communication phase* of $c = O(1)$ rounds (line 6), used to send $prefix$ and *suspicious* to all neighbors, followed by a (limited) modification of $prefix$.

The communication phase is described in detail in Sect. 3.1.3. In the same phase, each node receives $(prefix, suspicious)$ pairs from its neighbors, but does not know which node sent which message, nor how many nodes sent any of these messages (*multiplicity*).
After the communication phase, any node v checks if $prefix_v$ is a *locally higher prefix*, using the received pairs (see details below) and the previously gathered information. If this is the case, it appends a bit from its α-id to $prefix_v$ (if $prefix_v$ is a proper prefix of $\alpha(id_v)$), or does nothing (if $prefix_v = \alpha(id_v)$). Otherwise, it modifies $prefix_v$ depending on the highest detected $prefix$ value, and becomes a *follower*. It can no longer become a leader. If that modification removes bits from $prefix_v$, node v is said to be $suspicious$ for the following phase, and $suspicious_v$ is assigned to $true$ for one phase.

The five rules below associate conditions (predicates) to actions. A predicate evaluated to $true$ triggers the associated action. In line 8, these predicates are evaluated (by some node v) on the set of the received $(prefix, suspicious)$ pairs, in the given order of priority, and the first triggered action is performed.

1. If there exists a suspicious neighbor u, such that $prefix_u$ is a proper prefix of $prefix_v$, remove $min\{|prefix_v| - |prefix_u|, 3\}$ letters from the end of $prefix_v$.
2. If $prefix_v = (z \parallel 0 \parallel w)$ with $w \neq \epsilon$ and there exists a neighbor u with $prefix_u = (z \parallel 1 \parallel y)$, delete $|w|$ letters from the end of $prefix_v$.
3. If $prefix_v = (z \parallel 0)$ and there exists a neighbor u with $prefix_u = (z \parallel 1 \parallel y)$, then change $prefix_v$ to $(z \parallel 1)$.
4. If there exists a neighbor u with $prefix_u = (prefix_v \parallel 1 \parallel w)$ then append 1 to $prefix_v$.
5. If there exists a neighbor u with $prefix_u = (prefix_v \parallel 0 \parallel w)$ then append 0 to $prefix_v$.

If any of the predicates (of the rules 1-5) is true, $prefix_v$ is not a locally higher prefix. Indeed, if a neighbor u (of v) is suspicious and $prefix_u$ is a proper prefix of $prefix_v$, then a neighbor of u has a higher prefix than $prefix_v$, or is changing its $prefix$ according to rule 1 above. By deleting the last bits of $prefix_v$, node v is matching $prefix_v$ to an unknown but higher $prefix$. In all 4 other cases, $prefix_u$ is clearly a higher prefix than $prefix_v$, therefore $prefix_v$ modifies (a limited amount of) its last bits to more closely match $prefix_u$.
Additional local computations in lines 9-11 conclude a diffusion phase. Once a candidate's $prefix$ variable is well-formed (i.e., once $id_v = \alpha^{-1}(prefix_v)$), this node becomes a leader. If in later rounds it becomes a follower, then it withdraws from the leader role. Although this process violates the safety property, it is necessary in order to elect a leader, as the last remaining candidate cannot detect that it is the last, due to the lack of termination detection in this preliminary eventual LE version.

The 5 rules described above are an idea adopted from [5]. Thus the described information diffusion process satisfies Lemma 1 and Theorem 2 below, adopted from the results of [5] and adapted here to our beeping algorithm (see Sect. 3.1.4 for more details).

▶ **Lemma 1** (Beeping version of Lemma 8 in [5]). *Let u and v be two neighboring nodes. Then, $prefix_u$ and $prefix_v$ are identical, except in at most 6 (least significant) bits: without loss of generality, from the $|prefix_u|^{th}$ bit (possibly included) to the $|prefix_v|^{th}$ bit.*
Note that if the $|prefix_u|^{th}$ bit differs in $prefix_u$ and $prefix_v$, then $||prefix_u| - |prefix_v|| < 6$

▶ **Theorem 2** (Beeping version of Theorem 10 in [5]). *Let X be the maximum identifier. After $|\alpha(X)| + 6r$ phases of the information diffusion algorithm, all nodes within distance r (for any $r \geq 0$) from the node with id X have $prefix = \alpha(X)$. Thus, after at most $|\alpha(X)| + 6D$ phases, for each node v, $prefix_v = \alpha(X)$, and there is a unique candidate node.*

Proof. Let l be the maximum id node. We prove the theorem by induction on r.
Node l has the maximum identifier X, thus it appends a bit from $\alpha(X)$ in each diffusion phase. After $|\alpha(X)|$ phases, $prefix_l = \alpha(X)$. This concludes the case when $r = 0$.
For the induction step ($r > 0$), consider any given node u at distance $r + 1$ of node l, and one of its neighbors v at distance r from l. By Lemma 1, $prefix_u$ and $prefix_v$ differ in less than 6 bits. After $|\alpha(X)| + 6r$ phases, since $prefix_v = \alpha(X)$ (induction hypothesis), node v does not modify $prefix_v$ and node u necessarily corrects (removes, changes or adds) at least one of $prefix_u$'s bits in each of the 6 following phases, until $prefix_u = \alpha(X)$. ◀

Recall that a communication phase is composed of $c = O(1)$ rounds (c is defined in Sect. 3.1.3). This implies the following theorem.

▶ **Theorem 3.** *Uniform Eventual Leader Election is solved by Algorithm 1 in $O(D + \log n)$ rounds (in the beeping model).*

Proof. Let v be any given node and X the maximum identifier in the network. From Theorem 2, $prefix_v = \alpha(X)$ after $O(D + \log n)$ phases. Nodes have the leader's identifier by applying the α^{-1} function. Moreover, the maximum id node is well-formed after $|\alpha(X)| = O(\log n)$ phases, thus after $O(\log n)$ rounds. As a result, the maximum id node is, and remains, a leader henceforth. ◀

3.1.2 Balanced messages

A basic design technique called multi-slot design pattern [4], allows to communicate constant-size messages without the sender's id nor multiplicity, given a synchronous start. It works in communication phases of M rounds, if at most M different messages (in $\{1, \ldots, M\}$) are allowed. Beeping in the j^{th} round of a phase is equivalent to sending the message j. However, receivers cannot detect which (and how many) nodes sent that message. Thus, due to the beeping model's restrictions, if a node sends a message m, it receives no information about whether any of its neighbors also did.

Clearly, this design technique cannot be used to directly send $prefix$ values, as these values are in $\{1, \ldots, N\}$, and communication phases would be $O(N)$ rounds long. But this technique can be adapted to send the values of a locally constrained (k-balanced) variable. A variable *var* is said to be k-balanced if it satisfies the *k-balancing property*, that is, if the difference between neighboring *var* values is at most k (for every node v and neighboring node u, $|var_u - var_v| \leq k$).

If one wishes to communicate k-balanced messages, then it is enough to transmit, for a message m, the *remainder* $r = m \; mod(1 + 2k)$, using the previous technique, with phases of $M = 1 + 2k$ rounds (where k is known a priori to all nodes). Then, the receiver knowing at the same time its own remainder, the sender's remainder and the fact that the messages are k-balanced, can deduce the originally sent message (but does not know if multiple nodes have sent this message). Specifically, let v be the receiver and u the sender. Node v deduces the original message m_u from the received remainder message r_u: $m_u = m_v + r_u - r_v - \lfloor \frac{r_u - r_v}{k+1} \rfloor M$.

Consider the example depicted in Table 2 for $k = 4$. For a given node v, any message m_u sent by a neighboring node u is in $\{m_v - k, \ldots, m_v + k\}$. By transmitting the remainder $r_u = m_u \; mod(1 + 2k)$, node u indicates whether its message m_u is in the next 4 values or in the previous 4, respectively to m_v, and the exact position amongst the 4 possibilities (more precisely, through $r_u - r_v$). The remaining $-\lfloor \frac{r_u - r_v}{k+1} \rfloor M$ factor deals with the fact that some lower (than m_v) messages m_u result in a high remainder r_u, and some higher messages m_u in a low remainder r_u, due to the modulo operation. Node v can deduce the message m_u by using all of this information, along with its own message m_v.

Table 2 Communication of k-balanced messages, where $k = 4$ and $M = 9$. The executing node v, and its message value m_v, are highlighted. If v receives a message $r_u = 3$, it is able to deduce that the corresponding message m_u is 21.

Received remainder $r_u =$	'0'	'1'	'2'	'3'	'4'	'5'	'6'	'7'	'8'
$m_u - m_v =$ $(r_u - r_v) - \lfloor \frac{r_u - r_v}{k+1} \rfloor M$	-1	v	+1	+2	+3	+4	-4	-3	-2
Decoded message $m_u =$	18	19	20	21	22	23	15	16	17

The k-balanced message technique is of independent interest, and allows efficient algorithm design when nodes communicate locally-similar values.

3.1.3 Designing constant-size communication phases

In this section, we show that by applying the balanced messages approach, using only $O(1)$ beeping rounds, a node can deduce its neighbors' $prefix$ values (and whether some of them are suspicious), even though there are $O(N)$ different possible values of $prefix$.

From Lemma 1, we know that $|prefix|$ is a 6-balanced variable. Moreover, two neighboring nodes have similar $prefix$ values, which differ only in (at most 6 of) the last bits. Therefore, if a node can learn the last 6 bits of a neighboring $prefix$, and their exact positions, then it can fill up the empty bits (in more significant positions) using the bits from its own $prefix$. To learn that, one could use two consecutive communication subphases: the first communicates the *position* of the last bit (which is $|prefix|$, a 6-balanced variable) in a subphase with 13 rounds, and the second communicates an *ending* message with the last 6 bits (using a message from $\{1, \ldots, 2^6\}$, encoding all possible 6 letters combinations), in a subphase with 64 rounds. However, this does not work in $\mathcal{BEEP}$ because one needs to know, for every different ending message sent by neighbors, the corresponding position of the last bit (thus the corresponding position message). Although this is trivial in $\mathcal{ECONGEST}$, because messages from different neighbors are received on different edge ports, it is too costly to simulate this functionality in $\mathcal{BEEP}$ (see Sect. 1.1). Fortunately, as the message space is constant-size in both of these communication subphases, the Cartesian product of both message spaces is also constant-size. This allows to associate position and ending messages, using $O(1)$ rounds, even in $\mathcal{BEEP}$. Consequently, communication phases with 832 rounds (for messages in $\{1, .., 13\} \times \{1, \ldots, 2^6\}$) are needed to communicate enough information for a node to deduce all neighboring $prefix$ values.

On top of that, the nodes also need to communicate the boolean *suspicious*. For this reason, the message space is adapted to $\{1, .., 13\} \times \{1, \ldots, 2^6\} \times \{false, true\}$. This results in communication phases (introduced in Algorithm 1, Sect. 3.1.1) of length $c = 1664$ rounds, which although large, is still $O(1)$ size.

3.1.4 Remarks on the eventual leader election algorithm

As mentioned in the related work (Sect. 1.1), [5] is particularly relevant to our work. In this section, we discuss this in detail.
The structure of the information diffusion algorithm is essentially the same. The algorithm progresses in diffusion phases, consisting of a communication phase (corresponding to a single round in the considered $\mathcal{ECONGEST}$ model) where nodes send their $(prefix, suspicious)$

values, after which nodes change their $prefix$ variable depending on the $(prefix, suspicious)$ pairs received. Recall the 5 rules presented in Sect. 3.1.1: the set of the different possible changes for the $prefix$ variable is of a constant size, and these changes are meant to affect at most a constant number of (the last) bits of $prefix$. An important point in [5] is the proof that this set of changes allows the maximum identifier to spread over the network, in an optimal $O(D + \log n)$ number of phases. We use the same constant-size set of changes (for $prefix$). That is why Lemma 1 also applies to our algorithm.

However, the other core element of their information diffusion algorithm, the communication phase, cannot be used in $\mathcal{BEEP}$. In [5], nodes maintain up-to-date copies of the $prefix$ variables of their neighbors to circumvent the limited message size and can keep these copies up-to-date in a single communication phase of $O(1)$ rounds. In such a phase, nodes communicate what change was carried out (and which neighbor sent which message): sending the type of change is equivalent to sending the complete $prefix$ value in this situation. In $\mathcal{BEEP}$, nodes are unable to know which neighbor sent which message. Although this capability can be simulated, it is unlikely that it can be done without increasing the time complexity of [5]. Current methods result in a $O(\Delta^4)$ multiplicative factor (see discussion in Sect. 1.1).

One of the main contributions of this work is the introduction of the *k-balanced message* method to leverage the local constraints between (unbounded) values, which allows to communicate in $O(1)$ rounds. With the *k-balanced message* technique, a node can transmit a value of $prefix$ to its neighbors in $O(1)$ rounds (of $\mathcal{BEEP}$) only. This communication process differs greatly from that of [5].

3.2 Uniform Terminating Leader Election (Explicit LE)

Being often used as a primitive, the LE algorithm must be uniform and detect termination (e.g., so that it can be composed with other algorithms). Since classical approaches are not suited to $\mathcal{BEEP}$, we propose an explicit leader election algorithm using a different termination detection approach. Notice that, as mentioned previously, it is simple (in a synchronous setting) to transform the uniform eventual leader election algorithm, Algorithm 1, into a *non-uniform* one using knowledge of D and N, and thus of the time complexity expressed in terms of these parameters. Then, candidates can wait until the algorithm terminates, by counting rounds corresponding to the evaluated time complexity. However, this technique cannot be used here.

Instead, we use a primitive called *overlay networks*. We briefly describe it in Sect. 3.2.1. Then, in Sect. 3.2.2, an adapted version of this primitive is used to create a uniform termination detection component. This component is combined with the previously presented eventual leader election algorithm to obtain uniform explicit leader election.

3.2.1 Overlay network

The overlay network approach, in the context of leader election, was first used for $\mathcal{BEEP}$ in [12]. Such an overlay has a designated root, and consists of layers centered around the root. Nodes at a distance d from that root (*level d*), have *up links* (resp. *down links*) towards all neighboring nodes (of the overlay) at distance $d - 1$ (resp. at distance $d + 1$) from the root. Using these (virtual) links, the root can gather information about the network and disseminate it. The default behavior for non-root overlay nodes is to relay any beep received over an up (resp. down) link in some phase p, to all down (resp. up) links in phase $p + 1$.

Algorithm 2 Uniform Terminating Leader Election Algorithm.

```
1: IN: id: identifier ; OUT: leader: boolean, leaderId: identifier
2: $candidate := true$, $prefix := \epsilon$, $suspicious := false$        ▷ $\epsilon$ is the empty word
3: $leaderId := 0$, $leader := false$
4: for diffusion phase $p := 1$ ; $p{+}{+}$ do
5:     // First, a communication phase with $c = O(1)$ rounds.
6:     Communicate $(prefix, suspicious)$ to all neighboring nodes.
7:     // Then, apply predicates of rules 1 to 5 on received $(prefix, suspicious)$ pairs.
8:     Use received $(prefix, suspicious)$ values to update $prefix$, $candidate$ and $suspicious$.
9:     // Finally, termination detection phase with $s = O(1)$ rounds.
10:    Execute a termination detection phase.
11:    if $candidate$ and $prefix = \alpha(id)$ then
12:        If no beep is heard in down links, exit the loop.
13:    else
14:        If a beep is heard in up links, exit the loop.
15: $leaderId := \alpha^{-1}(prefix)$
16: if $candidate$ then $leader := true$        ▷ Last candidate becomes the leader
```

In more detail, overlays work in the following way. Time is divided into *overlay phases* of 9 rounds, where each phase consists of 3 subphases of 3 rounds each. The first 3 rounds are called *control* rounds, the next 3 - *up* rounds and the last 3 - *down* rounds. Each set of 3 rounds is numbered from 0 to 2.
When nodes join the overlay, they initialize a *depth* variable (in $\{0, 1, 2\}$), through which they know some information about their layer (and thus how to communicate with the other layers). The root node *joins* the overlay at a given time, and assigns itself $depth := 0$. The other nodes willing to join the overlay listen in all control rounds. Since overlay nodes beep in the control round *depth* (in all overlay phases), the joining nodes assign themselves $depth = beepHeard + 1 \ (mod\, 3)$, where $beepHeard$ is the smallest control round in which a beep was heard.

It is important that the *depth* variable satisfies some local constraints, to be guaranteed by the joining process. More specifically, for any distance d and for any given (overlay) node v in level d, all neighboring (overlay) nodes u in level $d-1$ (resp. in level $d+1$) must have $depth_u = depth_v - 1 \ (mod\, 3)$ (resp. $depth_u = depth_v + 1 \ (mod\, 3)$), where $-1 \ (mod\, 3) = 2$. With this property, nodes can listen over an up link (resp. down link) by listening in up (resp. down) round $depth - 1 \ (mod\, 3)$ (resp. $depth + 1 \ (mod\, 3)$). Moreover, nodes beep over an up link (resp. down link) by beeping in up (resp. down) round $depth \ (mod\, 3)$. In other words, communication through up and down links is the same as sending, or listening for, a *depth* message (using the multi-slot design pattern from [4], described in Sect. 3.1.2) using the corresponding subphase (a message from $M_{depth} = \{0, 1, 2\}$).

3.2.2 Termination detection component for explicit leader election

We describe the proposed *termination detection component* and its interactions with the eventual leader election algorithm (Algorithm 1). The termination detection component is meant to gather information from the whole network, on whether there are any candidates with a higher α-id. If there are none, the last candidate terminates and becomes leader. The combined final algorithm structure is given in Algorithm 2.

As in Algorithm 1, time is divided into diffusion phases, but these phases now include an additional *termination detection phase*. A termination detection phase consists of a border detection phase followed by an adapted overlay phase. The *border detection phase* is a communication phase for messages in $M_{prefix} = \{1, .., 13\} \times \{1, \dots, 2^6\}$, where nodes can detect if any of their neighbors has a different $prefix$ value (similar to the communication in Sect. 3.1.3). If that is the case for an overlay node (even the root) which has been part of the overlay for more than 6 phases, this node becomes a *border node* (i.e., there exists a neighbor with a different $prefix$ value). The *adapted overlay phase* is a communication phase with 3 subphases, each for messages in $M_{depth} \times M_{prefix}$. Each adapted overlay network is associated to a specific $prefix$ (i.e., that of the overlay's root, necessarily a candidate node). This prefix is used (communicated) so that nodes can detect whether the other endpoint of a down link or up link, is part of the same overlay (i.e., has the same $prefix$). Consequently, different overlay networks do not interfere with each other. A border detection phase has $|M_{prefix}|$ rounds and an adapted overlay phase has $9|M_{prefix}|$ rounds, thus a termination detection phase has $s = 10|M_{prefix}|$ rounds.

Upon having a well-formed $prefix$, each candidate designates itself as root and starts constructing an overlay network by using the termination detection phase. Nodes which have just joined the overlay and border nodes beep in their up links (relayed all the way back to the root) using the adapted overlay phase. As a result, the root hears beeps in its down links in each (termination detection) phase, until the overlay network covers the whole graph (Lemma 5). Moreover, the only overlay that can cover the whole graph is the overlay of the highest α-id node (because this node never changes $prefix$ depending on another node's α-id, and consequently never joins another candidate's overlay). Therefore, when the root hears no beeps in its down links (and is not a border node), it knows that its overlay covers the whole graph, and that it is the highest α-id node (thus the maximum id node). All other roots hear beeps in down links (or become border nodes), until their $prefix$ is changed.

In more detail, the *construction* of the adapted overlay networks is done as follows. Once a candidate node has a well-formed $prefix$ (after exactly $|\alpha(id)|$ diffusion phases), it sets itself up as an overlay's root (in phase $p = |\alpha(id)|$), but it stays silent for 6 termination detection phases (from phase p to phase $p + 5$) before beeping in the control rounds of phase $p + 6$ (and only in this phase). On the other hand, follower nodes with a well-formed $prefix$ attempt to join the overlay corresponding to $prefix$ right away. Once a follower node joins an overlay (in phase p'), it also stays silent for 6 termination detection phases before beeping in the control rounds of phase $p' + 6$. For any given node v that joins an overlay in termination detection phase p', its neighbors know if they join v's overlay or not, by phase $p' + 6$ at the latest (by Theorem 2). By staying silent for 6 termination detection phases upon joining, v ensures that all of its neighbors join the overlay at the same time (if they choose to join). Consequently, two nodes u and v, at the same distance d from an overlay's root r, never join r's overlay in different termination detection phases, and $depth_u = depth_v$. Otherwise, we could have $depth_u \neq depth_v$, which means a common neighbor of u and v at distance $d - 1$ from r would not have properly defined down links.

▶ **Lemma 4.** *Let r be the root of an overlay network. This overlay is properly constructed, that is, nodes at level d have the same depth value.*

Proof. Let us prove by induction that if a node at distance d from r joins r's overlay, then it is in phase $|\alpha(id_r)| + 6d$. Let us first consider a node v at distance 1 from r. For node v to join r's overlay, another overlay node must beep in the control rounds and $prefix_v$ must be equal to $\alpha(id_r)$, in the same phase. Notice that for any given two neighbors u and v, which are in different overlays, both nodes beep in different control rounds, because $prefix_u \neq prefix_v$.

In phase $|\alpha(id_r)| + 6$, r beeps in the control rounds, and thus v can join in that phase (if $prefix_v = \alpha(id_r)$). In addition, if $prefix_v \neq \alpha(id_r)$ in phase $|\alpha(id_r)| + 6$, then by Theorem 2, node v does not consider $\alpha(id_r)$ as the highest $prefix$ value it has encountered. As a result, it is impossible that $prefix_v = \alpha(id_r)$ after phase $|\alpha(id_r)| + 6$, and that v joins r's overlay after phase $|\alpha(id_r)| + 6$. The induction step ($d > 1$) is similar, starting from a node v at distance d from r. ◀

Because the adapted overlay networks are properly constructed, we can prove that as long as an overlay has not covered the whole network, follower nodes beep in their up links, stopping the root from becoming a leader. In more detail, after a candidate node beeps in the control rounds, it listens to its down links in every termination detection phase. As long as it hears a beep in these links, or is a border node, it does not become leader. Once no beep is heard, it becomes leader, sends a beep in its down links and terminates. On the other hand, after a follower node joins the overlay (in phase p), its beeps in its up links in the first 7 termination detection phases (from phase p to phase $p + 6$). It also beeps in the up links if it is a border node (and relays any beep heard through a down link). Finally, when a follower node hears a beep in its up links, it terminates. Consequently, before an overlay network covers the whole network, the root receives beeps in every (termination detection) phase.

▶ **Lemma 5.** *Let r be the root of an overlay network. Then from diffusion phase $|\alpha(id_r)| + 6$ onwards, node r hears beeps in its down links every phase, until it becomes a border node itself, or until its overlay covers the whole network.*

Proof. Let r be the root of an overlay network. From Lem. 4, r's overlay network is properly constructed, therefore the virtual links can be used. We define a (overlay) *downwards path* from node v to node u, as a sequence of down links starting in v and ending in u. A node u is *downwards reachable* from node v if there is an overlay downwards path from v to u.

Consider a follower node v, having just joined r's overlay (in phase p). Node v beeps in its up links for 7 termination detection phases after it joins (from phase p to phase $p + 6$). For each additional level in the overlay with nodes downwards reachable from v, v beeps in its up links during 7 additional termination detection phases (by relaying beeps heard in its down links, to its up links). Although the next layer (node u) is one further hop away from the root, and starts beeping in phase $p + 6$, v beeps during phase $p + 6$ (7^{th} termination detection phase after it joins) and relays u's first beep in phase $p + 7$. Consequently, there is no interruption in beeps sent through the up links. If an overlay node becomes a border node (some of its neighbors do not join in phase $p + 6$), then it beeps in up links in all phases $p' > p + 6$. If it exits the overlay, then its neighbors closer to the root become border nodes and beep in their up links. Therefore, the root keeps hearing beeps in its down links while levels are added to its overlay, but also if one of its overlay nodes becomes a border node. In that latter case, the root does not have the maximum id, and hears beeps in its down links until it becomes a border node itself. ◀

▶ **Theorem 6.** *Explicit Leader Election is solved (uniformly) in $O(D + \log n)$ rounds in the beeping model.*

Proof. The maximum identifier node v starts to construct its overlay network in phase $|\alpha(id_v)| + 6$, which is $O(\log n)$. For any given node $u \neq v$, $prefix_v$ never modifies its bits to match $prefix_u$. Consequently, v never joins u's overlay and v's overlay is the only one to grow until it covers the whole network, at a rate of adding a level every 6 diffusion phases. Thus, v's overlay covers the whole network after an additional $O(D)$ diffusion phases. Node v hears beeps in its down links for another additional $O(D)$ phases, since beeps from the last nodes to join the overlay take $O(D)$ rounds to reach v. After that, node v no longer hears beeps in its down links (Lemma 5) and is the only node in the network to terminate as leader. Then, it beeps in its down links, so that all nodes can terminate. ◀

3.3 Discussion and Perspectives

The deterministic LE algorithm presented in this section works without any change with an arbitrary (unbounded) id space $\{1, \ldots, U\}$. In this case, its time complexity is $O(D + \log U)$. For an unbounded id space, a known result from distributed bit complexity [11] gives a lower bound of $\Omega(\log U)$ for a network with two nodes. This implies a lower bound of $\Omega(D + \log U)$ for multi-hop networks. Consequently, the presented algorithm is asymptotically time-optimal even with an unbounded id space.

Furthermore, the algorithm can be modified to work if it starts with only a subset of nodes as candidates, or if the ids are not unique (as long as the highest id-encoding is still unique). Since a set of (non-unique) ids with a unique maximum can be generated without knowing n or N [17], this last variant can then be applied to obtain a randomized uniform (in both n and D) leader election algorithm.

4 Additional Results

LE is an important and often-used primitive when designing distributed algorithms. Thus, it makes sense that improving the time complexity of LE results in improved time complexities for other tasks. We propose improved algorithms for leader election in anonymous networks, MIS and coloring (in trees), and multi-broadcast.

4.1 Randomized Leader Election

When dealing with communication-restrictive models such as $\mathcal{BEEP}$, anonymity is especially important from an application viewpoint. Indeed, when considering large scale dynamic wireless networks, it might not be economically feasible to equip all nodes with unique identifiers. Additionally, nodes might be prevented from revealing their unique ids (explicitly or through their actions), due to privacy or security concerns [23]. For this case, a deterministic algorithm assuming unique identifiers can be adapted into a randomized one (w.h.p. time and safety guarantees) for anonymous networks, as stated in [13]. Indeed, one can generate a unique id w.h.p. by independently sampling $\theta(\log n)$ bits. But in return the knowledge of the network size n or at least some polynomial upper bound $N = O(n^c)$, is required.

4.2 MIS and 5-coloring for Trees

Symmetry breaking procedures such as maximal independent set (MIS) and coloring are important building blocks, especially in the communication-restrictive beeping model. Specifically, the MIS problem consists of choosing a set of nodes (*local leaders*) so that there are no two neighbors in the set (*independence*), and such that no other node of the network can be added to the set without causing the loss of the independence property. On the other hand, the c-coloring problem consists of assigning colors in $\{1, \ldots, c\}$ to the nodes of the network, such that neighboring nodes have different colors.
It is well-known that given a leader in tree networks (elected using $O(D + \log n)$ rounds), it is simple to 2-color the tree in $O(D)$ supplementary rounds. However, MIS and coloring have an $\Omega(\log n)$ lower bound (even in tree networks, as the bound from [21] holds for a graph

of disconnected pair of nodes), so this $O(D + \log n)$ 2-coloring algorithm is non optimal for most communication graphs. Still, using the proposed uniform leader election algorithm, we design uniform, asymptotically time-optimal $O(\log n)$ *MIS* and *5-coloring* algorithms in $\mathcal{BEEP}$, for tree networks.

We first give the algorithmic description of the 5-coloring algorithm. Roughly, low degree nodes are colored first using 3 colors, and the remaining nodes form a subgraph where the connected components have at most a logarithmic diameter. Using the LE algorithm, these connected components can be 2-colored in a logarithmic number of rounds. Now, we give more details as to how these steps are achieved. First, the $LimitedDegreeColoring$ algorithm from [3] is used to 3-color all nodes v with $deg(v) \leq 2$, in $O(\log n)$ rounds. Then, since all remaining nodes have a degree of at least 3, every remaining (non-colored) connected component (a tree) has a diameter of at most $\log n$. Thus, electing a leader for each such connected component requires $O(\log n)$ rounds. It is well-known that, in trees, coloring nodes according to their distance to the root can be done using 2 colors. This distance can be learnt by all nodes in $O(\log n)$ rounds. Specifically, nodes are synchronized after the leader election, and the leader broadcasts a beep, using a beep wave [13, 10] or reusing the overlay network from the leader election. The phase in which a node receives the broadcasted beep indicates its distance to the leader. Therefore the remaining non-colored nodes can be colored with another 2 colors, resulting in a 5-coloring for the communication graph.

From this 5-coloring, it is simple to compute an MIS in 5 additional rounds. Nodes with the same color form an independent set. Adding iteratively (at each round) nodes from each such set to a common independent set results in an MIS. Consequently, an MIS on the communication graph can also be computed in $O(\log n)$ rounds.

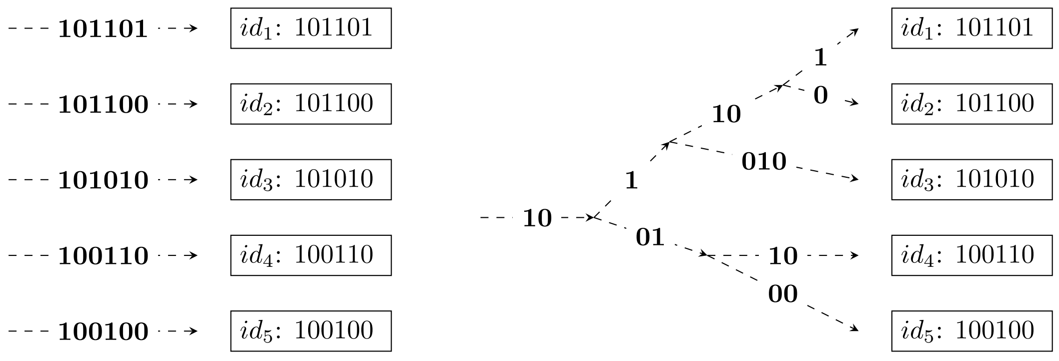
Notice that since all parts of the uniform 5-coloring algorithm are themselves uniform, it is a bit tricky to force nodes to resynchronize during the sequential execution. For this purpose, we use the EBET technique [3], to provide synchronization points in a uniform fashion - that is possible because, for every component of the proposed algorithm, the terminal state at a node can be detected locally - and thus solve the issue.

4.3 Multi-Broadcast with Provenance

Efficient communication primitives are fundamental building blocks in distributed computing, both for obtaining efficient algorithms and providing convenient abstractions of the actual communication mechanisms. These primitives are of even greater importance in $\mathcal{BEEP}$. When compared to other message-passing models, it is far more difficult to communicate messages throughout the network with beeps.

Now, consider the multi-broadcast problem. Multiple sources (k sources) have each a message they have to broadcast to all other nodes in the network. All messages are in $\{1, \dots, M\}$. In multi-broadcast with provenance, the k sources need to communicate their message, associated with their id, to all nodes in the network. In [10], an $O(D \cdot \log n + k \log \frac{nM}{k})$ round algorithm is given and the authors conjecture that the $D \cdot \log n$ term might be a lower bound. By presenting an $O(D + \log n)$ deterministic LE algorithm, this work shows that leader election is not a bottleneck for the multi-broadcast problem (whereas the previous deterministic LE algorithm required $O(D \cdot \log n)$ rounds). This suggests that $D \cdot \log n$ might be reducible to D in both the deterministic and randomized cases.

The multi-broadcast with provenance algorithm in [10] can be divided into three core components: leader election, communicating the ids of all k sources and finally using the order of these ids to communicate all messages properly to the leader (which then broadcasts the information to the network). In [10], the second component relies on the leader and



(a) Non-compact representation. **(b)** Compact representation.

Figure 1 Difference between non-compact and compact representations of k different values (ids), indicated by the number of bits used as labels.

performs k simultaneous binary searches, in $O(D \cdot \log n + k \log \frac{n}{k})$ rounds. Our contribution for this problem lies in improving the time complexities of the first and second components. The previous section (Sect. 3) improves the first component ([10] uses the leader election from [12]). More precisely, we use the leader election variant mentioned in Sect. 3.3, where the candidates can be a subset of all nodes. Here, only sources are candidates and the elected leader is the source with the maximum id. As for the second component, it is improved (whenever the number of sources k is sublogarithmic) by executing $k-1$ consecutive (variant) leader elections, where each leader election selects the source with the maximum id amongst the remaining non-elected sources. Notice that if nodes use their complete id for all $k-1$ consecutive leader elections, the time complexity is $O(k \cdot (D + \log n))$ rounds. By being more efficient and leveraging the information communicated through the previous leader elections, $k-1$ consecutive leader elections are executed in $O(k \cdot D + k \log \frac{n}{k})$ rounds only.

This result hinges on a compact manner of representing k unique values, which compresses the $k \log n$ bits required to communicate k identifiers consecutively, into $k \log \frac{n}{k}$ bits. As shown in Figure 1, after communicating id_1 (6 bits), communicating id_2 only takes one bit, and after that communicating id_3 takes an additional 3 bits. Thus, with this compact representation, after the first leader is elected (amongst sources), subsequent leader elections are more efficient as candidates for subsequent leader elections (non-elected sources) are not required to communicate their whole id. For this reason, we introduce the β-encoding: $\beta(i) = 0^{|bin|} \parallel 1 \parallel bin$ for an integer i and its binary representation bin. Contrary to α-encodings, the highest β-encoding is produced by integers with the shortest but highest binary representations.

Assume that all candidates for leader election (sources which have not yet been elected) are given an identifier $greaterID$, greater than their own. They compute a reduced identifier $reducedID$, consisting of all bits from the first difference with $greaterID$ onwards. Communicating $reducedID$ to other nodes is, in this setting, the same as communicating id since these other nodes have knowledge of $greaterID$ and thus deduce id from $reducedID$. Now, if candidates use the proposed deterministic LE algorithm with $\beta(reducedID)$, then the algorithm elects the node with the next maximum id value. Using this, the ids of all k sources are communicated to all nodes in $O(k \cdot D + k \log \frac{n}{k})$ rounds.

Thus, executing both $k-1$ consecutive leader elections and k binary searches in parallel, the k ids (of the sources) are communicated to all nodes in $O(min\{k, \log n\} \cdot D + k \log \frac{n}{k})$ rounds. Then, the messages are gathered and broadcast using the leader, in a further $O(D + k \log M)$ rounds, resulting in a $O(min\{k, \log n\} \cdot D + k \log \frac{nM}{k})$ algorithm for multi-broadcast with provenance.

References

1 Y. Afek, N. Alon, Z. Bar-Joseph, A. Cornejo, B. Haeupler, and F. Kuhn. Beeping a maximal independent set. *Distributed Computing*, 26(4):195–208, Aug 2013.

2 D. Alistarh, A. Cornejo, M. Ghaffari, and N. Lynch. Firefly synchronization with asynchronous wake-up. In *Workshop on Biological Distributed Algorithms*, 2014.

3 J. Beauquier, J. Burman, F. Dufoulon, and S. Kutten. Fast Beeping Protocols for Deterministic MIS and $(\Delta+1)$-Coloring in Sparse Graphs. In *IEEE INFOCOM*, 2018, to appear.

4 A. Casteigts, Y. Métivier, J. M. Robson, and A. Zemmari. Design Patterns in Beeping Algorithms. In *OPODIS*, pages 15:1–15:16, 2016.

5 A. Casteigts, Y. Métivier, J.M. Robson, and A. Zemmari. Deterministic leader election in $\mathcal{O}(D + \log n)$ time with messages of size $\mathcal{O}(1)$. In *DISC*, pages 16–28, 2016.

6 I. Chlamtac and S. Kutten. On broadcasting in radio networks - problem analysis and protocol design. *IEEE Transactions on Communications*, 33(12):1240–1246, 1985.

7 A. Cornejo and F. Kuhn. Deploying wireless networks with beeps. In *DISC*, pages 148–162, 2010.

8 A. Czumaj and P. Davies. Optimal leader election in multi-hop radio networks. *ArXiv e-prints*, 2015. `arXiv:1505.06149`.

9 A. Czumaj and P. Davies. Brief announcement: Optimal leader election in multi-hop radio networks. In *PODC*, pages 47–49, 2016.

10 A. Czumaj and P. Davies. Communicating with Beeps. In *OPODIS*, pages 1–16, 2016.

11 Y. Dinitz and N. Solomon. Two absolute bounds for distributed bit complexity. In *Structural Information and Communication Complexity*, pages 115–126, 2005.

12 K.-T. Förster, J. Seidel, and R. Wattenhofer. Deterministic leader election in multi-hop beeping networks. In *DISC*, pages 212–226, 2014.

13 M. Ghaffari and B. Haeupler. Near optimal leader election in multi-hop radio networks. In *SODA*, pages 748–766, 2013.

14 S. Gilbert and C. Newport. The computational power of beeps. In *DISC*, pages 31–46, 2015.

15 R. Guerraoui and A. Maurer. Byzantine fireflies. In *DISC*, pages 47–59, 2015.

16 S. Kutten, G. Pandurangan, D. Peleg, P. Robinson, and A. Trehan. On the complexity of universal leader election. In *PODC*, pages 100–109, 2013.

17 Y. Métivier, J.M. Robson, and A. Zemmari. Analysis of fully distributed splitting and naming probabilistic procedures and applications. *Theoretical Computer Science*, 584:115–130, 2015. Special Issue on Structural Information and Communication Complexity.

18 K. Nakano and S. Olariu. Randomized o(log log n)-round leader election protocols in packet radio networks. In *Algorithms and Computation*, pages 210–219, 1998.

19 S. Navlakha and Z. Bar-Joseph. Distributed information processing in biological and computational systems. *Commun. ACM*, 58(1):94–102, 2014.

20 D. Peleg. Time-efficient broadcasting in radio networks: A review. In *Distributed Computing and Internet Technology*, pages 1–18, Berlin, Heidelberg, 2007. Springer Berlin Heidelberg.

21 J. Schneider and R. Wattenhofer. What is the use of collision detection (in wireless networks)? In *DISC*, pages 133–147, 2010.

22 A. Scott, P. Jeavons, and L. Xu. Feedback from nature: An optimal distributed algorithm for maximal independent set selection. In *PODC*, pages 147–156, 2013.

23 J. Seidel. *Anonymous distributed computing: computability, randomization and checkability.* PhD thesis, ETH Zurich, Zürich, Switzerland, 2015. URL: `http://d-nb.info/1080812695`.