Beeping a Deterministic Time-Optimal Leader Election

Fabien Dufoulon, Janna Burman, Joffroy Beauquier

Beeping a Deterministic Time-Optimal Leader Election

Fabien Dufoulon
LRI, Université Paris-Sud, CNRS, Université Paris-Saclay, France
dufoulon@lri.fr

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

Joffroy Beauquier
LRI, Université Paris-Sud, CNRS, Université Paris-Saclay, 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 protocol 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 is sublogarithmic ($k < \log n$).

2012 ACM Subject Classification C.2.4 Distributed Systems, F.2.2 Nonnumerical Algorithms and Problems

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

Category Regular paper. Eligible for best student paper award.

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 first electing a leader to coordinate transmissions [11] (see also Sect. 4.3).

Wireless networks with severe restrictions on communication capabilities are an increasingly prevalent subject of study, e.g., [7, 18, 8, 13, 14, 9]. In order to represent these networks,

Cornejo and Kuhn [8] introduced a convenient formal framework: the *discrete beeping model*. In the discrete beeping 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 [16], which are reliant on primitive communications. Fireflies communicate through flashes of light [2, 15] and cells through the diffusion of specific chemical markers [1, 19].

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). One of the prime concerns is the design of *uniform* solutions, that is, of algorithms that do not require 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 wireless networks.
Amongst existing works for LE in the beeping model, 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 the beeping model, 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 deterministic and randomized (w.h.p. time) algorithms. Following this result, Czumaj and Davies [9, 10] present a randomized LE algorithm with $O(D+\log n)$ expected time in the beeping model. 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 the beeping model, 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 some presented here, are compared below (see Figure 1).

Figure 1 LE algorithms in the beeping model

Ref	Time	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 = O(n)
[12]	$O(D \cdot \log n)$: deterministic time	Deterministic	None
[9, 10]	$O(D + \log n)$ expected time	W.h.p.	N = O(n)
Here	$O(D + \log n)$: deterministic time	Deterministic	None
Here	$O(D + \log n)$ w.h.p.	W.h.p.	N = O(n)

[13, 9, 10, 12] concentrate on improving the time complexity of LE in general graphs. A different focus is presented in [14], where the goal is to minimize the size of the state machine representation of an (beeping model) algorithm solving randomized LE in single-hop networks.

Amongst the extensive leader election literature in other models, Casteigts et al. [6] is particularly relevant to our work. [6] 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. $\mathcal{ECONGEST}$ is much stronger than the beeping model, 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 encoding messages with both the colorings of sender and receiver, the constant-size $\mathcal{ECONGEST}$ model can be simulated, but with a prohibitive multiplicative factor of $O(\Delta^4)$ [4] (where Δ is the maximum degree).
Nevertheless, one of the main contributions of [6] is a rooted (in the maximum id node) spanning tree construction and *information diffusion* algorithm, designed to spread the maximum identifier efficiently, in a pipeline-like manner (rather than performing consecutive local comparisons of a complete identifier). This latter shift is crucial to the time-optimality of their algorithm, and is used in our work to improve on the $O(D \cdot \log n)$ result from [12].

1.2 Contributions

We present 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 the deterministic LE 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 the beeping model, outperforming all previous deterministic and randomized results.

Furthermore, using the proposed deterministic LE algorithm, we propose the first asymptotically time-optimal $O(\log n)$ time Maximal Independent Set (MIS) and 5-coloring algorithms for trees in the beeping model (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 the beeping model. 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 [9], when the number of sources is sublogarithmic ($k < \log n$), by executing k consecutive leader elections. Communication primitives are especially important in the beeping model, as they allow to deal with the interferences caused by simultaneous communications, and thus 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. Let $[k]$ be the set $\{1,\ldots,k\}$. The *identifier* of a node $u \in V$, $id(u)$, is an integer from $[N]$ where $N = n^c$ with a constant $c > 1$. N is an upper bound on the total number of nodes in G. Then, the *maximum length* over all identifiers in G is $l_{max} = O(\log N) = O(\log n)$.

Now, we give definitions pertaining to binary words. The operator $\|$ is for the *string concatenation*. The *length* of a binary word *bin* is denoted by $|bin|$. For any binary word *bin* and integer $j \in [|bin|]$, $bin[j]$ denotes the jth *most significant bit* of *bin*. Let x and y be two binary words, x is said to be the *prefix* (resp., *proper prefix*) of y if there exists a binary word (resp., non empty binary word) z such that $x \,\|\, z = y$. The empty word is denoted by ϵ.

The α-encoding [6] of an integer $i \in \mathbb{N}^{>0}$ is a binary word obtained from the binary representation *bin* of i. By definition, $\alpha(i) = 1^{|bin|} \,\|\, 0 \,\|\, bin$. In the LE section (Sect. 3), α-encodings of ids are used instead of ids, to ensure the algorithm works in a uniform (in n) manner, and thus from now on, we also use the term identifier to refer to α-encodings of ids. A binary word x is *well-formed* if there exists an integer i such that $x = \alpha(i)$. It is simple to prove that for every binary word x, there is at most one such integer i. Thus the α^{-1} function (α's "inverse") is defined on well-formed words.
We introduce a second encoding, the β-encoding: $\beta(i) = 0^{|bin|} \,\|\, 1 \,\|\, bin$. Whereas α-encodings are used to compare binary words (in a uniform manner), to look for the longest and highest binary words, β-encodings are used to look for the shortest but highest binary words (highest word amongst those with the shortest length).

2.2 Model Definitions

In the *beeping model*, 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*. The synchronous start assumption can be replaced by a slightly weaker variant, *wake-on-beep* [1], by using EBET [3], for a constant multiplicative overhead. In this variant, a subset of nodes wakes up in the first round, and all other nodes wake up if and only if a neighboring node beeps.
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 $\mathcal{ECONGEST}$ ($\mathcal{EC}$) model [17] is of interest in this paper. It is much stronger than the beeping model, as nodes communicate by sending messages of maximum length B (edge

bandwidth), commonly $O(\log n)$, in a round. Different messages can be sent to different neighbors and nodes receive the full content of all incoming messages, and distinguish between communication edges. It should be noted that messages from the same node are always received through the same edge. We use the term *constant-size* to express that the bandwidth B is bounded by some constant value.

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 from 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 does not know p (and is unable to infer it from the information it holds). 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.

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 is aware of when it terminates. Then, we define *terminating leader election*, where the algorithm terminates and nodes are aware that 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.

3 Leader Election Algorithms

Classical approaches used to solve leader election in $\mathcal{CONGEST}$ models do not work in the beeping model. Although they can be adapted using a transformer, doing so is too costly in most of the graphs (see discussion in the related work section: Sect. 1.1).
To solve the strongest version of LE, 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.

3.1 Uniform Eventual Leader Election

The uniform eventual leader election algorithm (Algorithm 1) is described first (Sect. 3.1.1). Then, in Sect. 3.1.2, k-balanced messages are introduced. They are used to allow constant-size *communication phases* composed of rounds and dedicated to the communication (large) messages respecting local constraints. Using the k-balanced message technique, a detailed

[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.

description of the communication phases (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 Eventual leader election

Algorithm 1 Eventual Leader Election Algorithm

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

In the *eventual leader election* algorithm (Algorithm 1), all nodes aim to spread their identifiers to the whole network (*information diffusion algorithm*). They start out as *candidates*, with two variables: Z and *suspicious*. The *suspicious* boolean is initialized to $false$. The binary word Z is initialized to the empty word ϵ. Z represents the prefix of an identifier, and most of the time, the highest prefix the node is aware of. A prefix Z_u is said to be *higher* than a prefix Z_v, if Z_v is a proper prefix of Z_u, or if $Z_u > Z_v$, i.e., in the first differing bit's position, Z_u has 1 and Z_v has 0, even if $|Z_u| < |Z_v|$.

Nodes execute *diffusion phases* (of c rounds) synchronously. A diffusion phase consists of one *communication phase* of $c = O(1)$ rounds (line 6), used to send Z and *suspicious* to all neighbors, which is described in detail in Sect. 3.1.3. In that same diffusion phase, each node receives $(Z, suspicious)$ 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 its Z_v value is a *locally higher prefix*, using the received pairs (see details below). If that is the case, then it appends a bit from its identifier to Z_v (if Z_v is a proper prefix of $\alpha(id_v)$), or does nothing (if $Z_v = \alpha(id_v)$). Otherwise, it modifies Z_v depending on the highest detected Z value, and becomes a *follower*. It can no longer become a leader. If that modification removes bits from Z_v, node v is said to be *suspicious* for the following phase, and $suspicious_v$ is assigned to true for one phase.

More precisely, in line 8, a set of predicates given below is evaluated on the set of the received $(Z, suspicious)$ pairs, in the given order of priority, and the appropriate described updates is done. If any of these predicates is true (where w and y are binary words), Z_v is not a locally higher prefix.

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

5. If there exists a neighbor u with $Z_u = (Z_v \parallel 0 \parallel w)$ then append 0 to Z_v.
Indeed, if a neighbor u is suspicious and Z_u is a proper prefix of Z_v, then a neighbor of u has a higher prefix than Z_v, or is changing (its Z variable) according to one (rule 1 above). By deleting the last bits of Z_v, node v is matching Z_v to that unknown node's Z value. In all 4 other cases, Z_u is a higher prefix than Z_v, therefore Z_v modifies its last bits to match Z_u.

Additional local computations in lines 9-12 conclude a diffusion phase. Once a node's Z variable is well-formed, that is, once $Z = \alpha^{-1}(x)$ for any binary word x, this node sets itself as 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 [6]. Thus the described information diffusion process satisfies Lemma 1 and Theorem 2 below, adopted from the results of [6] and adapted here to our beeping algorithm (see Sect. 3.1.4 for more details).

▶ **Lemma 1** (Beeping version of Lemma 8 in [6]). *Let u and v be two neighboring nodes. Then, $||Z_u| - |Z_v|| \leq 6$. Moreover, assume w.l.o.g. that $|Z_u| \leq |Z_v|$, then Z_u and Z_v are identical, except in at most 6 bits: from the $|Z_u|$th bit (possibly included) to the $|Z_v|$th bit. More precisely, if the $|Z_u|$th bit differs in Z_u and Z_v, then $||Z_u| - |Z_v|| < 6$*

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

Proof. Let l be the highest id node. We prove the theorem by induction on r.
Node l has the highest identifier X, thus it appends a bit from X in each diffusion phase. After $|\alpha(X)|$ phases, $Z_l = \alpha(X)$. 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, Z_u and Z_v differ in less than 6 bits, from the $|Z_u|$th bit (included) to the $|Z_v|$th bit. Since $Z_v = \alpha(X)$ (induction hypothesis), node u necessarily appends a bit from Z_v in each of the 6 following phases, until $Z_u = \alpha(X)$. ◀

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

▶ **Theorem 3.** *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 highest identifier in the network. From Theorem 2, $Z_v = \alpha(X)$ after $O(D + \log n)$ phases. Nodes have the leader's identifier by applying the α^{-1} function. As each diffusion phase consists of $c = O(1)$ rounds, $Z_v = \alpha(X)$ after $O(D + \log n)$ rounds.
Moreover, the highest id node is well-formed after $|\alpha(X)| = O(\log n)$ phases, thus after $O(\log n)$ rounds. As a result, the highest id node is, and remains, a leader from that point onwards. ◀

3.1.2 Balanced messages

A basic component in the beeping model, the multi-slot design pattern [5], allows to communicate constant-size messages with no sender id, nor multiplicity, given a synchronous

start. This component works in communication phases of M rounds, if it allows at most M possibles messages (in $\{1,\ldots,M\}$). Beeping in the jth round of a phase is equivalent to sending the message j, but 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 component cannot be used to directly send Z values, as these values are in $\{1,\ldots,N\}$, and communication phases would be $O(N)$ rounds long. But this component 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, for all $u \in \mathcal{N}(v)$, $|var_u - var_v| \leq k$).

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 component, with phases of $M = 1+2k$ rounds (where k needs to be previously known to all nodes). Then, the receiver node, knowing both its remainder, the sender's remainder and the fact that the messages are k-balanced, can deduce the sender's original message (but does not know if multiple nodes have sent this message).
Specifically, let v be the receiver and u the sender. Node v simply deduces the original message m_u from the received remainder message r_u: $m_u = m_v + r_u - r_v - \lfloor \frac{r_u - r_v}{k+1} \rfloor M$.

Figure 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.

Remainder received: $r_u =$	'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

Consider the example depicted in Fig. 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.

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 how using only $O(1)$ rounds in the beeping model, with the help of the k-balanced messages, a node can deduce its neighbors' Z values (and whether some of them are suspicious), even though there are $O(N)$ different possible values of Z.

From Lemma 1, we know that $|Z|$ is a 6-balanced variable. Moreover, two neighboring nodes have similar Z values, which differ only in (a constant number of) the last bits. Therefore, if a node knows the last 6 bits of a neighboring Z value, and their exact positions,

then it can fill up the empty bits (in more significant positions) using the bits from its own Z variable.
It is obvious that the last 6 bits and their exact positions can be deduced from the last 6 bits and the last bit's position. For that, one could use two consecutive communication subphases: the first communicates the *position* of the last bit (which is $|Z|$, 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 the beeping model because one needs to know, for every ending message, the corresponding position of the last bit (thus the corresponding position message). Although this is trivial in $\mathcal{ECONGEST}$, because the same communication edge would be used by a given neighboring node, it is too costly to simulate this functionality in the beeping model (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 the beeping model. Consequently, communication phases with 832 rounds (for messages in $\{1, .., 13\} \times \{1, \ldots, 2^6\}$) can be used to communicate enough information for a node to deduce all neighboring Z values.

Nevertheless, recall that nodes also need to know whether a neighbor removed bits from Z in the last round (indicated by the boolean *suspicious*). For that, the message space is adapted to $\{1, .., 13\} \times \{1, \ldots, 2^6\} \times \{false, true\}$, where the value of *suspicious* is also communicated. 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), [6] 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 $\mathcal{EC}$ model) where nodes send their $(Z, suspicious)$ values, after which nodes change their Z variable depending on the $(Z, suspicious)$ pairs received. Recall the 5 rules presented in Sect. 3.1.1: the set of the different possible changes for the Z variable is of a constant size, and these changes are meant to affect at most a constant number of (the last) bits of Z. An important point in [6] 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 Z). That is why Lemma 1 also applies to our algorithm.

However, the other core element of their information diffusion algorithm, the communication phase, displays fundamental flaws when considering the beeping model. In [6], nodes maintain up-to-date copies of the Z variables of their neighbors to circumvent the limited message size, and can keep these copies up-to-date in a single $O(1)$ rounds communication phase. 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 Z value in this situation.
In the beeping model, nodes are unable to know which neighbor sent which message (which communication edge transmitted a particular message). Although this capability can be simulated, it seems improbable that it can be done without increasing the time complexity of [6]. Current methods result in a $O(\Delta^4)$ multiplicative factor (see discussion in Sect. 1.1), because symmetry-breaking procedures are required to distinguish between neighbors beeping

at the same time.

One of the main contributions in this work is the introduction of the *k-balanced message* method to leverage the local constraints between (unbounded) values, to allow communication in $O(1)$ rounds. With the *k-balanced message* technique, a node can transmit a value of Z to its neighbors in $O(1)$ rounds (of the beeping model) only. This communication process differs greatly from that of [6].

3.2 Uniform Terminating Leader Election (Explicit LE)

The LE algorithm is often used as a primitive. It is thus essential for it to be uniform and detect termination (e.g., so it can be composed with other algorithms). Since classical approaches are not suited to the beeping model, we propose a uniform terminating leader election using a different termination detection approach.
It should be noted that, as mentioned previously, it is simple (in the synchronous setting) to transform the uniform eventual leader election algorithm, Algorithm 1, into a *non-uniform* algorithm (using knowledge of D and N). Indeed, since the time complexity of the algorithm is known to all nodes, candidates can wait until the algorithm terminates, i.e., when there is a single candidate with a well-formed Z.

First, we briefly describe in Sect. 3.2.1. the primitive that we use - the *overlay networks*. Then, in Sect. 3.2.2, an improved 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 the beeping model in [12]. An overlay network has a designated root, and consists of layers centered around that 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 links, the root can gather information about the network, and disseminate it. The default behavior for overlay nodes is to relay any message received over one up (resp. down) link to all down (resp. up) links.
Nodes have a *depth* variable (in $\{0,1,2\}$). This variable is initialized when nodes join the overlay, and needs to respect some local constraints (detailed below) when initialized. Once *depth* is properly initialized, it is key to communications over the up and down links (which cannot be explicitly used in the beeping model). Nodes communicate in *overlay phases* of 9 rounds. The first 3 rounds are called *control* rounds, the next 3 *up* rounds and the last 3 rounds *down* rounds. Each set of 3 rounds (*round triplet*) is numbered from 0 to 2.
To listen over an up link (resp. down link), a node listens in up (resp. down) round $depth-1 \ (mod\ 3)$ (resp. $depth+1 \ (mod\ 3)$), where $-1 \ (mod\ 3) = 2$. To communicate over an up link (resp. down link), a node beeps in up (resp. down) round $depth \ (mod\ 3)$. In other words, communication through up and down links is the same as sending a *depth* message using the corresponding triplet of rounds (a message from $M_{depth} = \{0,1,2\}$).

Finally, we describe the joining process. Nodes in the overlay beep in control round *depth*. Nodes looking to join listen in all control rounds. If a beep is heard (in round $beepHeard$), the node joins the overlay, by assigning itself $depth = beepHeard + 1 \ (mod\ 3)$. If more than one beep is heard, the node chooses the smallest one.
However, for the overlay to work properly, the *depth* variable has to satisfy some local constraints. 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. $depth_u = depth_v + 1 \ (mod\ 3)$). Otherwise, node v is unable to send messages over the up and down links.

In Sect. 3.2.2, a modified version of the described overlay network is used. Each round triplet (capable originally of sending messages from M_{depth}) is modified into a subphase capable of sending messages from $M_{depth} \times M_Z$, where $M_Z = \{1, .., 13\} \times \{1, \ldots, 2^6\}$. With this change, nodes can detect whether the other endpoint of a down link or up link, is communicating the same message in the M_Z field.

3.2.2 Termination detection component for explicit leader election

Algorithm 2 Uniform Terminating Leader Election Algorithm

1: **IN:** *id*: identifier ; **OUT:** *leader*: boolean, *leaderId*: identifier
2: $candidate := true$, $Z := \epsilon$, $suspicious := false$ ▷ ϵ is the empty word
3: $leaderId := \epsilon$, $leader := false$
4: **for** diffusion phase p ; p++ **do**
5: // *First, a communication phase with* $c = O(1)$ *rounds.*
6: Communicate $(Z, suspicious)$ to all neighboring nodes.
7: // *Then, apply predicates on received* $(Z, suspicious)$ *pairs.*
8: Use all received $(Z, suspicious)$ values to update Z, *candidate* and *suspicious*.
9: // *Finally, termination detection phase with* $s = O(1)$ *rounds.*
10: If termination is detected, exit the loop.
11: $leaderId := \alpha^{-1}(Z)$
12: **if** *candidate* **then** $leader := true$ ▷ Last candidate becomes the leader

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 higher id candidates. If there are none, the last candidate terminates and becomes leader. The combined final algorithm structure is given in Algorithm 2.
First, we describe the *construction* of the overlay networks. If they are not constructed properly, they cannot be used to gather information on whether there are any other candidates. We use modified overlay networks, where up and down links exist only between nodes with the same Z value, i.e., in the same overlay. Moreover, nodes with different Z values do not detect each other when beeping in the control rounds.
Once a candidate node has a well-formed Z (after exactly $|\alpha(id)|$ diffusion phases), it sets itself up as an overlay's root, but it waits 5 diffusion phases before beeping in the control rounds of the 6th phase (and only in this phase). On the other hand, follower nodes with a well-formed Z attempt to join the overlay corresponding to Z right away. Once a follower node joins an overlay, it also waits 5 phases before beeping in the control rounds of the 6th phase. By blocking follower nodes from joining an overlay too quickly, the algorithm makes that sure their *depth* variables are properly initialized. Moreover, the overlay network grows by one level every 6 diffusion phases (which aligns with the minimum information diffusion speed), unless a higher id root is detected.

▶ **Lemma 4.** *Let r be the root of an overlay network. Its overlay is properly constructed. That is, (r's overlay) nodes at level d have the same depth value.*

Proof. Let us prove by induction that for all nodes at distance d from r, if they join r's overlay, then they all join 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 Z_v must be equal to $\alpha(id_r)$, in the same phase. Notice that nodes that are in different overlays beep in different control rounds, because of the message modification.
In phase $|\alpha(id_r)| + 6$, r beeps in the control rounds, and thus v can join in that phase (if $Z_v = \alpha(id_r)$). In addition, if $Z_v \neq \alpha(id_r)$ in phase $|\alpha(id_r)| + 6$, then by Theorem 2, node v does not consider $\alpha(id_r)$ as the highest Z value it has encountered. As a result, it is impossible that $Z_v = \alpha(id_r)$ 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 the same, starting from a node v at distance d from r. ◀

Then, we detail how a candidate node detects that it is the last to remain as candidate. The idea is that, as long as an overlay has not covered the whole network, follower nodes send messages through up links, stopping the root from becoming a leader. Furthermore, only the overlay of the highest id node can cover the whole network.
After a candidate node beeps in the control rounds, it listens to its down links in every phase. As long as it hears a message through these links, or is a *border node* (there exists a neighbor with a different Z value), it does not become leader. Once no message is heard, it becomes leader. On the other hand, a follower node sends messages in the up links in the 7 phases after it joins the overlay. It also sends messages in the up links if it is a border node (and relays any message heard through a down link).
Consequently, before an overlay network covers the whole network, the root receives messages in every (termination detection) phase.

The termination detection phase builds upon the k-balanced message technique, introduced in Sect. 3.1.2. Specifically, a termination detection phase consists of a border detection phase followed by a modified overlay phase.
The *border detection phase* is a communication phase for messages in $M_Z = \{1, .., 13\} \times \{1, \ldots, 2^6\}$, where nodes can detect if any of their neighbors has a different Z value.
In the *modified overlay phase*, different overlay networks (with different Z values) do not share round triplets. When messages (from M_{depth}) are sent over up and down links, a message (from M_Z) is associated such that nodes can check whether the other endpoint of a down link, or up link, is in the same overlay network or not. A modified overlay phase has $3s'$ rounds, and thus a termination detection phase has $s = 10|M_Z|$ rounds.

▶ **Lemma 5.** *Let r be the root of an overlay network. Then from diffusion phase $|\alpha(id)| + 6$ onwards, node r receives messages in its down links every phase, until it is a border node or no longer part of the overlay, or until the overlay cover the whole network.*

Proof. Let r be the root of an overlay network. From Lemma 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 a overlay downwards path from v to u.

Consider a follower node v, having just joined r's overlay. Node v beeps in its up links in the first 7 phases after it joins. For each additional level in the overlay, with nodes that are downwards reachable from v, it beeps an additional 7 phases, such that there is no interruption in the up messages.
If an overlay node becomes a border node (some of its neighbors do not join in the 6th phase after it joins the overlay), then it does not stop sending beeps after the first 7 phases (as if

levels were constantly added to the overlay). If it exits the overlay, then its neighbors which are closer to the root become border nodes and beep in their up links, with no interruption in the up messages.
Therefore, the root keeps hearing message 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 highest 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 highest identifier node starts to construct its overlay network in phase $|\alpha(id)|+6$, which is $O(\log n)$. This overlay keeps growing until it covers the whole network, at a rate of a level every 6 diffusion phases. Therefore, the overlay covers the whole network after an additional $O(D)$ diffusion phases, and the last up messages are propagated upwards by the overlay for an additional $O(D)$ phases. After which, the root (highest id node) no longer hears messages in its down links (Lemma 5) and terminates as leader. By Lemma 5, it is the only node in the network to do so. Then, it broadcasts a down message to all nodes, so that they know when to terminate. ◀

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

Anonymous networks, when dealing with communication-restrictive models such as the beeping model, are especially important from an application viewpoint. Indeed, when considering large scale wireless networks, it might not be economically feasible to equip all nodes with unique identifiers. But nodes might be disinclined to reveal their unique ids (explicitly or through their actions), due to privacy or security concerns [20]. However, a deterministic algorithm assuming unique identifiers can be adapted into a randomized (w.h.p. time and safety guarantees) algorithm for anonymous networks, as stated in [13]. Indeed, by independently sampling $\theta(\log n)$ bits to create identifiers, these identifiers will be unique w.h.p., 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 differing 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 an extra $O(D)$ rounds. However, MIS and coloring have a $\Omega(\log n)$ lower bound (even in tree networks), so an $O(D + \log n)$ algorithm is non optimal for most communication graphs. Using the proposed uniform leader election algorithm, we design uniform, asymptotically time-optimal $O(\log n)$ *MIS* and *5-coloring* algorithms in the beeping model, for tree networks.

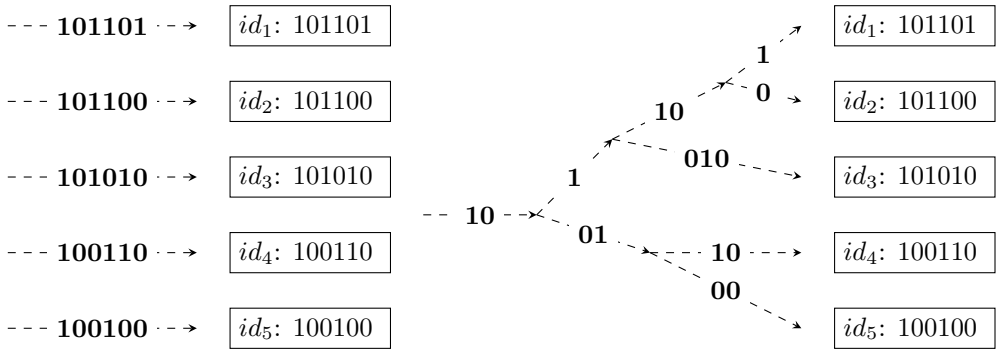
We give the algorithmic description of the 5-coloring algorithm. Low degree nodes are colored first using 3 colors, and the remaining nodes form a subgraph where the connected components have at most a logarithmic diameter. Using the LE algorithm, these connected components can be 2-colored in a logarithmic number of rounds.
Now, we give more details as to how these steps are achieved. First, the $LimitedDegreeColoring$ algorithm from [3, 4] is used to 3-color all nodes v with $deg(v) \leq 2$, in $O(\log n)$ rounds. Then, since all remaining (non-colored) connected components have diameter at most $\log n$, electing a leader for each such connected component requires $O(\log n)$ rounds. It is well-known that coloring nodes according to their distance to the root, in trees, 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, 11] 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. Thus 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.

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 comfortable abstractions from the actual communication mechanism. These primitives have even greater importance in the beeping model. When compared to other message-passing models, it is far more difficult to communicate messages throughout the network with beeps. Indeed, when considering both the limited message size and the interference produced by simultaneous beeps, a very delicate coordination between nodes is necessary for intra-network communication to succeed.

Now, consider the multi-broadcast problem. Multiple sources (k sources) have each a message they wish to broadcast to all other nodes in the network. All messages are in $\{1, \ldots, M\}$. In multi-broadcast with provenance, the k sources need to communicate their message, associated with their id, to all nodes in the network. Obviously, the most efficient solution to the multi-broadcast problem is to have nodes communicate messages simultaneously, as the interference in the beeping model is non-destructive (as opposed to radio networks). However, excessive interference hinders nodes from understanding and extracting the messages from the simultaneous communications. In [11], 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 using the deterministic LE algorithm proposed here, we prove that it is not, as it



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

Figure 3 Difference between non-compact and compact representations of k different values (ids), indicated by the number of bits used as labels

can be slightly reduced in cases where the number of sources is sublogarithmic ($k < \log n$). That suggest that $D \cdot \log n$ might be reducible to D in both the deterministic and randomized cases. Moreover, it is likely that using randomization, ranking the k sources can be done faster than with k consecutive leader elections.

The multi-broadcast with provenance algorithm in [11] can be divided into three core components: leader election, computing a ranking for the k sources and finally using the ranking to communicate all messages properly to the leader (who then broadcasts the information to the network). In [11], the second component relies on the leader and 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, as [11] uses the leader election from [12]. As for the second component, it is improved by executing $k-1$ consecutive leader elections (assuming the first LE was executing using sources only), resulting in $O(k \cdot D + k \log \frac{n}{k})$ rounds. However, the $k-1$ consecutive leader elections - using ids - takes $k \log n$ rounds instead of $k \log \frac{n}{k}$. Thus, it is essential to be more efficient and use the information communicated through the previous leader elections.

We do this by using 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 3, 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.
Assume all candidates for leader election are given an identifier id_g, greater than their own. They compute a reduced identifier id_r, consisting of all bits from the first difference with id_g onwards. Communicating id_r to other nodes is, in this setting, the same as communicating id since these other nodes have knowledge of id_g and thus deduce id from id_r. Now, if candidates use the proposed deterministic LE algorithm with $\beta(id_r)$, where β-encoding is defined in Sect. 2.1, then the algorithm elects the node with the next highest id value. Using this, we can compute a ranking of the k sources in $O(k \cdot D + k \log \frac{n}{k})$.
Executing both this ranking algorithm and the k binary searches in parallel, communicating all k ids (of the sources) can be done 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.

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 (Δ+1)-Coloring in Sparse Graphs (Extended Version). Technical report. URL: `https://hal.archives-ouvertes.fr/hal-01679099`.
4. J. Beauquier, J. Burman, F. Dufoulon, and S. Kutten. Fast Beeping Protocols for Deterministic MIS and (Δ+1)-Coloring in Sparse Graphs. In *IEEE INFOCOM*, 2018, to appear.
5. A. Casteigts, Y. Métivier, J. M. Robson, and A. Zemmari. Design Patterns in Beeping Algorithms. In *OPODIS*, pages 15:1–15:16, 2016.
6. 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.
7. I. Chlamtac and S. Kutten. On broadcasting in radio networks - problem analysis and protocol design. *IEEE Transactions on Communications*, 33(12):1240–1246, December 1985. `doi:10.1109/TCOM.1985.1096245`.
8. A. Cornejo and F. Kuhn. Deploying wireless networks with beeps. In *DISC*, pages 148–162, 2010.
9. A. Czumaj and P. Davies. Optimal leader election in multi-hop radio networks. *ArXiv e-prints*, May 2015. `arXiv:1505.06149`.
10. A. Czumaj and P. Davies. Brief announcement: Optimal leader election in multi-hop radio networks. In *PODC*, pages 47–49, 2016.
11. A. Czumaj and P. Davies. Communicating with Beeps. In *OPODIS*, pages 1–16, 2016.
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. Navlakha and Z. Bar-Joseph. Distributed information processing in biological and computational systems. *Commun. ACM*, 58(1):94–102, December 2014.
17. D. Peleg. *Distributed Computing: A Locality-Sensitive Approach*. Monographs on Discrete Mathematics and Applications. Society for Industrial and Applied Mathematics, 2000.
18. 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.
19. 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.
20. J. Seidel. *Anonymous distributed computing: computability, randomization and checkability*. PhD thesis, ETH Zurich, Zürich, Switzerland, 2015. URL: `http://d-nb.info/1080812695`.