Leader election with constant energy complexity

Ny Aina Andriambolamalala[1] and Vlady Ravelomanana[2]

[1]IRIF — University Denis Diderot — France. —Ny-Aina.Andriambolamalala@irif.fr
[2]IRIF UMR CNRS 8243 — University Denis Diderot — France. — vlad@irif.fr

April 12, 2019

Abstract

Intensively studied for about 30 years, leader election is a fundamental problem in distributed computing. Assuming that distributed network devices are battery-powered, managing power consumption is important. In this paper, we focus on energy complexity of leader election algorithms. Our results concern two distributed network models, single-hop beeping networks (BN) and single-hop radio networks with collision detection (RNCD). In these models, when the nodes are initially indistinguishable, we design randomized leader election algorithms terminating after sublinear number of rounds (in the size n of the network), with high probability and with no node being awake for more than a constant number of rounds. We also design randomized leader election algorithm with poly-logarithmic time complexity, log-high probability of success and constant energy complexity. These results hold even as all nodes are aware of a polynomial upper bound of n.

1 Introduction

As one of the most fundamental protocols in distributed computing, leader election problems have been extensively studied over the years [13, 14, 15, 7, 10, 18, 8]. In this paper, as distributed devices are battery-powered most of the time and since the waking time (sending messages and listening to the network) costs more energy than internal computations, we design randomized leader election algorithms minimizing the energy consumption per node (energy complexity) of the network. Our algorithms elect a leader in Radio Networks with Collision Detection (RNCD) as well as in Beeping Networks (BN) in $O(n^{O(1)})$ (resp. $O(\log^{O(1)} n)$) with high probability[1] (*w.h.p.*) (*resp.* log-high probability[2] (*w.Lh.p.*)) and have constant energy complexity even if no node knows the size of the network n but a polynomial upper bound $u > n$ is given in advance to all the nodes[3] More importantly, our algorithms achieve their designated goal by sending $O(1)$ bits with a waking time slots of $O(1)$ per node.

1.1 The network and communication models

Single-hop RNCD: Introduced by Chlamtac and Kutten in the 1880's [4], in this model, communications occur in synchronous rounds (time slots). In each round, each node independently decides whether to transmit a message of length $O(\log n)$ or to listen to the network or to remain idle (asleep). Only listening nodes can check the status of the common channel which can be:

[1]An event ε_n is said to occur with high probability if $\mathbb{P}[\varepsilon_n] \geq 1 - \frac{1}{n^c}$ for any constant $c > 0$ that does not depend on n.

[2]Event ε_n occurs with log-high probability if $\mathbb{P}[\varepsilon_n] \geq 1 - \frac{1}{\log^c n}$ for any constant $c > 0$ that does not depend on n.

[3]u is a polynomial upper bound of n if $u = O(n^{1+c})$ for some constant $c > 0$.

SINGLE if exactly one node is transmitting on the channel, NULL if no node is transmitting and COLLISION if at least 2 nodes are transmitting.

Single-hop BN: This model has been introduced in 2010 by Cornejo and Kuhn [5] and is strictly weaker than the RNCD one [1]. In the beep model, communication works as in RNCD but the nodes only can transmit a beep (1 bit messages) and listening nodes cannot distinguish between a single beep emitted by one of its neightbors or two or more beeps.

We assume that nodes are able to generate random discrete variables (see for instance Devroye [12]). We also assume that all nodes do not know the exact value of the size n of the network but they are given a common polynomial upper bound u of n. Note that without this upper bound assumption, Chang, Kopelowitz, Pettie, Wang and Zhan [3] have established a lower bound of $\Omega(\log \log^* n)$ for the energy complexity of election algorithms in the RNCD model. The nodes are initially completely anonymous and indistinguishable.

Note that in this paper even for the RNCD model, we only allow messages of one bit length. All proofs of Lemmas and Theorems are given in appendix due to space limitation.

1.2 Related works and new results

For single-hop networks, in the late 70's Tsybakov [17] and Capetanakis [2] designed deterministic leader election algorithms terminating in $O(\log n)$ rounds. These algorithms are asymptotically optimal, since Greenberg and Winograd [9] have established a lower bound of $\Omega(\log n)$ for deterministic algorithms.

On the randomized side, Willard [18] designed protocols working in expected $O(\log \log n)$ and in $O(\log n)$ time slots under the high probability requirement. Nakano and Olariu [16] provided a randomized algorithm with running time $O(\log \log n) + o(\log \log n) + O(\log 1/\varepsilon)$ and with probability of termination exceeding $1 - \varepsilon$. A lower bound of $\Omega(\log n)$ for this problem has been presented by Nakano and Olariu [16] for uniform protocols. Amongst other results, Ghaffari, Lynch and Sastry [8] extended this result to all protocols and presented a lower bound of $\Omega(\min\{\log(u/n), \log(1/\varepsilon)\})$ for randomized leader election algorithms with termination probability greater than $1 - \varepsilon$ (u is any upper bound of n).

In [3], Chang, Kopelowitz, Pettie, Wang and Zhan started by counting approximately the number of participants. This approximate counting phase costs $O(n^{o(1)})$ time slots with $O(\log \log^* n)$ energy complexity.

In this extended abstract, our algorithms design is based on the existence of discrete random variables (*r.v.*) X such that, if $X_1, X_2, \cdots, X_K$ are K independent copies of X, then, with probability greater than $p = 1 - O\left(\frac{1}{\log^{O(1)} K}\right)$,

(∗) the maximum of the X_i's is *unique* and

(⊙) such maximum is of order $\log^{O(1)} K$.

Using these properties, as each node s independently generates *r.v.* X_s according to a distribution satisfying *(∗)* and *(⊙)*, all nodes can find the interval of size $O(\log K \log \log K)$ containing the unique maximum without sending any message (with probability greater than p). The details will be given in section 1. Then, the algorithm terminates by finding the unique node holding the maximum and this latter simply becomes the leader.

The following table shows comparison between existing results and ours.

Existing results				Probability
Model	Time	Energy	messages size	of success
Kardas, Klonowski, Pajak [11] *RN Strong-CD*[4], n unknown	$O(\log^{\varepsilon} n)$	$O(\log\log\log n)$ 1-bit messages		in expectation
Chang, Kopelowitz, Pettie, Wang and Zhan[3], *RNCD* n unknown	$O(n^{o(1)})$	$O(\log\log^{*} n)$ messages of size $O(\log n)$		$1-\frac{1}{n}$
Our results				
RNCD and *BN*, n unknown Upper bound u of n known	$O(\log^{1+\varepsilon} n)$	$O(1)$ 1-bit messages		$1-\frac{1}{\log^{O(1)} n}$
RNCD and *BN*, n unknown Upper bound u of n known	$O(n^{1-\varepsilon})$	$O(1)$ 1-bit messages		$1-\frac{1}{n^{1-\varepsilon}}$

2 Leader election when n is known by all nodes

For the sake of clarity, we start by the basic scenario where all the nodes know n and we assume that n is large. Our goal is to find a discrete probability distribution X verifying properties *(∗)* and *(⊙)* of Section 1.2. To find such a distribution, we remark that

$$\int_a^b \frac{Ke^{-\sqrt{x}}}{2\sqrt{x}} e^{-Ke^{-\sqrt{x}}} dx = \left[e^{-Ke^{-\sqrt{x}}}\right]_a^b = e^{-Ke^{-\sqrt{b}}} - e^{-Ke^{-\sqrt{a}}}. \tag{1}$$

Thus by choosing suitable values for a and b (say $a = \frac{1}{4}\log^2 K$ and $b = 3\log^2 K$), we will find that $\int_a^b \frac{Ke^{-\sqrt{x}}}{2\sqrt{x}} e^{-Ke^{-\sqrt{x}}} dx \geq 1 - O(K^{-c})$ for some $c > 0$. Let X be a positive discrete *r.v.* Throughout this paper, let $P_k = \mathbb{P}[X = k]$ for $k \geq 0$. Observe that if $X_1, X_2, \cdots, X_K$ are K independent copies of X, the probability that the maximum of the X_i is unique is given by

$$\sum_{k=1}^{\infty} \binom{K}{1} \mathbb{P}[X = k] \times \mathbb{P}[0 \leq X \leq k-1]^{K-1} = \sum_{k=1}^{\infty} K\, P_k \left(\sum_{i=0}^{k-1} P_i\right)^{K-1}. \tag{2}$$

We aim to find a probability distribution X such that for all values of k,

$$P_k \geq \frac{e^{-\sqrt{k}}}{2\sqrt{k}} \qquad \text{and} \qquad \sum_{i=0}^{k-1} P_i \geq e^{-e^{-\sqrt{k}}}, \tag{3}$$

$$\text{then,} \qquad \sum_{k=1}^{\infty} K\, P_k \left(\sum_{i=0}^{k-1} P_i\right)^{K-1} \geq \sum_{k} \frac{K\, e^{-\sqrt{k}}}{2\sqrt{k}}\, e^{-Ke^{-\sqrt{k}}}, \tag{4}$$

where in the last summation, k varies in some precise interval. According to the observation (1), we can then apply Euler-MacLaurin summation formula to show that the maximum of the X_i's is unique with probability at least $O\left(1 - \frac{1}{K^c}\right)$.

Definition 1 (Definition of the distribution.)**.** *We define P_k as follows*

$$P_0 = \frac{1}{e^{\frac{1}{e}}} \text{ and for all } k > 0,\ P_k = e^{-e^{-\sqrt{k+1}}} - e^{-e^{-\sqrt{k}}}. \tag{5}$$

[4]Single-hop RN Strong-CD is a much stronger model than RNCD because both transmitting and listening nodes can check the status of the common channel.

Lemma 1. *P_k described by (5) defines a discrete r.v. and satisfies:*
(±) For all $k \geq 0$, $P_k \geq 0$.
(∘)$\sum_{k=0}^{\infty} P_k = 1$.

$$(\triangle) \qquad \text{For all } k > 0, \qquad \sum_{j=0}^{k-1} P_j \geq e^{-e^{-\sqrt{k}}}. \tag{6}$$

$$(\star) \qquad \text{For all } k, \qquad P_k \geq \frac{e^{-\sqrt{k}}}{2\sqrt{k}} - \frac{e^{-\sqrt{k}}}{7k}. \tag{7}$$

Proof. The full proof of Lemma 1 is given in appendix A.1. □
The following observation is crucial for our purpose.

Lemma 2 (Uniqueness of the maximum)**.** *For $1 \leq j \leq K$, Let (X_j) be K independent copies of r.v. distributed as described by (5). Then,*

$$\textbf{(a)} \qquad \mathbb{P}\left[\text{Card}\{l \text{ such that } X_l = \max_{1\leq j\leq K} X_j\} = 1\right] \geq 1 - \frac{1}{\log^{O(1)} K},$$

$$\textbf{(b)} \mathbb{P}\left[(\log K - \log\log\log K)^2 \leq \max_{1\leq j\leq K} X_i \leq (\log K + \log\log K)^2\right] \geq 1 - O\left(\frac{1}{\log K}\right),$$

(c) Let N_{I} be the number of nodes having $X_s \in \mathrm{I} = [(\log K - \log\log\log K)^2, (\log K + \log\log K)^2]$,

$$\mathbb{P}[N_{\mathrm{I}} \geq 2\log\log K] \leq O\left(\frac{1}{\log^{\frac{1}{3}} K}\right).$$

Proof. The full proof of Lemma 2 is given in appendix A.2.

Remark 1. *We defied P_k in (5) with $\sqrt{k}$ but it can be generalized by replacing $\sqrt{k}$ with $k^{\frac{1}{1+\alpha}}$, for any $\alpha \in]0, 1]$. This does not affect Lemma 2(a) but changes the interval containing the maximum in Lemma 2(b) to $[(\log K - \log\log\log K)^{1+\alpha}, (\log K + \log\log K)^{1+\alpha}]$ which is of length at most $O(\log^{1+\alpha} K)$.*

2.1 Beeping Networks Model

Our algorithms are designed to make each node s aware of its status$(s) \in \{$Leader, Eliminated$\}$. For the sake of ease, any node s having $status(s) =$ Eliminated is designed as Eliminated node. We do the same for all intermediate status. Let us denote by I the interval containing the maximum of the generated *r.v.* The length of I is denoted $|\,\mathrm{I}\,|$.

Description of the algorithm: Our algorithm is subdivided into 5 steps
(i) Generating a total of $\lceil \frac{e\sqrt{n}}{n} \rceil$ *r.v.* to satisfy the *w.h.p.* requirement.
(ii) Separating Candidate from Eliminated nodes.
(iii) Using Eliminated nodes as independent witnesses.
(iv) Eliminating all the Candidate nodes that not have $X_s \in \mathrm{I}_{max}$ (I_{max} is the part of I containing the (unique) maximum of all generated *r.v.*)
(v) Eliminating all the Candidate nodes that have X_s different from the maximum and designating the leader. We can now explain these steps in more details.

(i) At the beginning, each node s generates locally $\lceil \frac{e\sqrt{n}}{n} \rceil$ random integers X_{s_j} $(1 \leq j \leq \lceil \frac{e\sqrt{n}}{n} \rceil)$ according to the distribution P_k (5) and sets $X_s = \max_{1\leq j\leq \lceil \frac{e\sqrt{n}}{n} \rceil} X_{s_j}$.

(ii) For the $\lceil \frac{e^{\sqrt{n}}}{n} \rceil$ *r.v.*, $\mathrm{I} = [(\sqrt{n} - \log\log n)^2, (\sqrt{n} + \log n)^2]$ *w.h.p.* by Lemma 2*(b)*. Similarly, $\max_{1 \leq j \leq \lceil \frac{e^{\sqrt{n}}}{n} \rceil} \left(\max_{1 \leq s \leq n} X_{s_j}\right) \in \mathrm{I}$ *w.h.p.* by Lemma 2*(a)*. Observe that the interval I can be computed by all the nodes without any communication. Then, each node having $X_s \in \mathrm{I}$ sets its status to
$status(s) =$ CANDIDATE while the other nodes set $status(s) =$ ELIMINATED. Note that $|\mathrm{I}| \sim 2\sqrt{n}\log n$.

(iii) I is subdivided into $\lceil 2\log n \rceil$ parts of equal size, say, I_i
$(1 \leq i \leq \lceil 2\log n \rceil)$. Each ELIMINATED node randomly chooses to enter into the $\lceil 2\log n \rceil$ groups $G_1, ...G_{\lceil 2\log n \rceil}$ corresponding to the intervals $\mathrm{I}_1, \mathrm{I}_2, ... \mathrm{I}_{\lceil 2\log n \rceil}$. As the total number of *r.v.*s generated by the nodes satisfies
$K \leq O(e^{\sqrt{n}})$, by Lemma 2*(c)* the total number of CANDIDATE nodes is at most $O(\log n)$ *w.h.p.* Thus, roughly $O\left(\frac{n}{\log n}\right)$ nodes will be assigned to any group G_j and will serve as witnesses of what will happen during the time slots belonging to the interval I_j. Note that steps ***(i)***, ***(ii)*** and ***(iii)*** do not require any communication between the nodes.

(iv) All the nodes search $\mathrm{I}_{\max}$ "scanning" distributedly the intervals
$\mathrm{I}_1, \mathrm{I}_2, \cdots, \mathrm{I}_{\lceil 2\log n \rceil}$. At the first round t_0, CANDIDATE nodes having $X_s \in \mathrm{I}_{\lceil 2\log n \rceil}$ wake up and beep. The ELIMINATED nodes in groups $G_{\lceil 2\log n \rceil}$ and $G_{\lceil 2\log n \rceil - 1}$ wake up to listen to the network (as they serve as witnesses for the corresponding intervals). At time t_1 ($t_1 = t_0 + 1$), the ELIMINATED nodes that heard beep at t_0 beep and the ELIMINATED nodes in groups $G_{\lceil 2\log n \rceil - 1}$ and $G_{\lceil 2\log n \rceil - 2}$ wake up to listen to the network. Then, as for t_0 and t_1, all ELIMINATED nodes browse the groups two by two from $\{G_{\lceil 2\log n \rceil - 2}, G_{\lceil 2\log n \rceil - 3}\}$ to $\{G_2, G_1\}$ during the rounds t_2 to $t_{\lceil 4\log n \rceil - 1}$. It is important to note that up to now, each node wakes up at most 2 times. In parallel, all CANDIDATE nodes also browse all parts of I two by two from $\{\mathrm{I}_{\lceil 2\log n \rceil}, \mathrm{I}_{\lceil 2\log n \rceil} - 1\}$ to $\{\mathrm{I}_2, \mathrm{I}_1\}$ during the same rounds. During each round $t_j, 0 \leq j \leq \lceil 4\log n \rceil - 1$, let the current pair of groups (*resp.* part of I) be $\{G_c, G_{c-1}\}$ (*resp.* $\{\mathrm{I}_c, \mathrm{I}_{c-1}\}$). An ELIMINATED node s wakes up only if its group $G(s) \in \{G_c, G_{c-1}\}$. Similarly, a CANDIDATE node s wakes up only if its random value $X_s \in (\mathrm{I}_c \cup \mathrm{I}_{c-1})$. We also consider the rounds two by two from $\{t_0, t_1\}$ to $\{t_{\lceil 4\log n \rceil - 2}, t_{\lceil 4\log n \rceil - 1}\}$. Each pair of rounds corresponds to one part of the interval I (*resp. one group*) such that $\{t_0, t_1\}$ corresponds to $\mathrm{I}_{\lceil \log n \rceil}, \mathrm{I}_{\lceil \log n \rceil - 1}$ (*resp.* $G_{\lceil \log n \rceil}, G_{\lceil \log n \rceil - 1}$) and so on. For all k such that $0 \leq k \leq \lceil 2\log n \rceil - 1$, during any pair of rounds $\{t_{2k}, t_{2k+1}\}$,
- a CANDIDATE node s that has $G(s) = G_c$ and did not hear beep during $\{t_{2(k-1)}, t_{2(k-1)+1}\}$ knows that it is in I_{max}. s beeps at t_{2k+1}.
- an ELIMINATED node v that has $G(v) = G_c$ (recall that G_c is the first current group) and heard beep during the previous pair of rounds $\{t_{2(k-1)}, t_{2(k-1)+1}\}$ beeps at t_{2k} to notify the nodes on the next group that I_{max} has been found.
- a CANDIDATE node s that has $G(s) = G_{c-1}$ (recall that G_{c-1} is the second current group) listens to the network at t_{2k} to verify if I_{max} has already been found. If s hears beep at t_{2k}, s sets
$status(s) \leftarrow$ ELIMINATED. Otherwise, s beeps at t_{2k+1}.
- an ELIMINATED node v that has $G(v) = G_{c-1}$ listens to the network at t_{2k} to verify if I_{max} has already been found before.

For the sake of simplicity, we defined TESTLISTEN$(k, T, |T|, |\mathrm{I}_c|, \mathrm{I})$ and TESTSEND$(k, T, |T|, |\mathrm{I}_c|, \mathrm{I})$ to do the computation in one pair of rounds $\{t_{2k}, t_{2k+1}\}$ as follows.

TESTLISTEN$(k, T, |T|, |\mathrm{I}_c|, \mathrm{I})$ works exactly in two synchronous rounds where all ELIMINATED nodes in the current group and all CANDIDATE nodes having *r.v.* $X_s \in$ current part of I wake up to listen to the network. Nodes hearing beep set $status(s) =$ MARKED, after that, all awake nodes go back to sleeping state.

TESTSEND$(k, T, |T|, |\mathrm{I}_c|, \mathrm{I})$ works in two rounds where all ELIMINATED nodes in current group and all CANDIDATE nodes having *r.v.* $X_s \in$ current part of I wake up. On the first round, CANDIDATE nodes beep. On the second round, all MARKED nodes beep. After that, all awake

nodes go back to sleeping state.
For the sake of clarity, Let take a simple example. Supposing that $\mathrm{I} = \{\mathrm{I}_1, \mathrm{I}_2, ... \mathrm{I}_{15}\}$, all ELIMINATED nodes are in one of these 15 groups $G_1, G_2, ...G_{15}$. At any pair of steps $\{t_{2k}, t_{2k+1}\}$, let say $k = 3$ corresponding to $\{G_{12}, G_{11}\}$ and $\{\mathrm{I}_{12}, \mathrm{I}_{11}\}$, all nodes compute in parallel TESTSEND$(3, G, 15, |I_{12}|, \mathrm{I})$ and TESTLISTEN$(3, G, 15, |\,\mathrm{I}_{12}\,|, \mathrm{I})$. For a better understanding, we illustrate this example in Appendix B.
At the end of these $O(\log n)$ rounds, only the CANDIDATE nodes having $X_s \in \mathrm{I}_{max}$ remains (I_{max} is of length $O(\sqrt{n})$ by Lemma 2*(b)*).

v) All ELIMINATED nodes choose to enter one of $\lceil\sqrt{n}\rceil$ groups $Z_1, Z_2...Z_{\lceil\sqrt{n}\rceil}$. All nodes do the same computation as in previous step by replacing G by Z and each part I_i of I by each value in I_{max}. At the end of these $\lceil\sqrt{n}\rceil$ rounds, all MARKED nodes set $status(s) =$ ELIMINATED and the remaining CANDIDATE becomes the LEADER.
We are now ready to give some technical details concerning our protocols. The following Lemma states that there is no empty group on line 8 of Algorithm 3.

Lemma 3. *By distributing all the* ELIMINATED *nodes (line* 8 *of Algorithm 3) into* $c \log n$ *groups, where* c *is a positive constant such that* $c \geq 2$

$$\mathbb{P}[\textit{There is an empty group}] \leq e^{-\left(\frac{n-\log n}{16c\log n}\right)}.$$

Proof. In few words, there are $O(n/\log n)$ ELIMINATED nodes entering uniformly at random $O(\log n)$ groups. The detailed proof of Lemma 3 is given in appendix A.3.
Like in the previous Lemma, the next technical result shows that there is no empty group Z_i on line 16 of Algorithm 3(see also ***(v)*** of Section 2.1).

Lemma 4. *By distributing all the* ELIMINATED *nodes (on line* 16 *of Algorithm 3) into* n^α *groups, where* $\alpha \in]0, 1[$,

$$\mathbb{P}[\textit{There is an empty group}] \leq O\left(\frac{1}{e^{\frac{1}{8}n^{1-\alpha}}}\right).$$

Proof. The full proof of Lemma 4 is given in appendix A.4

Theorem 1. *In single-hop beeping networks of large size* n*, if* n *is known in advance by all the nodes, there is a randomized Monte-Carlo leader election algorithm that elects a leader in* $O(\sqrt{n})$ *time slots with probability at least* $1 - O\left(\frac{1}{\sqrt{n}}\right)$ *during which no nodes are awake for more than* 4 *time slots.*

Proof. The full proof of Theorem 1 is given in appendix A.5.

Algorithm 1. TestListen$(k, T, |T|, |\mathrm{I}_c|, \mathrm{I})$

Input : Each node s with status$(s) \in \{$Candidate, Eliminated$\}$, the current pair of rounds number k, the group name T, the number of groups $|T|$, the size of the part of interval $|\mathrm{I}_c|$, the interval to scan I
Output: Each node s with status$(s) \in \{$Candidate, Eliminated, Marked$\}$.

Each node in group $T_{|T|-k-1}$ wakes up at t_{2k} and t_{2k+1}
if $|\mathrm{I}_c| > 1$ **then**
  Each Active node s having $X_s \in \mathrm{I}_{|T|-k-1}$ wakes up at t_{2k}
else
  Each Active node s having $\{X_s\} = \mathrm{I}_{|T|-k-1}$ wakes up at t_{2k}
end
for *each awake node* **s do**
  if *s hears beep at time slot* t_{2k} *or* t_{2k+1} **then**
    s sets *status*(s) = Marked
  end
  s goes back to sleeping state
end

Algorithm 2. TestSend$(k, T, |T|, |\mathrm{I}_c|, \mathrm{I})$

Input : Each node s with status$(s) \in \{$Marked, Candidate, Eliminated$\}$,the current pair of rounds number k, the group name T, the number of groups $|T|$, the size of the part of interval $|\mathrm{I}_c|$, the interval to scan I
Output: Each node s with status$(s) \in \{$Candidate, Eliminated$\}$.

Each node in group $T_{|T|-k}$ wakes up at t_{2k} and t_{2k+1}
if $|\mathrm{I}_c| > 1$ **then**
  Each Active node s having $X_s \in \mathrm{I}_{|T|-k}$ wakes up at t_{2k}
else
  Each Active node s having $\{X_s\} = \mathrm{I}_{|T|-k}$ wakes up at t_{2k}
end
for *each awake node* **s do**
  if *status*(s) = Candidate **then**
    s beeps at t_{2k}
  end
  if *status*(s) = Marked **then**
    s beeps at t_{2k+1}
    set *status*(s) = Eliminated
  end
  s goes back to sleeping state
end

We note that the time complexity and the probability of success only depend on the number of random values we choose to generate locally on each node. So, given $\varepsilon \in]0, 1[$ by choosing to generate $\lceil \frac{e^{n^\varepsilon}}{n} \rceil$ instead of $\lceil \frac{e^{\sqrt{n}}}{n} \rceil$ *r.v.* on each node, we have the following result.

Theorem 2. *In single-hop beeping networks of large size n and for any $\varepsilon \in]0, 1[$, if n is known in advance by all the nodes, there is a randomized Monte-Carlo leader election algorithm that elects a leader in $O(n^\varepsilon)$ time slots with probability at least $1 - O\left(\frac{1}{n^\varepsilon}\right)$ during which no nodes are awake for more than 4 time slots.*

3 Leader election when no node knows n

3.1 Beeping Networks Model

It is more realistic to assume that no node knows their exact number but an polynomial upper bound u of n is given in advance to all the nodes. As in previous section, the main idea is to generate $\lceil e^{u^\varepsilon} \rceil$ copies of *r.v.* following the distribution defined by (5) for any $\varepsilon \in]0,1[$. Since the nodes do not know the real value of n, and u is of the form n^c ($c > 1$ being a constant), it is possible that during the execution of Algorithm 3, the random choice (line 16) lead to some empty groups. To avoid such a situation (*w.h.p.*), we have to set properly the value of ε.

Algorithm 3. LeaderElection(n)

Input : The number of nodes n
Output: Each node s with a status(s) $\in$ {Leader, Eliminated}.

1 Each node s generates $\lceil \frac{e\sqrt{n}}{n} \rceil$ r.v. X_{s_j} with $\mathbb{P}[X_s = k] = P_k$ given by (5).
2 Each node s sets $X_s = \max_{1 \le j \le \lceil \frac{e\sqrt{n}}{n} \rceil} X_{s_j}$ and $\mathrm{I} = [\lfloor (\sqrt{n} - \log\log n)^2 \rfloor, \lceil (\sqrt{n} + \log n)^2 \rceil]$
3 **for** ***each node*** **s do**
4 **if** $X_s \in I$ **then**
5 s sets $status(s)$ = Candidate ,subdivides I into $\lceil 2\log n \rceil$ parts
6 **else**
7 s sets $status(s)$ = Eliminated,
8 randomly chooses to enter group G_i, i from 1 to $\lceil 2\log n \rceil$
9 **end**
10 **for** i *from* 0 *to* $\lceil 2\log n \rceil - 1$ **do**
11 s executes TestSend($i, G, \lceil 2\log n \rceil, \lceil \sqrt{n} \rceil$, I) and TestListen($i, G, \lceil 2\log n \rceil, \lceil \sqrt{n} \rceil$, I) and remaining Candidate nodes know I_{max}
12 **end**
13 **if** $status(s)$ = Candidate **then**
14 s subdivides I_{max} into $\lceil \sqrt{n} \rceil$ parts
15 **else**
16 s randomly chooses to enter group Z_i, i from 1 to $\lceil \sqrt{n} \rceil$
17 **end**
18 **for** i *from* 0 *to* $\lceil \sqrt{n} \rceil - 1$ **do**
19 s executes TestSend($i, Z, \lceil \sqrt{n} \rceil, 1, \mathrm{I}_{max}$) and TestListen($i, Z, \lceil \sqrt{n} \rceil, 1, \mathrm{I}_{max}$)
20 **end**
21 **if** $status(s)$ = Candidate **then**
22 s sets status(s) = Leader
23 **end**
24 **end**

To do so, we define TestEmpty(j) as a protocol that finds if there is an empty group when the nodes choose to enter $\lceil n^\varepsilon \rceil$ groups. After that, we define the FindGroups(ε, u) algorithm as follows. All nodes execute the following loop until all groups are non empty.
- Each node s chooses randomly to enter a group h_i, i from 1 up to $\lceil u^\varepsilon \rceil$.
-All nodes call TestEmpty(j) for j varying from $\lceil u^\varepsilon \rceil$ down to 1 setting some nodes to be Marked.
- All Marked nodes beep at the end of this execution and if there is an empty group (at least one Marked node beeping), all nodes divide ε by 2, set $status(s)$ = Null and redo all the execution.
The following lemma shows the time complexity and energy complexity of Algorithm FindGroups(u, ε).

Lemma 5. *Algorithm* FindGroups(ε, u) *terminates after* $O(u^\varepsilon)$ *rounds during which no nodes are awake for more than* $O(1)$ *rounds. It assigns all nodes into* $\lceil u^\varepsilon \rceil$ *non empty groups and gives a common right value* ε *to the nodes.*

Proof. The full proof of Lemma 5 is given in appendix A.6.

Algorithm 4. TestEmpty(j)

Input : The round number j
Output: Each node s with status$(s) \in \{$Marked, Null$\}$.

```
1  for Each node s in group Z_{⌈u^ε⌉−j+1} do
2  |  s wakes up and beep at t_j
3  end
4  for Each node s in group Z_{⌈u^ε⌉−j} do
5  |  s wakes up and listen at t_j
6  |  if s does not hear a beep then
7  |  |  set status ← Marked
8  |  end
9  end
10 Each awake node goes back to sleep
```

Algorithm 6. FindGroups(ε, μ)

Input : A common value $\varepsilon \in]0, 1[$, upper bound u of n
Output: Each node s having one group g and common right value ε

```
1  Each node sets ε = 2ε /⋆ to avoid the loop to start with ε/2 ⋆/
2  do
3  |  Each node sets ε = ε/2
4  |  Each node s choose randomly to join a group h_i, i from 1 up to ⌈μ^ε⌉
5  |  for i from 1 to ⌈μ^ε⌉ do
6  |  |  All nodes execute TestEmpty(i)
7  |  end
8  |  All Marked nodes beep
9  while ∃Marked nodes
```

Having these two new procedures, we can now design a leader election algorithm when no node knows n but all nodes know a common upper bound $u > n$. To do so, we adapt Algorithm 3 as follows.

- Before generating $\lceil e^{u^\varepsilon} \rceil$ r.v. X_{s_j} with $\mathbb{P}[X_s = k] = P_k$ given by (5) (line 1 of Algorithm 3), all nodes call **Algorithm 6.** FindGroups(ε, μ) to calculate the suitable common value of ε such that there is no empty group when all nodes choose to enter $\lceil u^\varepsilon \rceil$ groups in line 16 of Algorithm 3.
- After that, all nodes do as in Algorithm 3 by replacing $\sqrt{n}$ by u^ε.
- Arriving at line 16 of Algorithm 3, instead of choosing to enter $\lceil u^\varepsilon \rceil$ groups, all nodes already have one of the $\lceil u^\varepsilon \rceil$ non empty groups found by calling FindGroups(ε, μ). With these modifications, the following results holds.

Theorem 3. *In single-hop beeping networks of large size* n*, if* n *is unknown by the nodes but a polynomial upper bound* u *of* n *is given in advance to all the nodes, there is a randomized Monte-Carlo leader election algorithm that elects a leader in* $O(u^c)$ *time slots with probability at least* $1 - O\left(\frac{1}{u^c}\right)$ *during which no nodes are awake for more than* $O(1)$ *times slots (*c *being a positive constant).*

3.2 Leader Election in RNCD

It is more challenging to design algorithms working under the BN model which is clearly weaker than the RNCD model. It is easy to adapt Algorithm 3 for the RNCD model. Indeed in this case, we have to redesign TestSend$(k, T, |T|, |\mathrm{I}_c|, \mathrm{I})$ and TestListen$(k, T, |T|, |\mathrm{I}_c|, \mathrm{I})$ to output Single when exactly one node sends 1-bit messages and Collision when more than one node transmit. Since the uniqueness (or not) of the node(s) holding the maximum of the *r.v.*s can be tested, Algorithm 3 can be adapted to obtain a Las Vegas protocol for the RNCD model. We then have the following important result.

Theorem 4. *In single-hop RNCD of large size n, if n no node knows n but all nodes have a common polynomial upper bound u of n, there is a randomized Las Vegas leader election algorithm that elects a leader with probability at least $1 - O\left(\frac{1}{u^c}\right)$ in $O(u^c)$ time slots, with no nodes awake for more than $O(1)$ time slots sending and receiving 1-bit messages (c is a positive constant).*

We can improve this time complexity by generating one *r.v.* in each node. However, this implies a weaker probability of success. Thus, by remark 1, there node holding the maximal *r.v.* is unique *w.Lh.p* and the following result holds.

Theorem 5. *In single-hop RNCD of large size n, if no node knows n but all nodes have a common polynomial upper bound u of n, there is a randomized Las Vegas leader election algorithm terminating with probability at least $1 - O\left(\frac{1}{\log^{1+\alpha} u}\right)$ in $O(\log^{1+\alpha} n)$ time slots, during which no nodes are awake more than $O(1)$ time slots sending and receiving 1-bit messages (α is a positive constant).*

Conclusion

To reach optimal energy complexity on leader election algorithm in single hop networks, we designed an algorithm based on the uniqueness of the maximum of $\lceil e^{n^\varepsilon} \rceil$ independent copies of discrete *r.v.* Its time complexity depends only on the rounds spent to find this maximum value. Given $\varepsilon > 0$, on beeping networks, we designed a randomized Monte-Carlo algorithm electing a leader with probability at least $1 - O\left(\frac{1}{n^\varepsilon}\right)$ in $O(n^\varepsilon)$ times, with no nodes awake for more than $O(1)$ time slots. This can be adapted to have a randomized Las Vegas leader election algorithm, terminating after $O(n^\varepsilon)$ rounds with probability at least $1 - \frac{1}{n^\varepsilon}$, having energy complexity $O(1)$, sending and receiving $1 - bit$ message. Our algorithms work even if n is not known by all nodes but an polynomial upper bound u of n is given in advance to all the nodes.

References

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

[2] Capetanakis, J.: Tree algorithms for packet broadcast channels. IEEE Trans. on Information Theory **25**(5), 505 – 515 (1979)

[3] Chang, Y.J., Kopelowitz, T., Pettie, S., Wang, R., Zhan, W.: Exponential separations in the energy complexity of leader election. In: Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing. pp. 771–783. ACM (2017)

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

[5] Cornejo, A., Kuhn, F.: Deploying wireless networks with beeps. In: International Symposium on Distributed Computing. pp. 148–162 (2010)

[6] Dubhashi, D., Panconesi, A.: Concentration of measure for the analysis of randomized algorithms. Cambridge (2009)

[7] Ghaffari, M., Haeupler, B.: Near optimal leader election in multi-hop radio networks. In: Proceedings of the twenty-fourth annual ACM-SIAM symposium on Discrete algorithms. pp. 748–766 (2013)

[8] Ghaffari, M., Lynch, N., Sastry, S.: Leader election using loneliness detection. Distributed Computing **25**(6), 427–450 (2012)

[9] Greenberg, A.G., Winograd, S.: A lower bound on the time needed in the worst case to resolve conflicts deterministically in multiple access channels. Journal of the ACM **32**(3), 589 – 596 (1985)

[10] Jurdziński, T., Kutyłowski, M., Zatopiański, J.: Energy-efficient size approximation of radio networks with no collision detection. In: International Computing and Combinatorics Conference. pp. 279–289 (2002)

[11] Kardas, M., Klonowski, M., Pajak, D.: Energy-efficient leader election protocols for single-hop radio networks. In: Parallel Processing (ICPP), 2013 42nd International Conference on. pp. 399–408. IEEE (2013)

[12] Luc, D.: Non-Uniform Random Variate Generation. Devroye's web page (2003), `http://www.nrbook.com/devroye/`

[13] Metcalfe, R., Boggs, D.: Ethernet: Distributed packet switching for local computer networks. Communications of the ACM **19**(7), 395–404 (1976)

[14] Nakano, K., Olariu, S.: Randomized leader election protocols in radio networks with no collision detection. In: International Symposium on Algorithms and Computation. pp. 362–373 (2000)

[15] Nakano, K., Olariu, S.: A survey on leader election protocols for radio networks. In: International Symposium on Parallel Architectures, Algorithms and Networks. I-SPAN'02. pp. 71–76. IEEE (2002)

[16] Nakano, K., Olariu, S.: Uniform leader election protocols for radio networks. IEEE Trans. on Parallel Distrib. Syst. **13**(5), 516 – 526 (2002)

[17] Tsybakov, B.S.: Free synchronous packet access in a broadcast channel with feedback. Problems Inform. Transmission **14**(4), 259 – 280 (1978)

[18] Willard, D.: Log-logarithmic selection resolution protocols in a multiple access channel. SIAM Journal on Computing **15**(2), 468–477 (1986)

Appendix

A Lemmas and Theorems proofs

A.1 Proof of lemma 1

(±) By (5), $P_0 > 0$.
For all $k > 0$, since $\frac{\partial}{\partial k} e^{-e^{-\sqrt{k}}} > 0$, $e^{-e^{-\sqrt{k}}}$ is increasing in k and

$$P_k = e^{-e^{-\sqrt{k+1}}} - e^{-e^{-\sqrt{k}}} > 0\,. \tag{8}$$

For part *(∘)*, we have

$$\sum_{k=0}^{\ell} P_k = P_0 + \sum_{k=1}^{\ell} P_k = e^{-e^{-\sqrt{\ell+1}}}\,, \tag{9}$$

as the sum has telescoped. Then,

$$\sum_{k=0}^{\infty} P_k = \lim_{\ell=\infty} \sum_{k=0}^{\ell} P_k = \lim_{\ell=\infty} e^{-e^{-\sqrt{\ell+1}}} = 1\,. \tag{10}$$

(△) We can prove this by replacing ℓ in (9) by $k-1$
(⋆) For sufficiently large k, we have,

$$P_k = \frac{e^{-\sqrt{k}}}{2\sqrt{k}} - \frac{e^{-\sqrt{k}}}{8k} - \frac{5e^{-\sqrt{k}}}{45k^{\frac{3}{2}}} + O\left(\frac{e^{-\sqrt{k}}}{k^2}\right) \geq \frac{e^{-\sqrt{k}}}{2\sqrt{k}} - \frac{e^{-\sqrt{k}}}{7k}\,. \tag{11}$$

□

A.2 Proof of Lemma 2

Let $M_K = \text{Card}\{l \text{ such that } X_l = \max_{1\leq j\leq K} X_j\}$ be the the number of *r.v.* that are the maximal,
(a) Applying Lemma 1 *(iii)* to (2), for some $p > 0$ and $q < \infty$,

$$\begin{aligned}\mathbb{P}[M_K = 1] &\geq K \sum_{k=p}^{q} \left(\frac{e^{-\sqrt{k}}}{2\sqrt{k}} - \frac{e^{-\sqrt{k}}}{7k}\right)\left(e^{-e^{-\sqrt{k}}}\right)^K \\ &\geq \left(\sum_{k=p}^{q} K\frac{e^{-\sqrt{k}}}{2\sqrt{k}} e^{-Ke^{-\sqrt{k}}}\right) - \left(\sum_{k=p}^{q} K\frac{e^{-\sqrt{k}}}{7k} e^{-Ke^{-\sqrt{k}}}\right)\,. \end{aligned} \tag{12}$$

Let $\phi(x) = K\frac{e^{-\sqrt{x}}}{2\sqrt{x}} e^{-Ke^{-\sqrt{x}}}$. By Euler-Maclaurin formula, we have

$$\sum_{k=p}^{q} \phi(k) = \frac{\phi(p)+\phi(q)}{2} + \int_p^q \phi(x)dx + \int_p^q \phi'(x)(x - \lfloor x\rfloor - \frac{1}{2})dx\,.$$

Since $\phi'(k)$ is not monotone, and $k - \lfloor k\rfloor - \frac{1}{2} \geq -\frac{1}{2}$, there exists some value $C < \log^2 K$ (C is of form $(1-\theta)\log^2 K$ where $\theta > 0$) verifying

for all $k \in [\frac{1}{\beta}\log^2 K, C]$

$$\phi'(k) \times (k - \lfloor k \rfloor - \frac{1}{2}) \geq \phi'(k) \times (-\frac{1}{2}) \,.$$

Then

$$\int_p^q \phi'(x)(x - \lfloor x \rfloor - \frac{1}{2})dx \geq \int_p^C \phi'(x)(-\frac{1}{2})dx \geq -\frac{\phi(C)}{2} + \frac{\phi(p)}{2} \,.$$

By Euler-Maclaurin summation formula and taking $p = \frac{1}{4}\log^2 K$ and $q = 4\log^2 K$, we have

$$\sum_{k=p}^{q} K \frac{e^{-\sqrt{k}}}{2\sqrt{k}} e^{-Ke^{-\sqrt{k}}} \geq \left(\int_{\frac{1}{4}\log^2 K}^{4\log^2 K} K \frac{e^{-\sqrt{x}}}{2\sqrt{x}} e^{-Ke^{-\sqrt{x}}} dx \right) - \frac{\phi(C)}{2} \,. \tag{13}$$

Using (1), we obtain

$$\sum_{k=p}^{q} K \frac{e^{-\sqrt{k}}}{2\sqrt{k}} e^{-Ke^{-\sqrt{k}}} \geq \left(e^{-\frac{1}{K}} - e^{-K^{\frac{1}{2}}} \right) - \frac{K^{1-\sqrt{1-\theta}}}{2\sqrt{1-\theta}\log K e^{K^{1-\sqrt{1-\theta}}}} \,.$$

Since $e^{-x} > (1-x)$ and $\theta > 0$,

$$\sum_{k=p}^{q} K \frac{e^{-\sqrt{k}}}{2\sqrt{k}} e^{-Ke^{-\sqrt{k}}} \geq 1 - \frac{1}{K} \,. \tag{14}$$

As $-\frac{2}{7\sqrt{k}}$ is increasing in k,

$$-\sum_{k=p}^{q} K \frac{e^{-\sqrt{k}}}{7k} e^{-Ke^{-\sqrt{k}}} \geq -\frac{2}{7\sqrt{p}} \sum_{k=p}^{q} K \frac{e^{-\sqrt{k}}}{2\sqrt{k}} e^{-Ke^{-\sqrt{k}}} \,.$$

Then, by taking p and q as defined in (13) and using (14),

$$-\sum_{k=p}^{q} K \frac{e^{-\sqrt{k}}}{7k} e^{-Ke^{-\sqrt{k}}} \geq -\frac{1}{\log K}\left(1 - \frac{1}{K}\right) \,. \tag{15}$$

Thus, by applying (14) and (15) to (12), we reach the desired result.

(b) By independance all the X_i are less than $(\log K + \log\log K)^2$ with probability

$$\left(\mathbb{P}\left[X_1 \leq (\log K + \log\log K)^2\right]\right)^K = \left(e^{-e^{-\log K - \log\log K}}\right)^K \geq 1 - O\left(\frac{1}{\log K}\right) \,.$$

There is at least one X_i greater than $(\log K - \log\log\log K)^2$ with probability

$$1 - \left(\mathbb{P}\left[X_1 \leq (\log K - \log\log\log K)^2\right]\right)^K = 1 - \left(e^{-e^{-\log K + \log\log\log K}}\right)^K = 1 - O\left(\frac{1}{\log K}\right) \,.$$

(c)

$$\mathbb{P}[(\log K - \log\log\log K)^2 \leq X_s \leq (\log K + \log\log K)^2] = \sum_{k=(\log K - \log\log\log K)^2}^{(\log K + \log\log K)^2} P_k$$

$$= e^{-e^{-\log K - \log\log K}} - e^{-e^{-\log K + \log\log\log K}} = e^{-\frac{1}{K\log K}} - e^{-\frac{\log\log K}{K}} \,.$$

For any sufficiently large x, we have :

$$-e^{-\frac{\log\log x}{x}} = \frac{\log\log x}{x} - 1 - \frac{\log\log^2 x}{2x^2} + O\left(\frac{\log\log^3 x}{x^3}\right)$$

And

$$e^{-\frac{1}{x\log x}} = 1 - \frac{1}{x\log x} + O\left(\frac{1}{x^2\log^2 x}\right).$$

Then, we obtain

$$\mathbb{P}[(\log K - \log\log\log K)^2 \leq X_s \leq (\log K + \log\log K)^2] = \frac{\log\log K}{K} - \frac{1}{K\log K} + O\left(\frac{1}{K^2\log^2 K}\right),$$

and

$$\mathbb{E}[N_{\mathrm{I}}] \sim \log\log K - \frac{1}{\log K}.$$

As a consequence, by Chernoff bound (see for example [6, Theorem 1.1]),

$$\mathbb{P}[N_{\mathrm{I}} \geq 2\log\log K] \leq e^{-\frac{1}{3}\log\log K} \leq O\left(\frac{1}{\log^{\frac{1}{3}} K}\right).$$

□

A.3 Proof of Lemma 3

Let $|G|$ be the number of nodes assigned to one group G. Applying Lemma 2*(c)* to $K = e^{\sqrt{n}}$, the number of ELIMINATED nodes on line 8 of Algorithm 3 is at most $n - \log n$

$$\mathbb{E}[|G|] = \left(\frac{n - \log n}{c\log n}\right).$$

By Chernoff bound, we have

$$\mathbb{P}[|G| \leq \left(\frac{n - \log n}{2c\log n}\right)] \leq e^{-\left(\frac{n-\log n}{16c\log n}\right)}.$$

□

A.4 Proof of Lemma 4

Let $|g|$ be the number of nodes assigned to one group g. Applying Lemma 2*(c)* to $K = e^{\sqrt{n}}$, the number of ELIMINATED nodes is at most $n - \log n$

$$\mathbb{E}[|g|] = \left(n^{1-\alpha} - \frac{\log n}{n^\alpha}\right).$$

The, by Chernoff bound, we have

$$\mathbb{P}[|g| \leq \frac{1}{2}\left(n^{1-\alpha} - \frac{\log n}{n^\alpha}\right)] \leq O\left(\frac{1}{e^{\frac{1}{8}n^{1-\alpha}}}\right).$$

□

A.5 Proof of Theorem 1

The running time of algorithm 3 is the sum of time complexity of TestSend$(k, T, |T|, |\mathrm{I}_c|, \mathrm{I})$ and TestListen$(k, T, |T|, |\mathrm{I}_c|, \mathrm{I})$ times $2\log n$, and time complexity of TestSend$(i, T, |T|, |\mathrm{I}_c|, \mathrm{I})$ and TestListen$(k, T, |T|, |\mathrm{I}_c|, \mathrm{I})$ times $\sqrt{n}$.
Since TestSend$(k, T, |T|, |\mathrm{I}_c|, \mathrm{I})$ and TestListen$(k, T, |T|, |\mathrm{I}_c|, \mathrm{I})$ has same time complexity (equals 2), the running time of this algorithm is $4\log n + 2\sqrt{n} = O(\sqrt{n})$.
The probability of success depends on the probability of uniqueness of the maximum value of $\lceil \frac{e\sqrt{n}}{n} \rceil$ copies of X with $\mathbb{P}[X_s = k] = P_k$ given by (5)and the probability of defining the right interval of this max which are $1 - O\left(\frac{1}{\sqrt{n}}\right)$ by Lemma 2. The success of this algorithm also depends on the probability that all created groups on line 8 and 16 of Algorithm 3 are non empty, given by Lemma 3 and Lemma 4.
Finally, the total waking time is sum of the energy complexity of TestSend$(k, T, |T|, |\mathrm{I}_c|, \mathrm{I})$ and TestListen$(k, T, |T|, |\mathrm{I}_c|, \mathrm{I})$ which is $2 + 2 = 4$.

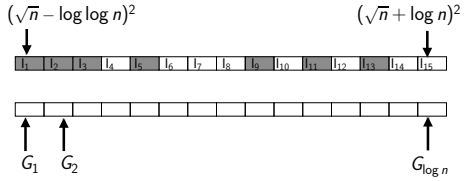
□

A.6 Proof of Lemma 5

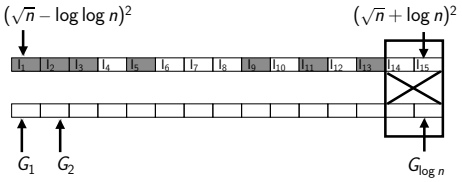
Since $u = n^\beta$ (β being constant), the time complexity of Algorithm FindGroups(ε, u) is at most $(\log \beta) \times u^\varepsilon = O(u^\varepsilon)$ rounds. And energy complexity is, as in Theorem 1, at most $(\log \beta) \times O(1) = O(1)$.
□

B Figures illustrating the execution of Algorithm 3

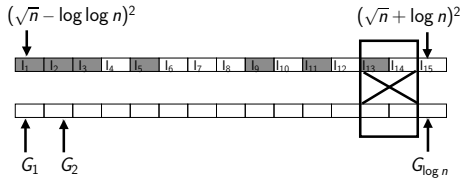
In the following figures, the first bare represents interval $\mathrm{I} = [(\sqrt{n} - \log\log n)^2, (\sqrt{n} + \log n)^2]$. We can see that it is subdivided into 15 parts. The light Grey cases are the parts of I in which any Candidate nodes have *r.v.* X_s. The second bare represents the $\lceil 2\log n \rceil$ groups in which all Eliminated nodes choose to enter. The kind of black sliding window represents the algorithm browsing all parts of I and all groups two by two during each round. The dark Grey cases represent the groups containing Marked nodes. We explain the execution of Algorithm 3 inside the figures below.



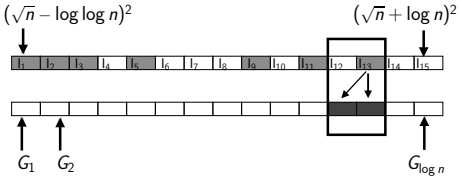
The initial configuration of the example, we suppose that $|\,\mathrm{I}_i\,| = 40$ for all $1 \leq i \leq 15$



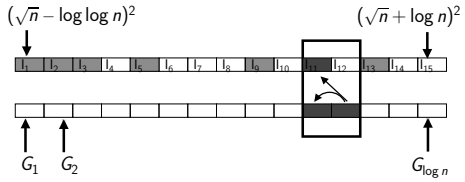
$\{t_0, t_1\}$: All nodes call TestSend$(0, G, 15, 40, \mathrm{I})$ and TestListen$(0, G, 15, 40, \mathrm{I})$. All Candidate nodes having $X_s \in \mathrm{I}_{15} \cup \mathrm{I}_{14}$ and all Eliminated nodes in group G_{15} and G_{14} wake up.



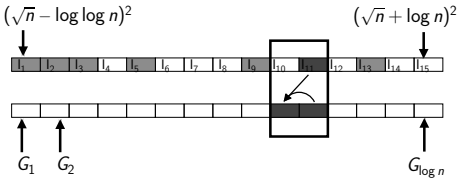
$\{t_2, t_3\}$: As in $\{t_0, t_1\}$, no node beeps



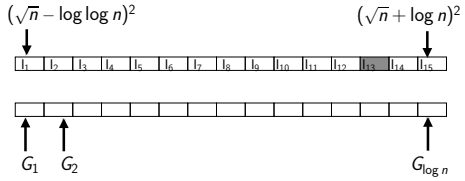
$\{t_4, t_5\}$: Candidate nodes having $X_s \in \mathrm{I}_{13}$ beep and all Eliminatednodes in groups G_{13}, G_{12} hear beep and get Marked



$\{t_6, t_7\}$: All Marked nodes in group G_{12} beep. Candidate nodes having $X_s \in \mathrm{I}_{11}$ and Eliminated nodes in group G_{11} wake up, hear beep and get Marked



$\{t_8, t_9\}$: All Marked nodes in group G_{11} and Marked nodes having $X_s \in \mathrm{I}_{11}$ beep. Eliminated nodes in group G_{10} wake up, hear beep and get Marked



At the end, all Candidate nodes that are not in the part of I containing the maximal of $r.v.$ are eliminated