Transmitting Once to Elect a Leader on Wireless Networks

ANDRIAMBOLAMALALA Ny Aina and RAVELOMANANA Vlady

IRIF UMR CNRS 8243— University of Paris — France
ny-aina.andriambolamalala@irif.fr
vlad@irif.fr

Abstract. Distributed wireless network's devices are battery-powered most of the time and transmitting a message uses more energy than receiving one which spends more energy than internal computations. Therefore in this paper, we will focus on the energy complexity of leader election, a fundamental problem in distributed computing. As the message's size impacts on the energy consumption, we highlight that our algorithms have almost optimal time complexities: Each device is allowed to send only once $1-bit$ message and to listen to the network during at most 2 time slots. We will firstly work on Radio Networks on which the devices can detect when a node transmits alone: RNstrongCD where both senders and receivers have collision detection capability, RNsenderCD, RNreceiverCD and RNnoCD. If the nodes know their number n, our algorithm elects a leader in optimal $O(\log n)$ time slots with a probability of $1 - 1/\operatorname{poly}(n)$.Then, if all nodes do not know n but know a common value u such that $\log n = \Theta(\log u)$, it has $O(\log^2 n)$ time complexity on RNnoCD and RNsenderCD. For RNreceiverCD and RNstrongCD, it has $O(\log^{1+\alpha} n)$ time complexity where $\alpha \in]0, 1[$ is constant. For the Beeping Networks model on which the devices cannot detect single transmissions, our algorithm has $O(n^{\alpha})$ time complexity with probability $1-1/\operatorname{poly}(n)$.

1 Introduction

Distributed leader election problem has been extensively studied over the years [11, 12, 17, 20, 21, 27]. It consists in all the n devices of the distributed system, denoted $s_1, s_2, \ldots, s_n$, agreeing on one device to be the leader in a decentralized manner. The study of its energy complexity gained in importance with the design of Low-power Wireless Sensor Devices [14, 24, 26]. On such devices, transmitting uses more energy than listening to the network which spends more energy than internal computations [3, 17, 24, 26]. For example, in [24], each device consumes respectively 1.8 Watts, 0.6 W and 0.05 W when transmitting, receiving a message and having radio switched off. Such energy consumption also depends on the collision detection capability [6, 17] and the size of messages that a device can send [2]. Recall that the energy complexity is the maximum over all devices of the time slot number during which any device is awake[1]. In this paper, we

[1] When it transmits or listens to the network.

design algorithms during which each node exchanges only a single bit message, transmits at most once and listens to the network during at most two time slots. This is in contrast with the works in [4, 11, 22] where the messages have $O(\log n)$ or even larger size [6, 10]. We found some similarities on our model and a distributed real-time communication model (Time Triggered Protocols) on which a unique sending slot is assigned to each node [18, 23]. This latter model was widely used for designing energy-efficient algorithms on Wireless Sensor Networks [1, 15]. We consider the single-hop[2] Networks defined below, the basic ingredient out of which larger multi-hop networks are built [22, 27].

Single-hop Radio Networks or RN. Introduced by Chlamtac and Kutten in the 80's [7], communications occur in synchronous time slots. At any time slot, a node independently decides whether to transmit, to listen to the network, or to remain idle or asleep. The network can have four status: SINGLE if exactly one node transmits, NULL if no node transmits and COLLISION if at least two nodes transmit. Only the SINGLE transmissions are received by the listening nodes. We consider four models. On RNstrongCD, both transmitters and listeners have collision detection capability and on RNsenderCD, only transmitters can detect collision. On RNreceiverCD or RNCD, only listeners can detect collision but transmitters can not and on RNnoCD, no device has collision detection.

Single-hop Beeping Networks or BN. This was introduced in 2010 by Cornejo and Kuhn [8] and is strictly weaker than the RN models as far as the message length and the collision detection capability are concerned [2]. It makes little demands on the devices which need only be able to do carrier-sensing as well as differentiating between a silence and the presence of a jamming signal on the network. The communications occur synchronously as in RN but the transmitting nodes cannot detect collisions and the listening nodes cannot distinguish between single and more beeps emitted by their neighbors.

To use randomness, we assume that the devices can generate discrete random variables (see for instance Devroye [19]). We also assume that these devices are initially anonymous and indistinguishable and can do any internal computations [6, 21]. They cannot communicate on the network when in a sleeping state. However, they can choose to wake up or to sleep at any time slot. All presented algorithms in this paper succeed with high probability[3] or *w.h.p.* for short.

1.1 Related works

Considering single-hop synchronous RNCD networks, in the late 70's, Tsybakov [25] and Capetanakis [5] designed deterministic leader election algorithms terminating in $O(\log n)$ time slots. Such algorithms are optimal since Greenberg and Winograd [13] have established a lower bound of $\Omega(\log n)$ on the time complexity for deterministic algorithms when all nodes know n. On the randomized side, Willard [27] designed protocols working in expected $O(\log \log n)$ and in $O(\log n)$ time slots under the high probability requirement when the nodes do not know

[2] The underlying graph of the network is complete.

[3] An event ε_n occurs *w.h.p.* if $\mathbb{P}[\varepsilon_n] \geq 1 - n^{-c}$ where c is a positive constant.

n. Given an error rate ε, Nakano and Olariu [22] provided a randomized algorithm terminating in $O(\log\log n) + o(\log\log n) + O(\log 1/\varepsilon)$ time slots with a probability exceeding $1-\varepsilon$. They also provided a lower bound of $\Omega(\log n)$ for uniform leader election protocols. Ghaffari, Lynch and Sastry [12] extended this lower bound to all protocols. Given an upper bound u of n to all the nodes, they presented a lower bound of $\Omega(\min\{\log(u/n), \log(1/\varepsilon)\})$ for leader election algorithms succeeding with probability greater than $1-\epsilon$. Amongst other results, Kardas, Klonowski and Pająk [17] designed a leader election algorithm for the RNstrongCD model where n is unknown, succeeding in $O(\log^{\epsilon} n)$[4] expected time slots with $O(\log\log\log n)$ energy complexity. When the nodes know $\Theta(n)$, Jurdziński, Kutyłowskiowski, and Zatopiański [16] designed an algorithm with $O(\log^* n)$[5] energy complexity and $O(\log n)$ time complexity on RNCD. Bender, Kopelowitz, Pettie and Young [4] then gave an $O(\log(\log^* n))$ upper bound for the energy complexity of leader election and approximate counting in the RNCD model when n is unknown by the nodes. In [6], Chang, Kopelowitz, Pettie, Wang and Zhan presented a leader election protocol for RNreceiverCD (*resp.* RNnoCD) with $n^{o(1)}$ time complexity and $O(\log\log^* n)$ energy complexity (*resp.* $O(\log^* n)$) when no node knows n. Amongst several important results, they proved a $\Omega(\log\log^* n)$ (*resp.* $\Omega(\log^* n)$) lower bound for the energy complexity of leader election in RNCD (*resp.* RNnoCD) for this setting.

1.2 Our results

The following table shows what differentiate our results from the existing results. Our algorithm's design stands out from regular randomized algorithms. It is

Existing results				
Assumptions	Model	Time	Energy	Probability
n known	RNCD [17]	$O(\log n)$	$O(\log\log\log n)$	$1-O(1/n)$
$\Theta(n)$ known	RNCD [16]	$O(\log n)$	$O(\log^* n)$	$1-O(1/n)$
n unknown	RNCD, RNnoCD [6]	$O\left(n^{o(1)}\right)$	$O(\log^* n)$	$1-O(1/n)$
n unknown	RNstrongCD, RNsenderCD	$O\left(n^{o(1)}\right)$	$O(\log\log^* n)$	$1-O(1/n)$
Our results				
n known	RNCD, RNnoCD	$O(\log n)$	3	$1-O(1/n)$
n known	RNstrongCD, RNsenderCD Section 2.1	$O(\log n)$	2	$1-O(1/n)$
n unknown	RNnoCD,	$O(\log^2 n)$	3	$1-O(1/n)$
and	RNsenderCD,Section 2.2	$O(\log^2 n)$	2	$1-O(1/n)$
$\Theta(\log n)$	$BN\ \alpha \in]0,1[$	$O(n^{\alpha/(\alpha+1)}\times$	2	$1-$
known	Section 3	$\log n)$		$O(n^{-\alpha/(\alpha+1)})$

based on the nodes locally generating random values before communicating on

[4] $\log^{\varepsilon} n = (\log n)^{\varepsilon}$ for any constant ε.
[5] $\log^* n$ represents the iterated logarithm of n.

the network in a deterministic way. Our algorithms have energy complexity of no more than 3 where each node transmits at most once and listens to the network during at most 2 time slots. As commonly assumed, the IDs of the nodes fit in $O(\log n)$ bits, then u is such that $\log n = \Theta(\log u)$ *i.e.* $u \in]n, n^c]$ where $c > 1$. Note that when n is known, the presented algorithm in Section 2.1 is optimal in view of both time ($\Theta(\log n)$) [22] and energy complexities [17] (see Appendix B.2).The shown result in [6] can be adapted to work on the setting considered in Section 2.2, with the same $O(\log^2 n)$ time complexity, $O(1)$ energy complexity and $O(\log n \log\log n)$ messages size (more details are given in Appendix B.5). The best previous result on this scenario was the $O(\log n)$ time complexity with $O(\log^* n)$ energy presented in [16] when all nodes know $\Theta(n)$.

2 Radio Networks

2.1 The nodes initially know the exact value of n

We start by assuming that all the nodes initially know the exact value of n and remembering that in the RNnoCD, only the listening nodes can differentiate from single, no transmitter or multiple transmitters. We take advantage of this ability to simulate loneliness detection [12] in such a model. Our goal is to cause the following events to occur during the execution of such an algorithm:
-(i) If $t_0 = 0$ is the initial time slot, there is a time slot $t_g = t_0 + g$ when exactly one node s_1 transmits alone while a set S_z of nodes listens to the network.
-(ii) Then, exactly one second node $s_2 \in S_z$ transmits alone at $t_g + 1$ (while s_1 listens to the network) to notify s_1 that it was elected. Thus, s_2 is the unique witness of the probable election of s_1. To fulfill this goal, our algorithm is based on each node locally generating random values and communicating on the network in a deterministic manner, to find out two consecutive unique[6] values. Therefore, we make each node generate one copy of a discrete random variable (r.v. for short) X such that if $X_1, X_2, \ldots, X_N$ are N independent copies of X:
$(\odot)$ There are 2 unique consecutive values X_i, X_{i+1} with a constant probability.
G(1/2)[7], the geometric distribution with parameter 1/2 respects $(\odot)$.

In the rest of the paper, we use $\lg a$ to denote the logarithm of a in base 2. We suppose that $\log a$, $\lg a$ and e^a are integers for any value a. Due to space constraint, all Theorem and Lemma's proofs are postponed in the Appendix A.

Lemma 1. *Let $X_1, X_2, \ldots, X_N$ be N independent copies of a r.v. distribution following* G(1/2). I *is a discrete interval of integers and* Card(I) *is the size of the interval* I. *We then have* $\mathrm{I} = \{\mathrm{I}_0, \mathrm{I}_1, \ldots \mathrm{I}_{\mathrm{Card(I)}-1}\}$ *where* $\mathrm{I}_r = \mathrm{I}_0 + r$ *is an integer. Let p be the probability that $\exists (i,j) \in [1,N]^2$ such that $X_i = \lg N, X_j = \lg N - 1$ and $X_l \notin \{\lg N - 1, \lg N\} \forall l \notin \{i,j\}$. We have*

$$p > \frac{1}{5}\left(1 - O\left(\frac{1}{N}\right)\right)$$

[6] A random value is said to be unique if it is held by exactly one node.
[7] If X is a r.v. distributed as G(1/2), $\mathbb{P}[X = k] = 2^{-k-1}$ for all $k \geq 0$.

Proof. Postponed in Appendix A.1. □

Overview of the algorithm: It works on RNnoCD and RNCD when each node knows n. In what follows, each node s_i has a status denoted Status(s_i), which can take one of the following values: Null is the initial status, Candidate if s_i is candidate to be the leader, Eliminated if s_i cannot be elected, Marked if s_i is temporarily marked to do some computations and Leader if s_i is the elected node. Any node s_i having Status(s_i) = Null is designated as a Null node and we do the same for all status. Each node is initially sleeping and is restricted to send a $1-bit$ message only once. Our algorithm is designed to make each node s_i aware of its final Status(s_i)$\in$ {Leader, Eliminated}.

Our main idea is to make each node s_i locally generate one random copy X_i of a r.v. X distributed as G(1/2). Then, all nodes *browse through*[8] the interval $\mathrm{I} = [\lg n - 1, \lg n]$, in order to find out which two of them have consecutive unique random values. For instance, by Lemma 1, a sequence of $X_1, X_2, \ldots, X_n$ with unique node s_1 (*resp.* s_2) holding $X_1 = \lg n - 1$ (*resp.* $X_2 = \lg n$) occurs with a constant probability. Thus such idea leads us to the election of s_1 with a constant probability in $O(\mathrm{Card(I)})$ time slots. Then, to reach the high probability requirement, we have to execute the latter described algorithm $O(\log n)$ times by keeping the energy complexity at a maximum of 3. To do so, our algorithm is subdivided into $2\log n + 1$ steps. All nodes are firstly distributed such that $O(n/\log n)$ nodes participate to each step and each node participates to only one step. During each such step, $O(n/\log n)$ nodes then do a leader election succeeding with a constant probability as described earlier.

Step 0: step's choice. Each node chooses uniformly at random in which step it will participate. Then, each step is subdivided into 3 Phases: candidacy, witnessing and browsing. Let S_z be the set of nodes participating to Step z, $\forall z > 0$.

Lemma 2. $\mathrm{Card}(S_z) \in [2n/5\log n, 3n/5\log n]$ *with a probability greater than* $1 - e^{-O(n/\log n)}$.

Proof. In Appendix A.2. □

For the sake of clarity, we describe the execution of Step 1 but this will be generalized for any Step z in the Algorithm 2. During the *candidacy* phase, each node in S_1 chooses to be Candidate or Eliminated. Then, on the *witnessing*[9] phase, each Eliminated node in S_1 chooses at which time slot of the *browsing* phase it will witness for the election of a node. Finally, during the *browsing* phase, all nodes in S_1 *browse through* I to elect a leader. I is defined by Lemma 1 by replacing N with $O(n/\log n)$. For greater clarity, we present Phase 3 before Phase 2.

Step 1 Phase 1: candidacy. At t_0, each node $s_i \in S_1$ locally generates one independent copy X_i of a r.v. X distributed as G(1/2). Based on Lemma 1, all nodes in S_1 having $X_i \in \mathrm{I} = [\lg(2n/5\log n) - 1, \lg(3n/5\log n)]$ then take the Candidate status and the other nodes of S_1 become Eliminated.

[8] At each time slot $t_0, t_1, \ldots, t_g$, each node s_i checks if the corresponding value I_g in the interval I is equal to its X_i, then transmits or does some computations at t_g.

[9] Listening to verify an election at the time slot.

Lemma 3. *There are* $O(\log n)$ CANDIDATE *nodes in each Step with a probability greater than* $1 - O(\log n/n)$.

Proof. In Appendix A.3. □

Step 1 Phase 3: browsing through I. This Phase uses an odd/even time slots scheduling. Even time slots $\{t_0, t_2, \ldots, t_{2g}\}$ are dedicated for transmissions and odd time slots $\{t_1, t_3, \ldots, t_{2g+1}\}$ are used for feedback. At t_0, each CANDIDATE node $s_i \in S_1$ checks if $X_i = \mathrm{I}_0$ s.t. $\mathrm{I}_0 = \lg(2n/5\log n) - 1$, then, transmits a $1 - bit$ message. Each S_1's CANDIDATE node s_i having $X_i = \mathrm{I}_1$ listens to the network. Then, at t_1, the nodes with $X_i = \mathrm{I}_0$ listen in their turn and if the listening nodes at t_0 received a message, they send a single bit feedback at t_1. A transmitting node at t_0 that receives the feedback at t_1 becomes LEADER and the other nodes become ELIMINATED. Each CANDIDATE node $s_i \in S_1$ executes these computations at each time slot $t_0, t_1, \ldots, t_g, \ldots, t_{2\,\mathrm{Card(I)}-1}$, checking if its $X_i = \mathrm{I}_g$ at t_g. It is possible to have several consecutive unique random values in the interval I, involving the election of multiple leaders. In order to bypass such a problem, we add the following Phase 2 before Phase 3.

Step 1 Phase 2: witnessing an election at a time slot and flooding the next rounds. After Phase 1, the $O(n/\log n)$ ELIMINATED nodes in S_1 (Lemma 2 and Lemma 3) are distributed to witness the probable election of a leader at each time slot of Phase 3. Let T be the time complexity of Phase 3. We have $\mathrm{T} = 2\,\mathrm{Card(I)} \leq 6$. At the round t_0, after executing Phase 1, each ELIMINATED node in S_1 chooses uniformly at random or UAR[10] one time slot t_w or time to witness from $\{t_0, t_2, \ldots, t_{T-2}\}$. So $t_w = \mathrm{UAR}(\{t_0, t_2, t_4\})$. These ELIMINATED nodes listen to the network at t_w and $t_w + 1$ during Phase 3. They receive messages at both points if a leader is elected. So, to avoid another leader election, each node chooses a time to flood[11] $t_f = \mathrm{UAR}(\{t_w + 2, \ldots, t_{(4\,\mathrm{Card(I)}\log n)-1}\})$ and transmits at t_f. By flooding all the remaining time slots, no other CANDIDATE node can transmit alone. The following figure illustrates the execution of one step of such algorithm whith 8 devices.

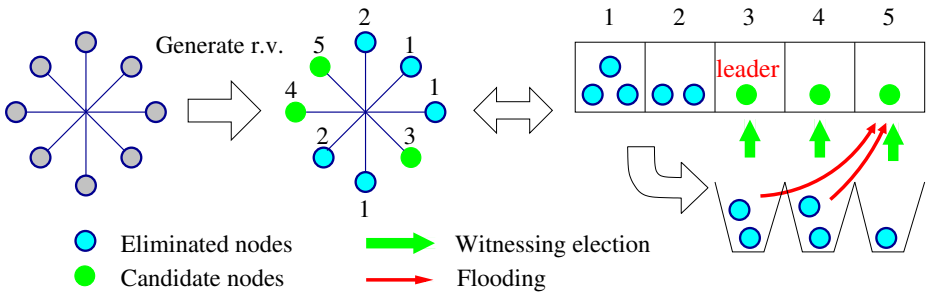

Fig. 1. Leader election succeeding with a constant probability for 8 devices and Card(I) = 3.

[10] UAR(B) return one value picked uniformly at random from the set B.
[11] Sending a message at the time slot if a leader has already been elected.

Algorithm 2. LeaderElection(n).

Input : The exact value of n.
Output: Each node s_i with a Status(s_i) $\in$ {Leader, Eliminated}.

Step 0: Each node enters a set UAR($\{S_1, S_2, \ldots, S_{2\log n}$) where S_z is the set of nodes that will participate in Step z.
Step 1 to Step 2 log n: **for** z *from* 1 *to* $2\log n$ **do**
  Step z Phase 1: Each node $s_i \in S_z$ locally generates a random value X_i distributed as G(1/2).
  if $X_i \in \mathrm{I} = [\lg(2n/5\log n) - 1, \lg(3n/5\log n)]$ **then**
    s_i sets Status(s_i) $\leftarrow$ Candidate.
  else
    s_i sets Status(s_i) $\leftarrow$ Eliminated.
  end
  Step z Phase 2: Each Eliminated node in S_z sets $t_w \leftarrow UAR(\{t_{(z-1)2\,\mathrm{Card(I)}}, t_{(z-1)2\,\mathrm{Card(I)}+2}, \ldots, t_{2z\,\mathrm{Card(I)}-2}\})$ and $t_f \leftarrow UAR(\{t_w + 2, \ldots, t_{(4\log n\,\mathrm{Card(I)})-1}\})$.
  Step z Phase 3: Each node runs the Browse(I) procedure.
  Each remaining Candidate node s_i sets $status(s_i) \leftarrow$ Eliminated.
end

Algorithm 1. Browse(I): called at Step z.

Input : Interval I.
Output: Each node s_i with Status(s_i) $\in$ {Leader, Eliminated}.

for g *from* 0 *to* Card(I) $-$ 1 **do**
  Each node sets $t = t_{2(z-1)\,\mathrm{Card(I)}+2g}$.
  Each Candidate node $s_i \in S_z$ with $X_i = \mathrm{I}_g$ sends $1 - bit$ at time slot t and listens at $t+1$.
  Each Candidate node $s_j \in S_z$ with $X_j = \mathrm{I}_{g+1}$ listens at t.
  if s_j *receives a message at* t **then**
    s_j transmits $1 - bit$ message at $t+1$.
  end
  if s_i *receives a message at* $t+1$ **then**
    s_i sets Status(s_i) $\leftarrow$ Leader.
  end
  Each Eliminated node $s_e \in S_z$ having $t_w = t$ listens at t and $t+1$.
  if s_e *receives a message at both* t *and* $t+1$ **then**
    s_e sets Status(s_e) $\leftarrow$ Marked.
  end
  Each Marked node that has $t_f = t$ transmits at t and sets Status(s_e) $\leftarrow$ Eliminated.
end

Lemma 4. *At least one node witnesses and at least one node floods during each time slot of the Algorithm 2 respectively with a probability greater than* $1 - e^{-O(n/\log n)}$ *and* $1 - e^{-O\left(n/\log^2 n\right)}$.

Proof. Postponed in Appendix A.4. □

Lemma 5. *During the execution of Algorithm 2, each node wakes up to transmit one bit during at most one time slot and listens to the network during at most two time slots.*

Proof. See Appendix A.5 for the detailed proof. □

Remark 1 *On the RNsenderCD and RNstrongCD models, each node can know when it transmits alone. Then, the other nodes do not have to notify the leader that it was elected. Thus, we can cause the* Candidate *nodes to never listen to the network and the* Eliminated *nodes to witness at only one time slot.*

Theorem 1. *In single-hop RNnoCD and RNCD (resp. RNstrongCD and RNsenderCD) networks of large size n, if all nodes know n, there is a randomized Monte-Carlo leader election algorithm that elects a leader in $O(\log n)$ time slots with a probability of at least $1 - O\left(n^{-1/3}\right)$. Each node transmits $1-$bit message no more than once and listens to the network for a maximum of two (resp. one) time slots.*

Proof. The full proof is given in Appendix A.6. □

Remark 2 *To simplify Algorithm 2, we made it run the* Browse(I) *protocol $2\log n$ times. Thus, it succeeds with probability $1 - O\left(n^{-1/3}\right)$. This can be improved to reach $1 - O\left(n^{-1}\right)$ by running* Browse(I) *$5\log n$ times. This remark can be applied to all Theorems on Radio Networks.*

2.2 The nodes do not know n

When the nodes do not know n but know its upper bound u such that $\log n = \Theta(\log u)$ *i.e.* $u \in]n, n^c]$ where $c > 1$, we adapt Algorithm 2 as follows. On Step 0, each node chooses UAR to participate in one of the remaining $2\log u$ Steps. Let S_z be the set of nodes participating in Step z. By Lemma 2, $\mathrm{Card}(S_z) \in [2n/5\log u, 3n/5\log u]$ *w.h.p.* Then on Phase 1 of each Step z, in order to have an interval containing the interval $\mathrm{I} = [\lg(2n/5\log u) - 1, \lg(3n/5\log u)]$, each node sets a new interval $\mathrm{J} = \left[\lg\left(2u^{1/c}/5\log u\right) - 1, \lg(3u/5\log u)\right]$. Then, on Phase 2, the nodes eliminated after Phase 1 set $t_w = \mathrm{UAR}(\{t_{(z-1)2\,\mathrm{Card(J)}}, t_{(z-1)2\,\mathrm{Card(J)}+2}, \ldots, t_{2z\,\mathrm{Card(J)}-2}\})$ and $t_f = \mathrm{UAR}(\{t_w+2, \ldots, t_{(4\log u\,\mathrm{Card(J)})-1}\})$. Finally, on Phase 3, all nodes run the Browse(J) protocol.

Theorem 2. *In single-hop RNnoCD and RNCD (resp. RNsenderCD and RNstrongCD) networks of large size n, if no node knows the exact value of n, but an upper bound u of n is given in advance to all the nodes, there is a randomized Monte-Carlo leader election algorithm succeeding in $O(\log^2 n)$ time slots with a probability greater than $1 - O\left(n^{-1}\right)$. Each node transmits during no more than one time slot and listens to the network during at most two (resp. one) time slots.*

Proof. The full proof is postponed in Appendix A.7. □

Remark 3 *The time complexity can be improved to be $O(\log^{1+\alpha} n)$ on RNCD and RNstrongCD, using the collision detection capability. We detail such improvement in Appendix B.1.*

3 Beeping Networks

In this section, we consider the BN model where neither a beeping node s_i nor listening nodes can detect if s_i beeps alone or not. The goal here is to make a node know that it beeped along *w.h.p.* without any feedback from the network. To do so, our main idea is based on the uniqueness of the maximum of n independent copies $Y_1, Y_2, \ldots, Y_n$ of the following new r.v.

Definition 1 (Definition of the distribution of the r.v. Y). *Throughout this paper, let $p_k = \mathbb{P}[Y = k]$ for all integers $k \geq 0$ defined for some $\alpha \in]0,1[$ as follows.*

$$p_0 = e^{-1} \text{ and } p_k = \exp\left(-k^{1/(1+\alpha)}\right) - \exp\left(-(k+1)^{1/(1+\alpha)}\right) \text{ for all } k > 0\,. \quad (1)$$

If such a maximum is unique with a probability p, we cause each node to generate a random copy of Y and our algorithm has to localize which node holds such a maximum. This latter node then becomes Leader.
The following observation is crucial for our purpose:

Lemma 6. *Let $Y_1, Y_2, \cdots, Y_N$ be N independent copies of a r.v. distributed as described by (1) and $m = \max_{1 \leq i \leq N}\{Y_i\}$*
(a) $\mathbb{P}\left[\text{Card}\{l \text{ such that } Y_l = m\} = 1\right] \geq 1 - O\left(1/\log^{\alpha} N\right)$.

(b) *Let* $\mu = \mathbb{P}\left[(\log N - \log\log\log N)^{1+\alpha} \leq m \leq (\log N + \log\log N)^{1+\alpha}\right]$. *We have*

$$\mu \geq 1 - O\left(1/\log N\right).$$

(c) *Set* $\mathrm{L} = [(\log N - \log\log\log N)^{1+\alpha}, (\log N + \log\log N)^{1+\alpha}]$,
and let q be the random variable $q = \text{Card}(\{l \text{ such that } Y_l \in \mathrm{L}\})$. Then,

$$\mathbb{P}[q \geq 3\log\log N] \leq O\left(\frac{1}{\log N}\right).$$

Proof. Postponed in Appendix A.8. □
The time complexities of our algorithms on BN, when the nodes know and do not know n are quite similar: $O(n^{\alpha/\alpha+1}\log n)$ and $O(n^{\alpha})$. See Append B.4 for more details. Thus, we immediately consider the case when the nodes do not know n.

3.1 The nodes do not know n

Each node knows an upper bound u of n such that $\log n = \Theta(\log u)$ *i.e.* $u \in]n, n^c]$ where $c > 1$ is known by the nodes. In order to reach the high probability requirement, we adapt Algorithm 2 to work on BN model with the following 3 phases. For better clarity, we present Phase 3 before Phase 2.

- **Phase 1:** Let $V = \exp\left(u^{\alpha/(c(\alpha+1))}\right)$. Based on Lemma 6 (a), each node s_i generates V random copies $Y_{i,1}, Y_{i,2}, \ldots, Y_{i,V}$ of a r.v. Y distributed as described by (1) and saves $Y_i = \max_{h=1,2,\ldots V}\{Y_{i,h}\}$. Then, according to Lemma 6 (b), with $N = nV$, each node computes $\mathrm{L}_0 = (\log V - \log\log\log V)^{1+\alpha}$ and $\mathrm{L}_{\mathrm{Last}} = (\log(uV) + \log\log V)^{1+\alpha}$. Each node s_i having Y_i in the interval of integers $\mathrm{L} = [\mathrm{L}_0, \mathrm{L}_{\mathrm{Last}}] = \{\mathrm{L}_0, \mathrm{L}_1, \ldots, \mathrm{L}_{\mathrm{Last}}\}$ such that $\mathrm{L}_r = \mathrm{L}_0 + r$, becomes CANDIDATE and the other nodes are ELIMINATED.
- **Phase 3:** Each CANDIDATE node *browses through* the interval L one value at a time as in the BROWSE(I) protocol but in reverse order from $\mathrm{L}_{\mathrm{Last}}$ to L_0 in order to find out which holds the maximum. This latter node becomes LEADER. Firstly, if a CANDIDATE node s_i has $Y_i = \mathrm{L}_{\mathrm{Last}}$, it becomes LEADER and beeps at t_0. At each time slot $t_0, t_1, \ldots, t_g$, each CANDIDATE node checks if $Y_i = \mathrm{L}_{\mathrm{Last}} - (g+1)$. If at any time slot t_g, a CANDIDATE node has $Y_i = \mathrm{L}_{\mathrm{Last}} - (g+1)$, it listens to the network at t_g. If it does not hear a beep at t_g, it beeps at t_{g+1} and becomes LEADER.

As some values in L may not be picked by any node *i.e.* there can be time slots during Phase 3 where node neither beeps nor listens to the network, the algorithm can elect more than one leader. To circumvent this problem, we introduce the following new witnessing procedure which consists of flooding all time slots after an election.

- **Phase 2:** All ELIMINATED nodes sets $t_w = UAR(\{t_0, \ldots, t_{\mathrm{Card(L)}-1}\})$. They will listen to the network at t_w on Phase 3. Then, if a node beeps at a time slot t_g of Phase 3, all nodes hearing a beep: the CANDIDATE nodes with $Y_i = \mathrm{L}_{\mathrm{Card(L)}} - (g+1)$ and the ELIMINATED nodes with $t_w = t_g$, have to beep at t_{g+1} in order to notify the next CANDIDATE nodes (which become ELIMINATED) that a LEADER has already been elected.

Due to space constraint, we present the Algorithm 3 in Appendix B.3.

Theorem 3. *Fix $\alpha \in]0,1[$, in single-hop BN networks of large size n, if no node knows n, but an 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 $O(n^\alpha)$ time slots with a probability of $1 - O\left(n^{-\alpha^2/\alpha+1}\right)$. Each node transmits and listens during a maximum of one time slot.*

Proof. We give detailed proof in Appendix A.9. □

Conclusion

We designed leader election algorithms taking into account their energy consumption and their time complexities while each device can transmit $1 - bit$ message once and can listen to the network during a maximum of 2 time slots. Our algorithm design is based on each node locally generating random values with a probability distribution and communicating in a deterministic manner on the network to find out which node has a unique value. The latter node becomes

the leader. The time complexity only depends on the time slots spent to localize such a node. Assuming that the nodes are initially indistinguishable and know n, our randomized algorithm terminates in optimal $O(\log n)$ time slots *w.h.p.* in the Radio Networks with and without collision detection. If a common value $\alpha \in]0,1[$ is given to all nodes, it has $O(n^{\alpha/\alpha+1})$ time complexity for the Beeping Networks. For the realistic case when the nodes do not know n, if a common upper bound u such that $\log n = \Theta(\log u)$ is given in advance to all the nodes, our algorithms terminate in $O(\log^2 n)$ time slots for the RN models and $O(n^\alpha)$ for BN. Some existing results can be adapted to reach $O(1)$ energy complexity on the models studied in this paper [4, 6] but we present the first results with each node transmitting at most once and listening to the network during at most two time slots, exchanging $1 - bit$ messages. Optimal energy complexity has been reached in [6] for the Radio Networks models when the nodes have no information about the topology of the network, but designing a polynomial time leader election for the BN model matching such lower bounds is open.

References

1. Affoua Thérese Aby, Alexandre Guitton, Pascal Lafourcade, and Michel Misson. Slack-mac: Adaptive mac protocol for low duty-cycle wireless sensor networks. In *International Conference on Ad Hoc Networks*, pages 69–81. Springer, 2015.
2. Yehuda Afek, Noga Alon, Ziv Bar-Joseph, Alejandro Cornejo, Bernhard Haeupler, and Fabian Kuhn. Beeping a maximal independent set. *Distributed computing*, 26(4):195–208, 2013.
3. Matthew Barnes, Chris Conway, James Mathews, and DK Arvind. Ens: An energy harvesting wireless sensor network platform. In *2010 Fifth International Conference on Systems and Networks Communications*, pages 83–87. IEEE, 2010.
4. Michael A Bender, Tsvi Kopelowitz, Seth Pettie, and Maxwell Young. Contention resolution with log-logstar channel accesses. In *Proceedings of the forty-eighth annual ACM symposium on Theory of Computing*, pages 499–508. ACM, 2016.
5. John I CAPETANAKIS. Tree algorithms for packet broadcast channels. *IEEE Trans. on Information Theory*, 25(5):505 – 515, 1979.
6. Yi-Jun Chang, Tsvi Kopelowitz, Seth Pettie, Ruosong Wang, and Wei Zhan. Exponential separations in the energy complexity of leader election. In *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing*, pages 771–783. ACM, 2017.
7. Imrich Chlamtac and Shay Kutten. On broadcasting in radio networks–problem analysis and protocol design. *IEEE Transactions on Communications*, 33(12):1240–1246, 1985.
8. Alejandro Cornejo and Fabian Kuhn. Deploying wireless networks with beeps. In *International Symposium on Distributed Computing*, pages 148–162, 2010.
9. Devdatt Dubhashi and Alessandro Panconesi. *Concentration of measure for the analysis of randomized algorithms*. Cambridge, 2009.
10. Pierre Fraigniaud, Amos Korman, and David Peleg. Towards a complexity theory for local distributed computing. *Journal of the ACM (JACM)*, 60(5):35, 2013.
11. Mohsen Ghaffari and Bernhard Haeupler. Near optimal leader election in multi-hop radio networks. In *Proceedings of the twenty-fourth annual ACM-SIAM symposium on Discrete algorithms*, pages 748–766, 2013.

12. Mohsen Ghaffari, Nancy Lynch, and Srikanth Sastry. Leader election using loneliness detection. *Distributed Computing*, 25(6):427–450, 2012.
13. Albert G Greenberg and Schmuel Winograd. 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.
14. Chunlong Guo, Lizhi Charlie Zhong, and Jan M Rabaey. Low power distributed mac for ad hoc sensor radio networks. In *GLOBECOM'01. IEEE Global Telecommunications Conference (Cat. No. 01CH37270)*, volume 5, pages 2944–2948. IEEE, 2001.
15. Yueshun He, Ping Du, Kun Li, and Shang Yong. An optimization algorithm based on the monte carlo node localization of mobile sensor network. *International Journal of Simulation–Systems, Science & Technology*, 17(20), 2016.
16. Tomasz Jurdziński, Mirosław Kutyłowski, and Jan Zatopiański. Weak communication in single-hop radio networks: adjusting algorithms to industrial standards. *Concurrency and Computation: practice and experience*, 15(11-12):1117–1131, 2003.
17. Marcin Kardas, Marek Klonowski, and Dominik Pająk. Energy-efficient leader election protocols for single-hop radio networks. In *Parallel Processing (ICPP), 2013 42nd International Conference on*, pages 399–408. IEEE, 2013.
18. Fan Liu, Ajit Narayanan, and Quan Bai. Real-time systems. 2000.
19. Devroye Luc. *Non-Uniform Random Variate Generation*. Devroye's web page, 2003.
20. Robert M Metcalfe and David R Boggs. Ethernet: Distributed packet switching for local computer networks. *Communications of the ACM*, 19(7):395–404, 1976.
21. Koji Nakano and Stephan Olariu. Randomized leader election protocols in radio networks with no collision detection. In *International Symposium on Algorithms and Computation*, pages 362–373. Springer, 2000.
22. Koji Nakano and Stephan Olariu. Uniform leader election protocols for radio networks. *IEEE transactions on parallel and distributed systems*, 13(5):516–526, 2002.
23. Hoon Oh and Trung-Dinh Han. A demand-based slot assignment algorithm for energy-aware reliable data transmission in wireless sensor networks. *Wireless networks*, 18(5):523–534, 2012.
24. Krishna M Sivalingam, Mani B Srivastava, and Prathima Agrawal. Low power link and access protocols for wireless multimedia networks. In *1997 IEEE 47th Vehicular Technology Conference. Technology in Motion*, volume 3, pages 1331–1335. IEEE, 1997.
25. Boris S Tsybakov. Free synchronous packet access in a broadcast channel with feedback. *Problems Inform. Transmission*, 14(4):259 – 280, 1978.
26. Marcos Augusto M Vieira, Claudionor N Coelho, DC Da Silva, and José Monteiro da Mata. Survey on wireless sensor network devices. In *EFTA 2003. 2003 IEEE Conference on Emerging Technologies and Factory Automation. Proceedings (Cat. No. 03TH8696)*, volume 1, pages 537–544. IEEE, 2003.
27. Dan Willard. Log-logarithmic selection resolution protocols in a multiple access channel. *SIAM Journal on Computing*, 15(2):468–477, 1986.

Appendix

A Detailed Proofs

A.1 Proof of Lemma 1

Let $X_1, X_2, \ldots, X_N$ be N independent copies of a random variable X following a geometric distribution with parameter 1/2, G(1/2). That is $q_k = \mathbb{P}[X_j = k] = 2^{-k-1}$. Our main idea is to prove that for N copies of G(1/2), with a strictly positive probability, there exists two consecutive unique values. Let p be the probability

$$\mathbb{P}[\exists (i,j) \in [1,N]^2 \text{ s.t. } X_i = \lg N, X_j = \lg N - 1, X_k \notin \{\lg N - 1, \lg N\} \forall k \notin \{i,j\}]. \tag{2}$$

By independence of the X_i's,

$$p = \sum_{k=\lg N-1}^{\lg N} 2\binom{N}{2} q_k q_{k-1} \left(1 - q_k - q_{k-1}\right)^{N-2}.$$

We have

$$p > (N^2 - N) \sum_{k=\lg N-1}^{\lg N} 2^{-(2k+1)} \left(1 - 3\left(2^{-(k+1)}\right)\right)^N.$$

Let $f(k) = 2^{-(2k+1)} \left(1 - 3\left(2^{-(k+1)}\right)\right)^N$. For any constant c, we have

$$f\left(\lg N + c\right) = 2^{-(2\lg N + 2c + 1)} \left(1 - 3\left(2^{-(\lg N + c + 1)}\right)\right)^N.$$

As

$$p > \left(N^2 - N\right) \left(f(\lg N - 1) + f(\lg N)\right),$$

a bit algebra leads to

$$p > \left(2e^{-3} + \frac{e^{-3/2}}{2}\right) \left(1 - O\left(\frac{1}{N}\right)\right).$$

Numerically, we have

$$2e^{-3} + \frac{e^{-3/2}}{2} \simeq 0.211 > \frac{1}{5},$$

so that,

$$p > \frac{1}{5} \left(1 - O\left(\frac{1}{N}\right)\right).$$

A.2 Proof of Lemma 2

If n nodes choose uniformly at random to enter into $2\log n$ groups, fix a group and let r be the random variable representing the number of nodes in such a group. We have $\mathbb{E}[r] = n/2\log n$. By means of a Chernoff bound (see for example [9, Theorem 1.1]),

$$\mathbb{P}\left[r \leq (1-\delta)\,\mathbb{E}[r]\right] < 1 - e^{-\mathbb{E}[r](\delta^2/2)}, \tag{3}$$

and

$$\mathbb{P}\left[r \geq (1+\delta)\,\mathbb{E}[r]\right] < 1 - e^{-\mathbb{E}[r](\delta^2/(2+\delta))}. \tag{4}$$

Then, by taking $\delta = 1/5$, we have

$$\mathbb{P}\left[r \leq \frac{2n}{5\log n}\right] < 1 - e^{-O(n/\log n)},$$

and

$$\mathbb{P}\left[r \geq \frac{3n}{5\log n}\right] < 1 - e^{-O(n/\log n)}.$$

A.3 Proof of Lemma 3

Recall that S_z is the set of nodes participating to the Step z in Algorithm 2. Let $X_1, X_2, \ldots, X_{\mathrm{Card}(S_z)}$ be $\mathrm{Card}(S_z)$ copies of a r.v. X distributed as G(1/2). We know that $\mathrm{Card}(S_z) \in [2n/5\log n, 3n/5\log n]$ *w.h.p.* by Lemma 2. We have

$$q = \sum_{k=\lg(2n/5\log n)-1}^{\lg(3n/5\log n)} q_k = \frac{5\log n}{n} - \frac{5\log n}{6n}.$$

If h is the random variable representing $\mathrm{Card}(\{i \text{ such that } X_i \in [2n/5\log n, 3n/5\log n]\})$, $\mathbb{E}[h] = \mathrm{Card}(S_z) \times q$ is constant and by means of a Chernoff bound, we reach the desired result.

A.4 Proof of Lemma 4

According to Lemma 2 and Lemma 3, after the Step 0 of Algorithm 2, there are at most $\log n$ CANDIDATE nodes and at least $2n/5\log n - \log n$ ELIMINATED nodes in each set S_z *w.h.p.* Then, those ELIMINATED nodes choose uniformly at random to witness one of the $2\,\mathrm{Card(I)}$ time slots of the step in which they participate (we have $2\,\mathrm{Card(I)} \leq 6$). Let l be a random variable representing the number of nodes witnessing each time slot. $\mathbb{E}[l] = O(n/\log n)$ and by means of a Chernoff bound,

$$\mathbb{P}\left[l < O\left(\frac{n}{\log n}\right)\right] < 1 - e^{-O(n/\log n)}. \tag{5}$$

As a consequence, at least $O(n/\log n)$ nodes (those witnessing the time slots corresponding to an election) are distributed to send a message during the remaining $O(\log n)$ time slots. In the same way as for (5), at least $O(n/\log^2 n)$ nodes are assigned to flood each time slot with probability greater than $1 - e^{-O(n/\log^2 n)}$.

A.5 Proof of Lemma 5

The CANDIDATE nodes in any set S_z only wake up during the Step z when they run the BROWSE(I) protocol. Let us consider the case of a CANDIDATE node s_1. It may only transmit at a time slot t_{2g} when it finds the value I_g in the interval I which is equal to X_1. Such a value I_g is then said to be under checking at t_{2g}.

s_1 wakes up and listens to the network twice when the values $\mathrm{I}_a, \mathrm{I}_b$ in I such that $\mathrm{I}_a = X_1 - 1$ and $\mathrm{I}_b = X_1 + 1$ are under checking. On the other hand, an ELIMINATED node s_2 wakes up and listens to the network exactly twice in Phase 3, at t_w and $t_w + 1$. It may also transmit once at a time slot equal to t_f, picked by s_2 in Phase 2, if it receives a message at both t_w and $t_w + 1$.

A.6 Proof of Theorem 1

By Lemma 1, a leader can be elected with a strictly positive constant probability by running the BROWSE(I) protocol once. Thus, the Algorithm 2 elects a leader *w.h.p.* in $O(\log n)$ time slots.
According to Lemma 4, a maximum of one leader is elected with a probability greater than $1 - e^{-O\left(n/\log^2 n\right)} \geq 1 - O\left(n^{-1}\right)$ for sufficiently large n. During its execution, each node wakes up during at most three time slots, transmitting once and listening to at most two time slots for the RNnoCD and RNCD (Lemma 5). Applying Remark 1 to Algorithm 2, we have each node listening at exactly on time slot for the RNstrongCD and RNsenderCD models.

A.7 Proof of Theorem 2

By Appendix A.6 and Remark 2, the presented adaptation of Algorithm 2 in Section 2.2 terminates in $4\,\mathrm{Card(J)}\log n = O(\log^2 n)$ rounds with probability greater than $1 - O\left(n^{-1}\right)$. Applying $\mathrm{Card(J)} = O(\log n)$ to Lemma 4, no more than one leader is elected with probability $1 - e^{-O\left(n/\log^4 n\right)} \geq 1 - O\left(n^{-1}\right)$ for large n. By Lemma 5, each node transmits during at most one time slot and listens twice to the network for the RNnoCD and RNCD. Applying Remark 1 to Algorithm 2, we have each node listening to exactly one time slot for the RNstrongCD and RNsenderCD models.

A.8 Proof of Lemma 6

Let $Y_1, Y_2, \ldots, Y_N$ be N independent copies of the r.v. Y distributed as described by (1) Let $m = \mathrm{Card}\{l \text{ such that } Y_l = \max_{1 \leq i \leq N} Y_i\}$ and $p = \mathbb{P}[m = 1]$. By independence of the Y_i's,

$$p = N \sum_{k=1}^{\infty} p_k \left(\sum_{i=0}^{k-1} p_i \right)^{N-1}.$$

Proof of (a) By the definition of p_k,

$$\sum_{i=0}^{k-1} p_i = 1 - \exp\left(-k^{1/(1+\alpha)}\right) > \exp\left(-\exp\left(-k^{1/(1+\alpha)}\right) - \exp\left(-2k^{1/(1+\alpha)}\right)\right),$$

and for k large enough,

$$p_k = \frac{\exp\left(-k^{1/(1+\alpha)}\right)}{(1+\alpha)k^{(\alpha/(1+\alpha))}} - O\left(\frac{\exp\left(-k^{1/(1+\alpha)}\right)}{k^{2\alpha/(1+\alpha)}}\right).$$

Let us set $\beta = 1/(1+\alpha)$.

For some large values a an b such that $0 \ll a < b < \infty$, we have

$$p > \sum_{k=a}^{b} A \times B \times C,$$

where

$$A = N\left(\frac{\beta \exp\left(-k^{\beta}\right)}{k^{1-\beta}} - O\left(\frac{\exp\left(-k^{\beta}\right)}{k^{\alpha^2\beta}}\right)\right),$$

$$B = \exp\left(-N\exp\left(-k^{\beta}\right)\right),$$

and

$$C = \exp\left(-N\exp\left(-2k^{\beta}\right)\right).$$

As $\exp\left(-N\exp\left(-2k^{\beta}\right)\right)$ is increasing in k,

$$p > \exp\left(-N\exp\left(-2a^{\beta}\right)\right) \sum_{k=a}^{b} A \times B\,. \tag{6}$$

Let us first consider $\sum_{k=a}^{b} A \times B$. Let

$$\phi(k) = N\frac{\beta\exp\left(-k^{\alpha}\right)}{k^{1-\beta}}\exp\left(-N\exp\left(-k^{\beta}\right)\right).$$

$$\sum_{k=a}^{b} A \times B = \sum_{k=a}^{b}\phi(k) - O\left(\sum_{k=a}^{b} N\frac{\beta\exp\left(-k^{\beta}\right)}{k^{\alpha^2\beta}}\exp\left(-N\exp\left(-k^{\beta}\right)\right)\right).$$

At this step, applying the standard Euler-Maclaurin formula and noting that

$$\int_r^s \phi(k)dk = \exp\left(-N\exp\left(-s^{\beta}\right)\right) - \exp\left(-N\exp\left(-r^{\beta}\right)\right),$$

we obtain

$$\sum_{k=a}^{b}\phi(k) \geq \int_r^s \phi(k)dk \geq \left(\exp\left(-N^{-\alpha}\right) - \exp\left(-N^{1/2}\right)\right).$$

Since $e^{-x} > (1-x)$ for all $x \in]0, \frac{1}{2}]$ and recalling that $\beta = 1/(1+\alpha)$,

$$\sum_{k=a}^{b} N \frac{\beta \exp\left(-k^{\beta}\right)}{k^{1-\beta}} \exp\left(-N \exp\left(-k^{\beta}\right)\right) \geq 1 - \frac{1}{N^{\alpha}}\,. \tag{7}$$

Secondly, as $-k^{-(1-\beta)}$ is increasing in k, we have

$$-O\left(\sum_{k=a}^{b} N \frac{\exp\left(-k^{\beta}\right)}{k^{2(1-\beta)}} \exp\left(-N \exp\left(-k^{\beta}\right)\right)\right) \geq -O\left(\frac{1}{a^{\beta}} \sum_{k=a}^{b} \phi(k)\right)\,.$$

Using (7) and taking $a = ((1-\alpha)\log N)^{1+\alpha}$, it yields

$$-O\left(\sum_{k=a}^{b} N \frac{\exp\left(-k^{\beta}\right)}{k^{2(1-\beta)}} \exp\left(-N \exp\left(-k^{\beta}\right)\right)\right) \geq -O\left(\frac{1}{\log^{\alpha} N}\left(1 - \frac{1}{N^{\alpha}}\right)\right)\,. \tag{8}$$

Thirdly, let us consider $\exp\left(-N \exp\left(-2a^{\beta}\right)\right)$ in (6). Fix $a = ((1-\alpha)\log N)^{1+\alpha}$,

$$\exp\left(-N \exp\left(-2a^{\beta}\right)\right) = \exp\left(-N \exp\left(-(2\alpha - 2)\log N\right)\right) > 1 - O\left(\frac{1}{n^{2\alpha-1}}\right)\,. \tag{9}$$

Finally, by applying (7), (8) and (9) to (6), we reach the desired result.

Proof of (b) By independence of the Y_i's, all the values $Y_1, Y_2, \cdots, Y_N$ are less than $(\log N + \log\log N)^{1+\alpha}$ with probability

$$q = \left(\mathbb{P}\left[Y_1 < (\log N + \log\log N)^{1+\alpha}\right]\right)^N = \left(1 - \exp\left(-\log N - \log\log N\right)\right)^N\,.$$

After some algebra, we obtain

$$q \geq 1 - O\left(\frac{1}{\log N}\right)\,.$$

In the same way, there is at least one value Y_i greater than $(\log N - \log\log\log N)^{1+\alpha}$ with probability

$$p = 1 - \left(\mathbb{P}\left[Y_1 < (\log N - \log\log\log N)^{1+\alpha}\right]\right)^N\,.$$

By definition of p_k,

$$p = 1 - \left(1 - \exp\left(-\log N + \log\log\log N\right)\right)^N = 1 - O\left(\frac{1}{\log N}\right)\,.$$

Proof of (c) Let $\psi = \mathbb{P}[(\log N - \log\log\log N)^{1+\alpha} \leq Y < (\log N + \log\log N)^{1+\alpha}]$. By the definition of p_k,

$$\psi = \frac{\log\log N}{N} - \frac{1}{N \log N}.$$

Then, we obtain $\mathbb{E}[\psi] = \log\log N - 1/\log N$. As a consequence, by means of a Chernoff bound,

$$\mathbb{P}[\psi \geq 3\log\log N] \leq e^{-\log\log N} \leq O\left(\frac{1}{\log N}\right).$$

A.9 Proof of Theorem 3

The time complexity of Algorithm 3 comes from the time spent in Phase 2 when all nodes *browse through* the interval L in reverse order. It takes Card(L) time slots which is $O\left(\log^{1+\alpha}(nV)\right) = O(u^{\alpha/c}) = O(n^{\alpha})$ where $V = \exp\left(u^{\alpha/(c(1+\alpha))}\right)$ for some constant $\alpha \in]0,1[$ and $\mathrm{L} = [(\log V - \log\log\log V)^{1+\alpha}, (\log(uV) + \log\log V)^{1+\alpha}]$.

The success probability of Algorithm 3 firstly depends on the probability that the maximum of all generated random values in Phase 1 is unique, which is greater than $1 - O\left(\log^{-\alpha}(nV)\right) = 1 - O\left(u^{-\alpha^2/(c(1+\alpha))}\right) = 1 - O\left(n^{-\alpha^2/(1+\alpha)}\right)$. It also depends on the probability that each time slot of the Phase 3 is witnessed by at least one node. By means of a Chernoff Bound, as by Lemma 6 (c), at least $n - \log n$ ELIMINATED nodes are distributed to witness each of the Card(L) $= n^{\alpha}$ time slots of Algorithm 3, this probability is of at least $1 - e^{-O(n^{1-\alpha})} \geq 1 - O\left(n^{-\alpha^2/(1+\alpha)}\right)$ for large n.

During the execution of Algorithm 3, a CANDIDATE node only transmits once during Phase 3 when it finds the value in L which is equal to its random value Y_s. In the same way, an ELIMINATED node transmits at most once in Phase 3 after its witnessed time slot t_w. All CANDIDATE nodes (*resp.* ELIMINATED) listen to the network during one time slot, when the value in L corresponding to $Y_i + 1$ is under checking (*resp.* at its witnessed time slot t_w).

B More about Algorithms

B.1 A faster algorithm for RNCD and RNstrongCD

On RNCD and RNstrongCD, as the listening nodes can detect collisions, we only need to make the property (i) given in Section 2.1 occur with a constant probability. So, we only need to have a unique node s_1 transmitting alone at any time slot t_g while a set S of nodes listen to the network. Then the listening nodes receive a message at t_g and send feedback to the unique sender at $t_g + 1$. s_1 listens at its turn at $t_g + 1$, detects a collision and knows that it transmitted alone. s_1 is then elected. Thus, instead of the property $(\odot)$ in Section 2.1, we need a r.v. distribution Z such that, if each node s_i generates one random copy Z_i of Z, there exists a unique value Z_j with a constant probability. For $\alpha \in]0,1[$, the following Lemma proves that the r.v. Z distributed as

$$\mathbb{P}[Z = k] = \exp\left(-k^{1/\alpha}\right) - \exp\left(-(k-1)^{1/\alpha}\right) \text{ verifies such a property} \quad (10)$$

and that the unique value is equal to $\log^\alpha n$.

Lemma 7. *Fix $\alpha \in]0,1[$, let $Z_1, Z_2, \cdots, Z_N$ be N independent copies of a r.v. Z distributed as described by (10). Let ν be the probability that there exists a value $Z_j = \log^\alpha N$ such that $\mathrm{Card}(Z_j) = 1$. We have*

$$\nu > \frac{1}{e}\left(1-\frac{1}{e}\right)\left(1-\frac{1}{N}\right).$$

Proof. There exists a value $Z_j = \log^\alpha N$ such that $\mathrm{Card}(Z_j) = 1$ with probability

$$\nu = \binom{N}{1}\mathbb{P}[Z = \log^\alpha N]\left(1-\mathbb{P}[Z=\log^\alpha N]\right)^{N-1}.$$

By the definition of $\mathbb{P}[Z=k]$ and with $\psi(N) = \exp(-\log N) - \exp\left(-(\log^\alpha N+1)^{1/\alpha}\right)$,

$$\nu = N\psi(N)\left(1-\psi(N)\right)^N.$$

As $(\log^\alpha N+1)^{1/\alpha} > \log N + 1$,

$$\nu > N\left(\frac{1}{N}-\frac{1}{eN}\right)\left(1-\frac{1}{N}\right)^N > \left(1-\frac{1}{e}\right)\frac{1}{e}\left(1-\frac{1}{N}\right).$$

□

When the nodes do not know n but know $u \in]n, n^c]$, we adapt Algorithm 2 to work on RNCD and RNstrongCD as follows. It is subdivided into $2\log u+1$ Steps.

Step 0: step choice. Each node chooses UAR to participate in one of the remaining $2\log u$ Steps, as a leader may be elected at each step with a constant probability by Lemma 7.

For the sake of clarity, we only describe the execution of Step 1, but all steps work as Step 1.
Step 1: As for Algorithm 2, it is subdivided into 3 Phases.

Phase 1: candidacy. Each node $s_i \in S_1$ locally generates one independent copy Z_i of a r.v. Z distributed as (10). In order to have an interval containing $[\log^\alpha(2n/5\log u), \log^\alpha(3n/5\log u)]$ by Lemma 7 with $N = \mathrm{Card}(S_1)$, each node sets $\mathcal{J} = \left[\log^\alpha\left(2u^{1/c}/5\log u\right), \log^\alpha(3u/5\log u)\right]$. If a node has $Z_i \in \mathcal{J}$, it becomes Candidate and the other nodes become Eliminated. We have $\mathrm{Card}(\mathcal{J}) < \log^\alpha(3u) = O(\log^\alpha n)$.

Phase 2: witnessing an election at a time slot and flooding the remaining rounds after an election. Each Eliminated node after Phase 1 sets $t_w = \mathrm{UAR}(\{t_0, t_1, \ldots, t_{2\log^\alpha(3u)-1}\})$ and $t_f = \mathrm{UAR}(\{t_w+2, \ldots, t_{4\log^\alpha(3u)\log u-1}\})$.

Phase 3: (*browsing through* $\mathcal{J}$). It works as the Phase 3 of Algorithm 2 but at any time slot t_{2g}, if a node receives a message, it does not need to be CANDIDATE to send feedback at $t_{2g}+1$. All the listening nodes that received a message at t_{2g}: the CANDIDATE nodes with $Z_i = \mathcal{J}_g$ and the ELIMINATED nodes with $t_w = t_{2g}$, send feedback to the unique sender s_1 at $t_{2g}+1$. So s_1 detects collision at $t_{2g}+1$ and knows that it transmitted alone. s_1 becomes LEADER and all the ELIMINATED nodes that received a message at t_{2g} flood all the remaining rounds of the algorithm so that no other CANDIDATE node can transmit alone.

All the nodes do these computations for $S_1, S_2, \ldots, S_{2\log u}$ and we have the following result.

Theorem 4. *Fix $\alpha \in]0,1[$, in single-hop RNCD (resp. RNstrongCD) networks of large size n, if no node knows the exact value of n, but an upper bound u of n is given in advance to all the nodes, there is a randomized Monte-Carlo leader election algorithm succeeding in $O(\log^{1+\alpha} n)$ time slots with a probability greater than $1 - O\left(n^{-\frac{1}{2}}\right)$. Each node transmits at most once and listens to the network during at most two (resp. one) time slots.*

Proof. The time complexity comes from $2\,\mathrm{Card}(\mathcal{J})\log u = O(\log^{1+\alpha} n)$. As *browsing through* $\mathcal{J}$ once elects a leader with a positive constant probability, a unique leader is elected *w.h.p.* after $O(\log n)$ executions of such protocol.

Each CANDIDATE node transmits once if the value in $\mathcal{J}$ corresponding to its Z_i is under checking and listens once to the network after such a time slot. An ELIMINATED node listens once at t_w and may transmit once at t_f if it witnessed the time slot corresponding to the first election. □

B.2 Energy bounds for leader election when n is known

We can see in [17] that if the nodes know n and if E denotes the maximum number of time slots during which a node has to transmit in order to elect a leader on a uniform randomized algorithm (see [17, Section III.A.]), for some constant $\varepsilon > 0$,

$$\mathbb{P}\left[E > \varepsilon\right] \leq \frac{10}{9}\left(\frac{e}{\varepsilon+1}\right)^{\varepsilon+1} e^{-(\varepsilon-2)/2\log n} + \left(1-\frac{1}{e}\right)^{\sqrt{n}}.$$

Then, by taking $\varepsilon = 2$, 3 and 4,

$$\mathbb{P}\left[E > 2\right] \leq cste,$$

$$\mathbb{P}\left[E > 3\right] \leq O\left(\frac{1}{\sqrt{n}}\right),$$

$$\mathbb{P}\left[E > 4\right] \leq O\left(\frac{1}{n}\right).$$

B.3 Algorithm 3

Algorithm 3. LeaderElection(u, α, c).

Input : An upper bound u of n, a constant $\alpha \in]0,1[$ and a constant c such that $u < n^c$.
Output: Each node s_i with a Status$(s_i) \in \{$Leader, Eliminated$\}$.

Phase 1: Each node s_i locally sets $V = \exp\left(u^{\alpha/(c(\alpha+1))}\right)$, generates V random copies $Y_{i,1}, Y_{i,2}, \ldots, Y_{i,V}$ of a r.v. Y distributed as described by (1) and saves $Y_i = \max_{h=1,2,\ldots,V}\{Y_{i,h}\}$. Then, s_i sets $\mathrm{L}_0 = (\log V - \log\log\log V)^{1+\alpha}$ and $\mathrm{L}_{\mathrm{Last}} = (\log(uV) + \log\log V)^{1+\alpha}$.
if $Y_i \in \mathrm{L} = [\mathrm{L}_0, \mathrm{L}_{\mathrm{Last}}]$ **then**
  s_i sets Status$(s_i) \leftarrow$ Candidate.
else
  s_i sets Status$(s_i) \leftarrow$ Eliminated.
end
Phase 2: Each Eliminated node s_e sets $t_w = \mathrm{UAR}(\{t_0, \ldots, t_{\mathrm{Card(I)}-1}\})$.
Phase 3: Each Candidate node s_i having $Y_i = \mathrm{L}_{\mathrm{Last}}$ beeps at t_0 and sets Status$(s_i) \leftarrow$ Leader.
for g *from* 0 *to* $\mathrm{Card(L)} - 2$ **do**
  Each Eliminated node s_e that has $t_w = t_g$ listens to the network at t_g.
  if s_e *hears* Beep *at* t_g **then**
    s_e beeps at $t_g + 1$.
  end
  Each Candidate node s_i having $Y_i = \mathrm{L}_{\mathrm{Last}} - (g+1)$ listens to the network at t_g.
  if s_i *does not hears* Beep *at* t_g **then**
    s_i beeps at $t_g + 1$ and sets Status$(s_i) \leftarrow$ Leader.
  else
    s_i beeps at $t_g + 1$ and sets Status$(s_i) \leftarrow$ Eliminated.
  end
end

B.4 Leader election on Beeping Networks when n is known

When the nodes initially know n, Algorithm 3 can be adapted as follows. In Phase 1, let $V = \exp\left(n^{1/1+\alpha}\right)/n$. Each node s_i locally generates V random copies $Y_{i,1}, Y_{i,2}, \ldots, Y_{i,V}$ of Y distributed as described by (10) and saves $Y_i = \max_{h=1,2,\ldots,V}\{Y_{i,h}\}$. Each node then computes $\mathrm{L}_0 = \left(n^{1/1+\alpha} - \log\log n\right)^{1+\alpha}$ and $\mathrm{L}_{\mathrm{Last}} = \left(n^{1/1+\alpha} + \log n\right)^{1+\alpha}$. After that, the candidacy, the witnessing and the browsing procedures work exactly as in Algorithm 3.

Theorem 5. *Fix $\alpha \in]0,1[$, in single-hop BN networks of large size n, if all nodes know the exact value of n, there is a randomized Monte-Carlo leader election*

algorithm that elects a leader in $O(n^{\alpha/\alpha+1}\log n)$ *time slots with probability* $1 - O\left(n^{-\alpha/\alpha+1}\right)$*. Each node transmits exactly once and listens once to the network.*

Proof. As in Appendix A.9, the time complexity is $\text{Card}(\text{L}) = O(n^{\alpha/\alpha+1}\log n)$ and its probability of success is greater than $1 - \log^{-\alpha}(nV) = 1 - O\left(n^{-\alpha/\alpha+1}\right)$. □

B.5 Adapting [6] on Radio Networks when the nodes know u

The leader election algorithms designed in [6] work when the nodes have no information about the topology of the network. If the nodes initially know an upper bound $u \in]n, n^c]$ for some constant $c > 1$, their algorithm [6, Section 6.3] can be adapted to elect a leader with $O(1)$ energy complexity as follows:

As all nodes know a value $v \in](1/c)\log n, \log n]$, each node sets a set of infinite integers $D = \{d_1, d_2, \dots\}$ where $d_1 = v$ and $d_i = d_{i-1} + v$. The nodes do a checkpoint at each time slot $d_i \in D$.

Initial Setup: Each node then gets a label k, with probability $\frac{1}{\sqrt{2^k}}$, for each integer $k \geq d_1$. Let S_k be the set of all nodes labeled k.

Finding an Estimate: For $k = k_0, k_0 + 1, k_0 + 2, \dots$, with $k_0 = d_1$, each node in S_k runs Verify $\left(\sqrt{2^k}\right)$ as described in [6, Section 6.1].

For the case that a checkpoint is met, *i.e.* $k = d_i$, let L_e (*resp.* L_o) be the set of leaders elected in Verify $\left(\sqrt{2^l}\right)$ for even (*reps.* odd) l so far. All nodes in L_o simultaneously announce their labels, while the other nodes listen to the network. If exactly one message is sent, the algorithm terminates, otherwise, repeat it with L_e.

As $k = O(\log n)$, the time complexity of Verify $\left(\sqrt{2^k}\right)$ is $O\left(\log\left(\sqrt{2^k}\right)\right) = O(\log n)$ [6, Section 6.1] and the time complexity of this adapted algorithm is $O(\log^2 n)$. The energy complexity of Verify$(\sqrt{2^k})$ is as follows: During the Assignment step (Step 1 in [6, Section 6.1]), each node may transmit during at most β time slots where β is a sufficiently large constant. Then, during Step 2 (Checking the correctness of estimate), the energy complexity of the **Census** algorithm is $O(\alpha(n))$ which is the inverse Ackermann of n. By calling such a **Census** algorithm, there is at least one node, the representative of a group [6, Section 4.1], that has to send the list of all ID s of all nodes in the group. Thus, as there are at most $O(\log n)$ ID s, taken from $\left[1, \log\left(\sqrt{2^k}\right)\right]$. Each ID is encoded into $O(\log\log n)$ bits, so the nodes must be able to transmit a message of size $O(\log n \log\log n)$.

When a checkpoint is met, each leader may transmit once and all the nodes may listen twice to the network. Thus, during the whole algorithm, each node

may transmit $c(\beta+1+O(\alpha(n)))=O(1)$ times and listen to the network during $c(\beta+2+O(\alpha(n)))=O(1)$ time slots.
As this algorithm relies on collision detection, it cannot be adapted to work on the BN, where the nodes cannot distinguish between one and multiple beeps.

B.6 Adapting [4] on Beeping Networks when u is known

In this section, we use the circuit simulation protocol designed in [4] to elect a leader on the Beeping Networks if the nodes have a common upper bound $u \in]n, n^c]$ where $c > 1$. Let $\mathbb{C}$ be a boolean circuit with d constant-fan-in gates. The control slot of the circuit simulation protocol [4, Section 2] takes expected $O\left(1 + d/n\right)$ energy per node. The circuit slot also has $2 + O(d/n)$ waking time. The authors noted that if a node wants to know the result of $\mathbb{C}$, it has to wake up and listen to the network during d time slots where d is the number of gates. Thus, the energy complexity of one distributed simulation of a circuit $\mathbb{C}$ is $d + 3 + O(d/n)$ in expectation. Let us give a trivial example of a leader election algorithm using such circuit simulation. Each node firstly chooses uniformly at random an ID from $[u^2] = [1, n^{2c}[$. Obviously whoever holds the maximum number is unique with probability at least $1 - 1/n^{c-1}$. Each node encodes its ID into a code-word $R = R_1, R_2, \ldots$ in base $u^{\alpha/n} \in]n^{\alpha/n}, n^{\alpha}]$ for some constant $\alpha \in]0, 1[$. Using the circuit-simulation technique, one can find the maximum index "R_1" among all the ID s with $d + 3 + O(d/n)$ energy complexity in expectation. The nodes then find the maximum R_2 among all nodes that have the maximum R_1 and so on. As the length of each R_i is at most $\log_{n^{\alpha/c}}(n^{2c}) = 2c^2/\alpha$, this algorithm terminates in $O(n^{\alpha})$ time slots with $(2c^2(d + 3 + O(d/n)))/\alpha$ energy complexity. We note that this is at least $2(d + 3) > 6$.