Energy Efficient Naming in Beeping Networks

Ny Aina Andriambolamalala(✉) and Vlady Ravelomanana(✉)

Université de Paris, IRIF, CNRS, 75013 Paris, France
Ny-Aina.Andriambolamalala@irif.fr, vlad@irif.fr

Abstract. A single-hop beeping network is a distributed communication model in which each station can communicate with all other but only by $1-bit$ messages called beeps. In this paper, we focus on resolving two fundamental distributed computing issues: the naming and the counting on this model. Especially, we are interested in optimizing energy complexity and running time for those issues. Our contribution is to have design randomized algorithms with an optimal running time of $O(n \log n)$ and optimal $O(\log n)$ energy complexity whether for the naming or the counting for a single-hop beeping network of n stations.

Keywords: Distributed · Initialization · Naming · Energy · Optimal · Beep

1 Introduction

Introduced by Cornejo and Kuhn in 2010 [8], the beeping model makes little demands on the devices which need only to be able to do carrier-sensing, differentiating between silence and the presence of a jamming signal on the network (considered as $1-bit$ message or one beep). Such devices have unbounded local power computation [6]. They note in [8] that carrier-sensing can typically be done much more reliably and requires significantly less energy and other resources than message-sending models. Minimizing such energy consumption per node arises as all nodes are battery-powered. Since sending or receiving messages costs more energy than internal computations, the energy consumption is measured by the maximal waking time of any node (beeping or listening to the network) [6,16,17,19,21,29]. It is more realistic when the nodes have no prior information about the topology of the network and are initially indistinguishable (have no identifier denoted ID). To break such symmetry, researchers designed various protocols such as leader election ([6,11,12,15,17–19,23,26]) Maximal Independent Set ([1,28]) and naming protocols ([2,7,13,20,21]). In this paper, we consider the naming problem on the single-hop[1] beeping networks which consists in assigning a unique label $\ell \in \{1, 2, ...n\}$ to each node. On the previously described model, we design an energy optimal randomized naming algorithm succeeding

[1] The underlying graph of the network is a complete graph.

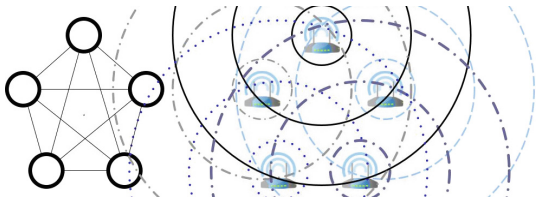
M. R. Palattella et al. (Eds.): ADHOC-NOW 2019, LNCS 11803, pp. 355–369, 2019.
https://doi.org/10.1007/978-3-030-31831-4_25

in $O(n \log n)$ time slots with high probability[2] (*w.h.p.*), having $O(\log n)$ energy complexity. We start by presenting a deterministic algorithm naming M nodes ($M \leq n$) in $O(M \log n)$ time slots with $O(M + \log n)$ energy (Sect. 2). We then consider the case when n is unknown in Sect. 3. This is then adapted to solve the counting problem, consisting in assigning their exact number to all nodes (Sect. 3). Thereafter, we use derandomization techniques to adapt our algorithm in order to have a deterministic one if n is known beforehand (Sect. 4) terminating with $O(\log n)$ energy complexity. As customary in deterministic settings, we assume that the nodes have unique ID $\in \{1, 2, ...N\}$ (N is a polynomial upper bound of n). Finally, we prove a lower bound of $\Omega(\log n)$ on the energy complexity for naming a beeping network in Sect. 5 and present maple simulation results illustrating our works in Sect. 6.

1.1 The Models

In a single-hop beeping network, nodes communicate with each other via a shared beeping channel.

As shown in the following Figure, this can be used for modeling an ad hoc network where all nodes are in each other's communication range. The nodes can send $1-bit$ messages and do a carrier sensing in order to detect any transmission.



At each synchronous discrete time slot, a node independently decides whether to transmit a beep, to listen to the network or to remain idle (asleep). Only listening nodes can receive the state of the common channel which can be, Beep if at least one node is transmitting or Null when no node transmits. This model is also called BL or Beep Listen model. In this paper, we use in general the BL except for the randomized counting protocol for which we use the $B_{\text{CD}}L$ model (Beep with Collision Detection Listen) where transmitters can detect collisions [1,28].

1.2 Related Works and New Results

As a fundamental distributed computing problem [20], many results exist for the naming problem. Let us first consider the simplest model, the single-hop network, where the underlying graph of the network is complete. In [13], Hayashi, Nakano and Olariu presented a $O(n)$ running time randomized protocol for radio networks with collision detection (RNCD). Later, Bordim, Cui, Hayashi, Nakano and Olariu [2] presented an algorithm terminating *w.h.p.* in $O(n)$ time slots, and $O(\log n)$ energy complexity. In [22], for radio network with no collision detection

[2] An event ε_n occurs with high probability if $\mathbb{P}[\varepsilon_n] \geq 1 - \frac{1}{n^c}$ for any constant $c > 0$.

(RNnoCD), Nakano and Olariu designed a protocol terminating in $O(n)$ time slots *w.h.p.* with $O(\log \log n)$ energy complexity. The results on beeping model appeared very recently when Chlebus, De Marco and Talo [7] presented their algorithm terminating in $O(n \log n)$ time slots *w.h.p.* for the BL model and provided $\Omega(n \log n)$ lower bound on the time complexity. Moreover, Casteigts, Métivier, Robson and Zemmari [4] presented a counting algorithm for the $B_{\text{CD}}L$ model terminating in $O(n)$ time slots *w.h.p.* They noticed that adapting their algorithm to the BL model will cost a logarithmic slowdown in time complexity. The following Table shows our results on single-hop networks (Table 1).

Table 1. Our results

Problem and model	Time complexity	Energy complexity	Succeed with probability
Randomized naming in BL network Indistinguishable nodes, n unknown, Theorem 2, 5	$O(n \log n)$	$\Theta(\log n)$	$1 - O(\frac{1}{n^c})$ $c > 0$
Unconditional Deterministic naming in BL Unique ID $\in \{1, N\}$, n unknown, Theorem 1	$O(n \log n)$	$O(n)$	-
Derandomized Deterministic naming in BL Unique ID $\in \{1, N\}$, n unknown, Theorem 4	$O(n \log n)$	$O(\log n)$	$1 - O(\frac{1}{n^c})$ $c > 0$
Randomized Counting in BL Theorem 3, 5	$O(n \log n)$	$\Theta(\log n)$	$1 - O(\frac{1}{n^c})$ $c > 0$

The more realistic model where the underlying graph of the network is an arbitrary connected graph (it is called the multi-hop network model) also gained in importance as subject of researches [24]. The only analysis for the initialization protocol in such a multi-hop case was given in [27] and was restricted to a set of nodes randomly thrown in a square. It will then be very interesting to adapt our designed protocols to work on such a model.

2 New Approach: Deterministic Naming of M Nodes

Let N be a polynomial upper bound of n known by the nodes, each node having a unique identifier denoted ID $\in \{1, 2, ...N\}$. In the next sections, N is randomly approximated by the nodes if unknown and the nodes randomly generate unique ID *w.h.p.* In this section, we use a known method consisting in representing ID by its binary encoding and sending the obtained bits one by one in reverse order [14]. If M nodes ($M \leq n$) hold such unique ID, they firstly encode their ID into a binary code-word denoted CID = [CID[1]CID[2]...CID[$\lceil \log_2 N \rceil$]] such that CID[i] $\in \{0, 1\}$ (CID[1] corresponds to $2^{\lceil \log_2 N \rceil}$ and CID[$\lceil \log_2 N \rceil$] corresponds to 2^0). Each participant then sends its CID bit by bit during $\lceil \log_2 N \rceil = O(\log n)$ time slots in order to know if it has the largest ID of all participants. If a node detects that one of its neighbors has a higher ID, it gets eliminated (it is no longer a candidate to take the next available label). Then, the unique node holding the

largest ID gets such next available label. Such an algorithm can be considered as M deterministic seasons $S_1, S_2, ...S_M$ (the nodes do not know M). In one season S_j, each node sends its CID bit by bit during $\lceil \log_2 N \rceil$ steps $t_1, t_2, ...t_{\lceil \log_2 N \rceil}$. We define the TEST($i$) protocol, called at a step t_i and taking the step number 'i' as parameter. It encodes one bit CID[i] into two communication steps $t_{i,0}$ and $t_{i,1}$ and outputs a status $\in$ {ELIMINATED, ACTIVE, NULL}.

-TEST(i): If the node s running TEST(i) has CID[i] = 0, then it beeps at $t_{i,0}$ and listens to the network at $t_{i,1} = t_{i,0} + 1$. If it hears beep at $t_{i,1}$, TEST(i) returns ELIMINATED. Otherwise, it returns NULL. If s has CID[i] = 1, then it listens at $t_{i,0}$. If it hears beep at $t_{i,0}$, TEST(i) returns ACTIVE, otherwise, it returns NULL.

Then at any step t_i, by executing TEST(i), each participant knows if at least one of them has CID[i] = 1. In such case, each node s having CID[i] = 0 gets eliminated until the next season S_{j+1}. At the end of the season S_j, the last non-eliminated node takes the label j. By looping these computations until no node remains unlabeled, this method produces a naming algorithm terminating in $O(M \log n)$ time slots.

Energy Optimization Principle: The latter algorithm is not energy efficient because all nodes have to be awake during the whole $O(M \log n)$ time slots. To improve such energy consumption, we remark that each node s must be awake only during two specific set of steps in order to know if any of its neighbors has a higher ID. Thus, we introduce the following two definitions of such steps.

Definition 1 (Step To Listen: STL**).** *A* STL *is one step t_i recorded by the node s during any season S_j. A node s receiving* TEST*(i) =* ELIMINATED *records i into* STL *and on the next seasons $S_{j+1}, ...S_M$, s wakes up and listens at $t_{i,0}$ in order to verify if it is still eliminated at this step. s may not sleep after t_i.*

Definition 2 (Steps To Notify: STN**).** *A* STN *is a set of steps $\{t_i, t_k, ...\}$ recorded by the node s_1 during any season S_j. A node s_1 receiving* TEST*(i) =* ACTIVE *knows there is at least one node s_2 having* CID[i] = 0 *while it has* CID[i] = 1*. It saves i into* STN *because at the next seasons, it has to beep at $t_{i,1}$ in order to notify that s_2 is still eliminated at this step. When s_1 adds i into* STN*, it has no more active neighbor holding* CID[k] = 1, $k > i$*. Thus, s_1 empties* STL*.*

Description of the Energy Efficient Algorithm: All nodes are initially sleeping and run the following computations during some seasons $S_1, S_2, \ldots$ until being labeled. For any season S_j ($j \in \{1, 2, \ldots, M\}$ and the nodes do not know M), a node s wakes up only at the first step t_i found in its STL or STN. If such $i \in$ STN, then s sleeps before moving at t_{i+1}. Otherwise, if $i \in$ STL and s has TEST(i) = ELIMINATED, then it sleeps until the next season. s stays awake and moves on t_{i+1} if TEST(i) = ACTIVE. At the end of season S_j, the last remaining awake node sets $\ell = j$, empties STN and STL and sleeps. For a better comprehension, we represent the execution of the algorithm by a binary

tree as done in [10]. One path of such a tree represents the CID of a device and one of its edges represents one bit of such CID. In the figures showing such representation, we consider the execution of the naming algorithm for only one device. In such figures, the hexagons represent the STL, the squares represent the STN and the circles represent the other waking steps. The number inside these shapes represents the season during which the node wakes up.

Algorithm 1. DeterministicNaming(N) on any node s

Input : Upper bound N of n, unique ID $\in \{1, 2, ...N\}$
Output: Node s has unique label $\ell \in \{1, 2, ...M\}$

encode ID into binary code-word CID $= \{0,1\}^{\lceil \log_2 N \rceil}$;
$\ell \leftarrow 0$; STL $\leftarrow$ Null; STN $\leftarrow$ Null; $S \leftarrow 1$; $Test \leftarrow$ Null.
while $\ell = 0$ **do**
  for $i \leftarrow 0$ *to* $\lceil \log_2 N \rceil$ **do**
    if $i \in$ STL **then**
      wake up at t_i; $Test \leftarrow$ Test(i)
      if $Test =$ Eliminated **then**
        sleep
      end
    end
    if $i \in$ STN **then**
      wake up at t_i, run Test(i) and sleep
    end
    if s *is awake and* $i \notin$ STL **then**
      $Test \leftarrow$ Test(i)
      if $Test =$ Active **then**
        add i into STN and empty STL
      end
      if $Test =$ Eliminated **then**
        add i into STL and sleep
      end
    end
  end
  if s *is awake* **then**
    $\ell \leftarrow S$; STL $\leftarrow$ Null; STN $\leftarrow$ Null; $Test \leftarrow$ Null; sleep
  end
  $S \leftarrow S + 1$
end

The presented example in the following Figure is for 9 devices having ID $\in \{15, 14, 13, 12, 11, 10, 9, 8, 7\}$. The black leaf represents the device having CID = [1010]. s wakes up at step t_1 of season S_1 (1 inside a circle for the root node in the figure). It has CID[1] = 1 and hears that some nodes have CID[1] = 0 then saves 1 into its STN (Fig. 1).

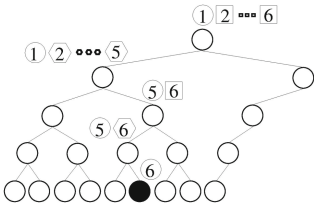


Fig. 1. Example of execution of Algorithm 1

It wakes up at t_2 (1 inside a circle for the next node in the left of the root). As CID[2] = 0 and s hears that some nodes have CID[2] = 1, it adds 2 into its STL and sleeps until the end of S_1. s wakes up at t_1 of S_2 because 1 is in its STN (2 inside a square for the root). Then it wakes up at t_2 as 2 is in its STL (2 inside an hexagons for the left node after the root). As there remains a node having CID[2] = 1, it sleeps until the end of the season S_2. s do the same computations for seasons $S_3, \ldots, S_6$ and gets labeled at S_6.

Lemma 1. *In single hop beeping networks of size n, there is a deterministic algorithm naming some M participating nodes in $O(M \log n)$ time slots with no node being awake for more than $O(M + \log n)$ steps.*

Proof. Algorithm 1 terminates deterministically in $M \times \lceil \log N \rceil = O(M \log n)$ time slots. In the following, let W_s be the total waking times of any node s in the previously defined Algorithm 1, W_{STN}, W_{STL} and W_{other} correspond to STN total waking time, STL and other total waking times. Similarly, $(W_{STN})_{worst}$, $(W_{STL})_{worst}$ and $(W_{other})_{worst}$ are the worst waking times of all nodes. We have

$$W_s = W_{\text{STN}} + W_{\text{STL}} + W_{other} \leq (W_{\text{STN}})_{worst} + (W_{\text{STL}})_{worst} + (W_{other})_{worst}\,. \quad (1)$$

In order to find $(W_{STN})_{worst}$ and $(W_{STL})_{worst}$, we can simulate a complete binary tree to be the tree representation of the networks devices as done in [10]. For a better comprehension, we illustrate how we obtained the two following figures in Appendix 1 and Appendix 2.

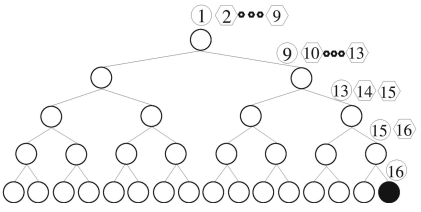


Fig. 2. Worst case for STL.

The node s having $(W_{STL})_{worst}$ (the black node in this Figure) wakes up T times (T Seasons) at any step t_i of STL until no other node has a higher ID. This value T is at most half of participants on t_1 and gets halved every i. We can see (by the hexagons shapes), that s wakes up $\frac{M}{2} + 1$ times in season S_1 (Fig. 2).

Furthermore, as for STL, a node s wakes up at $t_i \in$ STN during as many times (Seasons) as the number of nodes with a higher ID. I.e., half the number of participants at step t_1. This value gets halved every i and we have (Fig. 3)

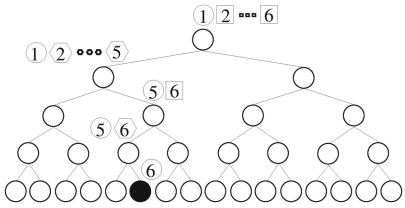


Fig. 3. Worst case for STN.

$$(W_{STN})_{worst} \sim (W_{STL})_{worst} \leq \sum_{i=1}^{M} \left(\frac{M}{2^i} + 1 \right) \leq O(M) \tag{2}$$

A node s wakes up just once at any step t_i not in STN and STL (we can see that by the round shapes in the Figures). Hence, we have

$$(W_{other})_{worst} \leq \sum_{i=1}^{\log N} O(1) \leq O(\log N) \leq O(\log n)\,. \tag{3}$$

□

A Maple simulation illustrates our results in Sect. 6.

Theorem 1. *In single-hop beeping networks of size n, if no node knows n but a polynomial upper bound N of n is given in advance to all nodes and the nodes have a unique* ID $\in \{1, 2, ...N\}$*, there is an energy efficient deterministic naming algorithm, assigning unique label to all nodes in $O(n \log n)$ time slots, with no node being awake for more than $O(n)$ time slots.*

Proof. Applying Lemma 1 to $M = n$, we reach the desired result. □

In Sect. 3, we use Algorithm 1 as a subroutine to design a randomized energy efficient naming protocol, having $O(\log n)$ energy complexity. To do so, we distribute the nodes into $O\left(\frac{n}{\log n}\right)$ groups in order to have $\Theta(\log n)$ nodes in each group. The main idea is to make $\Theta(\log n)$ nodes running the DETERMINISTIC-NAMING(N) protocol $O\left(\frac{n}{\log n}\right)$ times (each group executes DETERMINISTIC-NAMING(N) one time) instead of n nodes calling DETERMINISTICNAMING(N) one time. This leads us to a $O(\log n)$ waking time per node.

3 Energy Efficient Randomized Algorithms

We assume that the total number of nodes is unknown and that the nodes are initially indistinguishable. All the nodes then have to know a linear approximation u of n. This approximation problem was well studied in the distributed computing area. Brandes, Kardas, Klonowski, Pająk and Wattenhofer [3] designed a randomized linear approximation algorithm, terminating *w.h.p.* in $O(\log n)$ rounds. Our main idea is to make all nodes approximating u in $O(\log n)$ time

slots using algorithm presented in [3], which can be parameterized in order to have $u \in [\frac{1}{2}n, 2n]$ in $O(\log n)$ time slots (*i.e.* $2u \in [n, 4n]$ is locally known by the nodes). Then each node chooses uniformly at random to enter into one of $\lceil \frac{2u}{\log(2u)} \rceil = O(\frac{n}{\log n})$ groups.

Lemma 2. *As a classical result (see for instance [25]), if n nodes randomly and uniformly choose to enter into $\lceil \frac{n}{\log n} \rceil$ groups, there is at most $4 \log n$ nodes in each group with high probability.*

Proof. The probability to enter any group G_i is $O\left(\frac{\log n}{n}\right)$. As a consequence, if $|G_i|$ denotes the number of nodes in a group G_i, then $\mathbb{E}[|G_i|] = O(\log n)$. Hence, by means of Chernoff bound, $|G_i| \leq O(\log n)$ with probability at least $1 - O\left(\frac{1}{n}\right)$. □

After that, each node takes a unique ID uniformly from $\{1, 2, ...(2u)^2\}$. We then sequentially run DeterministicNaming$((2u)^2)$ on each group one group at a time. Firstly, each node in the group G_1 works during at most $\lceil \log(2u)^2 \rceil = O(\log n)$ time slots to name itself. Then, during extra $O(\log n)$ time slots, the last labeled node in G_1 sends its label bit by bit to the next group G_2. In parallel, all the nodes in G_2 wake up and listen to the network during $O(\log n)$ time slots. Those nodes save the received value into a variable ℓ_{prev}. By running DeterministicNaming(N), all nodes in G_2 have a label $\ell \in \{1, 2, \ldots, |G_2|\}$. Then, each of them has to update $\ell \leftarrow \ell + \ell_{prev}$ in order to make a labeling $\in \{1, 2, \ldots, n\}$. We apply these computations to each couple of groups $\{\{G_1, G_2, \}, \{G_2, G_3\}, \ldots\}$ one by one.

To know if any node s has the last label of its group, we modify the DeterministicNaming(N) algorithm such that a node labeled at a season S_j wakes up during the entire season S_{j+1} and listens to the network, finding out if there remain unlabeled nodes. This extra $O(\log n)$ waking time doesn't affect our $O(\log n)$ energy complexity.

Theorem 2. *In single-hop beeping networks of size n, if n is unknown by all nodes and nodes are initially indistinguishable, there is an energy efficient randomized naming algorithm, assigning a unique label to all nodes in $O(n \log n)$ rounds w.h.p, with no node being awake for more than $O(\log n)$ time slots.*

Proof. The latter described algorithm uses DeterministicNaming(N) and is therefore quasi deterministic. As by [3], $u = \Theta(n)$, if we note the time complexity of DeterministicNaming$(N = (2u)^2)$ algorithm by T_D, our naming algorithm terminates in $\lceil \frac{2u}{\log(2u)} \rceil \times T_D$ time slots. Then by Lemma 2, the number of participants is at most $O(\log n)$ *w.h.p.* Thus, by using $M = O(\log n)$ in Lemma 1 we get $T_D = O(\log^2 n)$, implying the $O(n \log n)$ time complexity of our randomized naming algorithm.
Therefore, each node s is awake only during the execution of the DeterministicNaming$((2u)^2)$ protocol and $O(\log n)$ extra times for checking if s has the last label as well as sending it to the next group. Consequently, the energy complexity is $O(\log n)$. □

Using such an algorithm, we can design a counting algorithm with $O(n \log n)$ time complexity and $O(\log n)$ energy complexity on the single-hop BL network. To do so, we add the following computations. As it terminates after at most $\lceil \frac{2u}{\log(2u)} \rceil \times \lceil \log^2(2u) \rceil = \lceil 2u \log(2u) \rceil$, all nodes wake up after $\lceil 2u \log(2u) \rceil$ time slots (counted from the first time slot of the Season S_1 for the first group) and the last labeled node send its label bit by bit.

Theorem 3. *In single-hop beeping networks of size n, if n is unknown by all nodes and nodes are initially indistinguishable, there is an energy efficient randomized counting algorithm allowing all the nodes to know the exact number of the participants, terminating in $O(n \log n)$ rounds w.h.p, with no node being awake for more than $O(\log n)$ time slots.*

Proof. If at the end of the last group $G_{\lceil \frac{2u}{\log(2u)} \rceil}$, all nodes wake up and the last labeled node sends its label bit by bit, this value corresponds *w.h.p.* to the exact number of nodes on the network. □

4 Deterministic Energy Efficient Naming Algorithm

The randomized part of our algorithm consists in the assignment of all nodes to $O\left(\frac{n}{\log n}\right)$ groups of size $O(\log n)$ each. Then, the nodes execute the DeterministicNaming(N) protocol one group at a time in order to have each node awake for at most $O(\log n)$ time slots. In this Section, we consider a network of n nodes that know the exact value of n. Each node has a unique ID taken from $\{1, N\}$ where N is a polynomial upper bound of n. Our goal is to do the previous group assignment in a deterministic manner, with a very small error rate. To do so, we use a hash function in order to map each node's ID into $\lceil \frac{n}{\log n} \rceil$ values, such that the nodes holding the same value belong to the same group.

Celis, Reingold, Segev and Wieder [5] construct such hashing function, by encoding integer values into binary code-words of length $O(\log n \log \log n)$, such that there is at most $O(\frac{\log n}{\log \log n})$ integers mapped to the same code-word with a probability greater than $1 - O\left(\frac{1}{n^c}\right)$, c being a positive constant. Having this in mind, each node firstly maps its ID into a code-word, using the hashing function described in [5]. The nodes having the same code-word are in the same group. Then, the nodes having the first code-word (the nodes in the first group) execute DeterministicNaming(N) in order to be labeled. The last labeled node sends ℓ bit by bit to the next group during $O(\log n)$ time slots when the nodes having the second code-word listen to the network. Those nodes in the second group run DeterministicNaming(N) and add the previously received label to the latter computed label. The last labeled node in the group 2 sends ℓ to the next group and so on.

With such adaptations, we have the following result.

Theorem 4. *In single-hop beeping networks of size n, if n is known in advance by all nodes and nodes have a unique* ID $\in \{1, 2, ...N\}$ *(N is a polynomial upper*

bound of n), there is an energy efficient deterministic naming algorithm, assigning unique label $\ell \in \{1, 2, ...n\}$ *to all nodes in* $O(n \log n)$ *time slots, having no node being awake for more than* $O(\log n)$ *time slots, with a probability of error less than* $O\left(\frac{1}{n^c}\right)$*, for some constant* $c > 0$ *independent of* n*.*

5 Lower Bound on Energy Complexity

In [7], the authors presented an $\Omega(n \log n)$ lower bound for the running time of any randomized naming algorithm. We use such a lower bound in order prove the following result.

Theorem 5. *The energy complexity of any randomized algorithm solving the naming problem with constant probability is* $\Omega(\log n)$*.*

Proof. It was proved in [7] that any randomized algorithm for naming n stations requires $\Omega(n \log n)$ expected time slots to succeed with a probability of error smaller than $\frac{1}{2}$. Their proof uses the Yao's minimax principle and is combined to Shannon's entropy [9]. We use such running time lower bound to prove the Theorem 5 by contradiction. Let us first remind that the time complexity of any distributed algorithm is measured by the communication time instead of local computations and that the energy complexity is measured by the maximal waking (communication) time of any node. We suppose that there is a randomized naming algorithm with $o(\log n)$ energy complexity. *i.e.* each node communicates on the network during at most $o(\log n)$ time slots when running such an algorithm. It is then straightforward to see that the total communication time (*i.e.* time complexity by definition) of such algorithm is at most $o(n \log n)$. This contradicts the given lower bound of $\Omega(n \log n)$ for time complexity in [7]. □

6 Maple Simulation

In this Section, we present a maple simulation of Algorithm 1: DETERMINISTIC-NAMING(N) where n, the total number of nodes, varies from 10^5 to 10^{10}, 10^6 by 10^6. The $X - axis$ corresponds to the values of n while the $Y - axis$ corresponds to the waking time numbers.

For each value of n, $N = n^2$ and M nodes participate to the naming task (for sake of simplicity, we fix $M = \lceil \log_2 N \rceil$). For each value of n, we randomly choose one of the M participating nodes in order to count the total number of its waking time. In the following Figure, the green (or grey) graph represents the total waking time of any node s_i taken randomly from the M participating nodes $s_1, s_2 ... s_M$ for each values of n. The blue (or bold black) graph is the values of M for each value of n and the red (or black) graph represents $c \times M$ (here $c = 3.6$) (Fig. 4).

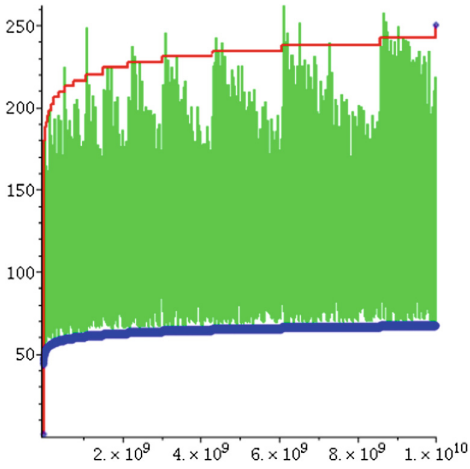


Fig. 4. The energy complexity of nodes taken randomly from a set of $\lceil \log N \rceil$ nodes. (Color figure online)

The maple codes are available in

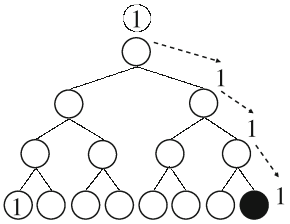
https://www.irif.fr/~nixiton/initLoop.mw or in
https://www.irif.fr/~nixiton/initLoop.pdf

7 Conclusion

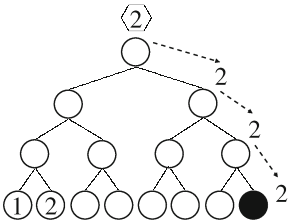
In this paper, we focus on the naming problem in single-hop beeping networks. We start by a deterministic version, when the nodes known N, a polynomial upper bound of n and all nodes have a unique ID $\in \{1, N\}$. Such a protocol has $O(n \log N)$ time complexity and $O(n)$ energy complexity. Then, when the nodes do not know the exact value of n and are initially indistinguishable, we design a randomized algorithm terminating in optimal $O(n \log n)$ time slots *w.h.p.*, and optimal $O(\log n)$ energy complexity. We have also established that for the same task, $\Omega(\log n)$ waking time slots are necessary for any randomized algorithm to succeed with a constant probability. Our algorithm can be used for the counting problem, returning the exact number of the nodes in $O(n \log n)$ time slots, with $O(\log n)$ energy complexity. By means of derandomization, we devise an energy-efficient deterministic naming algorithm that errs with probability less than $O\left(\frac{1}{n^c}\right)$ terminating in $O(n \log n)$ time slots with $O(\log n)$ energy complexity. Our protocols has optimal time and energy complexity for the single-hop network. It will be then interesting to consider how to adapt such a protocol to work on the multi-hop beeping network model which is much more realistic than the single-hop one.

Appendix 1: Worst Case for STL

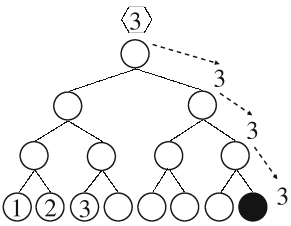
Here, we show a simulation of the execution of Algorithm 1 on the worst case for STL in a complete binary Tree to count the number of waking time of this node.



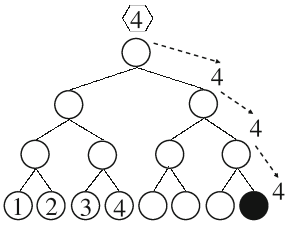
At season S_1 the node s wakes up once at step t_1, gets eliminated and sleeps until season S_2. Node 1 gets labeled.



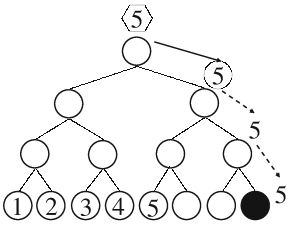
At S_2, s wakes up once at step $t_1 \in$ STL, gets eliminated and sleeps until S_3. Node 2 gets labeled.



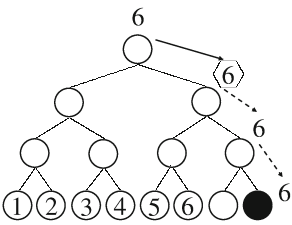
At S_3, s wakes up at step $t_1 \in$ STL, gets eliminated and sleeps until S_4. Node 3 gets labeled.



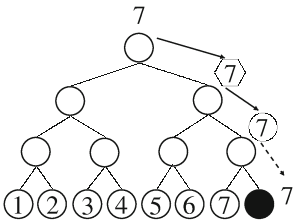
At S_4, s wakes up at step $t_1 \in$ STL, gets eliminated and sleeps until S_5. Node 4 gets labeled.



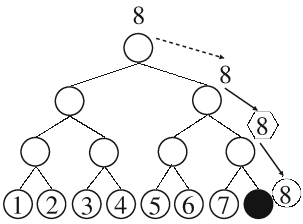
At S_5, s wakes up at step $t_1 \in$ STL. Since there is no more node that can eliminate it, s remain awake until t_2, gets eliminated there and sleeps until S_6. Node 5 gets labeled.



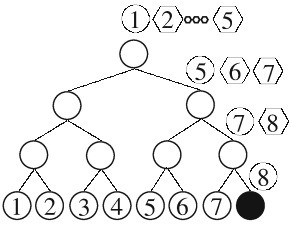
At S_6, s wakes up at step $t_2 \in$ STL, gets eliminated and sleeps until S_7. Node 6 gets labeled.



At S_7, s wakes up at step $t_2 \in$ STL, remains awake until t_3, gets eliminated at t_3 and sleeps until S_8. Node 7 gets labeled.



At S_8, s wakes up at step $t_3 \in$ STL, remains awake until t_4. s gets labeled at t_4.



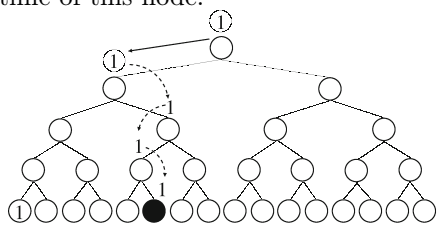
All STL waking times

Legends: hexagons represent the STL waking steps of the node, squares are the STN waking steps and circles represent the other waking steps. The numbers

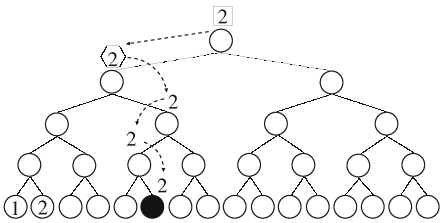
inside these shapes represent the season where the node wakes up. Numbers without any shape represent the sleeping steps of the node. Dotted lines represents the transition between two steps t_i, t_{i+1} on any season where the node starts to sleeps or remains sleeping. And solid lines the transition between two steps t_i, t_{i+1} on any season where the node wakes up or remains awake.

Appendix 2: Worst Case for STN

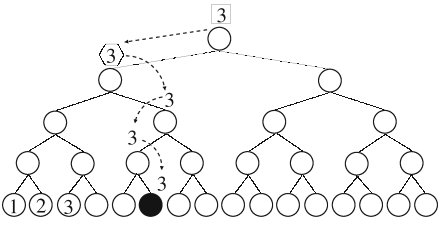
In this section, we show a simulation of the execution of Algorithm 1 on the worst case for STN in a complete binary Tree to count the number of waking time of this node.



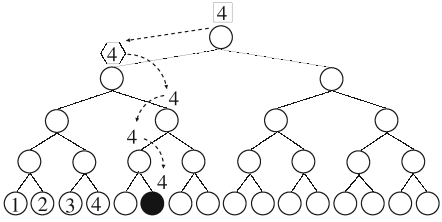
At season S_1 the node s wakes up once at step t_1, remains awake until t_2, gets eliminated at t_2 and sleeps until season S_2. Node 1 gets labeled.



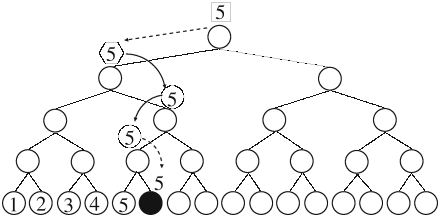
At S_2, s wakes up at step $t_1 \in$ STN, sleeps and wakes up at $t_2 \in$ STL, gets eliminated at t_2 and sleeps until season S_2. Node 2 gets labeled.



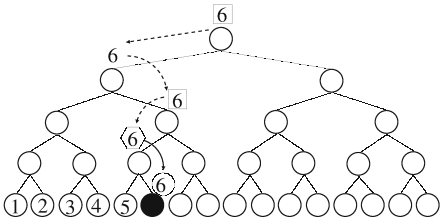
At S_3, s wakes up at step $t_1 \in$ STN, sleeps and wakes up at $t_2 \in$ STL, gets eliminated at t_2 and sleeps until season S_4. Node 3 gets labeled.



At S_4, s wakes up at step $t_1 \in$ STN, sleeps and wakes up at $t_2 \in$ STL, gets eliminated at t_2 and sleeps until season S_5. Node 4 gets labeled.



At S_5, s wakes up at step $t_1 \in$ STN, sleeps and wakes up at $t_2 \in$ STL, Since there is no more node that can eliminate it there, s remain awake until t_4 where it gets eliminated, and sleeps until S_6. Node 4 gets labeled.



At S_6, s wakes up at step $t_1 \in$ STN, sleeps and wakes up at $t_3 \in$ STN, sleeps an wakes up at $t_4 \in$ STL. Since there is no more node that can eliminate it there, s remain awake until t_6 and gets labeled.

References

1. Afek, Y., Alon, N., Bar-Joseph, Z., Cornejo, A., Haeupler, B., Kuhn, F.: Beeping a maximal independent set. Distrib. Comput. **26**(4), 195–208 (2013)
2. Bordim, J.L., Cui, J., Hayashi, T., Nakano, K., Olariu, S.: Energy-efficient initialization protocols for ad-hoc radio networks. ISAAC 1999. LNCS, vol. 1741, pp. 215–224. Springer, Heidelberg (1999). https://doi.org/10.1007/3-540-46632-0_23
3. Brandes, P., Kardas, M., Klonowski, M., Pająk, D., Wattenhofer, R.: Approximating the size of a radio network in beeping model. In: Suomela, J. (ed.) SIROCCO 2016. LNCS, vol. 9988, pp. 358–373. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-48314-6_23
4. Casteigts, A., Métivier, Y., Robson, J.M., Zemmari, A.: Counting in one-hop beeping networks. Theoret. Comput. Sci. (2016, to appear)
5. Celis, L.E., Reingold, O., Segev, G., Wieder, U.: Balls and bins: smaller hash families and faster evaluation. SIAM J. Comput. **42**(3), 1030–1050 (2013)
6. 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)
7. Chlebus, B.S., De Marco, G., Talo, M.: Naming a channel with beeps. Fundamenta Informaticae **153**(3), 199–219 (2017)
8. Cornejo, A., Kuhn, F.: Deploying wireless networks with beeps. In: Lynch, N.A., Shvartsman, A.A. (eds.) DISC 2010. LNCS, vol. 6343, pp. 148–162. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-15763-9_15
9. Cover, T.M., Thomas, J.A.: Elements of Information Theory. Wiley Series in Telecommunications and Signal Processing, 2nd edn. Wiley, Hoboken (2006)
10. Fuchs, M., Hwang, H.K.: Dependence between external path-length and size in random tries. arXiv preprint arXiv:1604.08658 (2016)
11. 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)
12. Ghaffari, M., Lynch, N., Sastry, S.: Leader election using loneliness detection. Distrib. Comput. **25**(6), 427–450 (2012)
13. Hayashi, T., Nakano, K., Olariu, S.: Randomized initialization protocols for packet radio networks. In: ipps, p. 544. IEEE (1999)
14. Jacquet, P., Milioris, D., Mühlethaler, P.: A novel energy efficient broadcast leader election. In: 2013 IEEE 21st International Symposium on Modelling, Analysis and Simulation of Computer and Telecommunication Systems, pp. 495–504. IEEE (2013)
15. Jurdziński, T., Kutyłowski, M., Zatopiański, J.: Efficient algorithms for leader election in radio networks. In: Proceedings of the Twenty-First Annual Symposium on Principles of Distributed Computing, pp. 51–57. ACM (2002)
16. Jurdziński, T., Kutyłowski, M., Zatopiański, J.: Energy-efficient size approximation of radio networks with no collision detection. In: Ibarra, O.H., Zhang, L. (eds.) COCOON 2002. LNCS, vol. 2387, pp. 279–289. Springer, Heidelberg (2002). https://doi.org/10.1007/3-540-45655-4_31
17. Kardas, M., Klonowski, M., Pajak, D.: Energy-efficient leader election protocols for single-hop radio networks. In: 2013 42nd International Conference on Parallel Processing (ICPP), pp. 399–408. IEEE (2013)

18. Kutten, S., Pandurangan, G., Peleg, D., Robinson, P., Trehan, A.: Sublinear bounds for randomized leader election. In: Frey, D., Raynal, M., Sarkar, S., Shyamasundar, R.K., Sinha, P. (eds.) ICDCN 2013. LNCS, vol. 7730, pp. 348–362. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-35668-1_24
19. Lavault, C., Marckert, J.F., Ravelomanana, V.: Quasi-optimal energy-efficient leader election algorithms in radio networks. J. Inf. Comput. **205**(5), 679 (2007)
20. Nakano, K.: Optimal initializing algorithms for a reconfigurable mesh. J. Parallel Distrib. Comput. **24**(2), 218–223 (1995)
21. Nakano, K., Olariu, S.: Energy-efficient initialization protocols for radio networks with no collision detection. In: 2000 International Conference on Parallel Processing, pp. 263–270 (2000)
22. Nakano, K., Olariu, S.: Energy-efficient initialization protocols for single-hop radio networks with no collision detection. IEEE Trans. Parallel Distrib. Syst. **11**(8), 851–863 (2000)
23. Nakano, K., Olariu, S.: Uniform leader election protocols for radio networks. IEEE Trans. Parallel Distrib. Syst. **13**(5), 516–526 (2002)
24. Perkins, C.E., et al.: Ad Hoc Networking, vol. 1. Addison-Wesley, Reading (2001)
25. Raab, M., Steger, A.: "Balls into Bins" — A Simple and Tight Analysis. In: Luby, M., Rolim, J.D.P., Serna, M. (eds.) RANDOM 1998. LNCS, vol. 1518, pp. 159–170. Springer, Heidelberg (1998). https://doi.org/10.1007/3-540-49543-6_13
26. Ramanathan, M.K., Ferreira, R.A., Jagannathan, S., Grama, A., Szpankowski, W.: Randomized leader election. Distrib. Comput. **19**(5–6), 403–418 (2007)
27. Ravelomanana, V.: Randomized initialization of a wireless multihop network. In: Proceedings of the 38th Annual Hawaii International Conference on System Sciences, pp. 324b–324b. IEEE (2005)
28. Scott, A., Jeavons, P., Xu, L.: Feedback from nature: an optimal distributed algorithm for maximal independent set selection. In: Proceedings of the 2013 ACM Symposium on Principles of Distributed Computing, pp. 147–156. ACM (2013)
29. Vlady, R.: Time-optimal and energy-efficient size approximation of radio networks. In: 2016 International Conference on Distributed Computing in Sensor Systems (DCOSS), pp. 233–237. IEEE (2016)