Efficient algorithms for leader election in radio networks*

Tomasz Jurdziński∗†, Mirosław Kutyłowski‡, and Jan Zatopiański†

∗ Institute of Computer Science, Technical University Chemnitz
†Institute of Computer Science, Wrocław University
‡Institute of Mathematics, Wrocław University of Technology and Dept. of Math. and Computer Science, A. Mickiewicz University, Poznań

Abstract

We present energy efficient algorithms for leader election in single channel single-hop radio networks with no collision detection. We present a deterministic solution with sublogarithmic energy cost (the best previous result was $O(\log n)$) and show a double logarithmic lower bound. We prove that this lower bound holds in a randomized case, in a certain sense.

For the case, when the number n of active stations can be approximated in advance, we show a randomized algorithm with energy consumption $O(\log^* n)$ that yields a result with high probability (the best previous result was $O(\log\log n)$).

1 Introduction

Radio network (RN, for short) is a model of growing importance that describes characteristics of networks of pocket radio stations. They are hand-held devices running on batteries with no central control. Moreover, the set of activated stations is unknown, so somehow the network has to organize itself. On the other hand, a RN requires no infrastructure and therefore it is attractive for several application areas such as disaster-relief, law-enforcement, logistics and so on.

Since communication means in RNs are quite different from the classical distributed systems, it is crucial to design efficient algorithms for fundamental tasks in this environment. Problems such as broadcasting, wakeup and leader election have been intensively studied (see [2, 8, 4, 5, 12] as starting points for references). Even if RN received much attention recently, algorithmic understanding of their capabilities still needs further development. In this paper, we design new methods for leader election in RN, improving significantly efficiency upon the previous results. We also prove lower bounds showing that some of our solutions must be close to the optimal ones.

1.1 Model

A RN is a distributed system that consists of computers equipped with radio transceivers, henceforth referred as *stations*. As usual for parallel and distributed computing, there are many differences between computing models called RN due to diverse technical details that may have immense impact on algorithmic issues in communication. Here we consider one of the weakest versions of the model in which leader election problem is relatively hard.

The stations of a RN are identical and execute the same code, they work synchronously (it has been pointed that this is realistic due to GPS technology). We split the execution time into

* partially supported by KBN, grants 7 T11C 032 21, 7 T11C 032 20, and DFG, grant GO 493/1-1

time slots, called here *steps*. Throughout this paper we assume that there is a single communication channel available to all stations (so we consider *single-channel, single-hop* RNs). Multiple channels might be used to speed up communication, however we shall see that even one channel is enough to achieve high performance.

During a step a station may either send (broadcast) a message or listen to the channel. If exactly one station sends, then all stations that listen at this moment receive the message sent. If at least two stations send, then a *collision* occurs. It is usually assumed that during a collision the messages sent are scrambled so that the stations cannot receive them. In this case the stations that are listening either are able to detect a collision (*collision detection RN*) or a collision is indistinguishable from the situation when no station is sending (*no collision-detection RN*, no-CD RN for short). Both models (and also hybrid cases) are interesting due to different technological situations; we concentrate on a more difficult task of efficient leader election for no-CD RN.

If a station sends a message during step i and no other station is sending at this moment, then we say that the station *succeeds* and that step i is *successful*.

The stations of a RN fall into two categories. Some of them are switched off and does not participate in the protocol. We call them *inactive*. The other stations are *active* and we assume that they stay active during the whole protocol. The set of all active stations is called *active set*. An active station might be either *awake* or *sleeping*. In the first case, a station sends a message or listens to the communication channel. We make here standard assumption that a station sending to the network can simultaneously listen. (This is not true for each technology, however we point out that this capability has deep consequences for communication complexity.) For further discussion on the RN model see e.g. [2, 1, 6, 10–14, 17].

1.2 Leader election problem

In *leader election problem* active stations have to choose a leader among themselves. That is, at the end of the protocol a single station has status *leader* and all other active stations have status *non-leader*.

In the literature, two kinds of randomized RN algorithms are considered: in the first scenario the number of active stations is unknown, in the second scenario this number is known or can be approximated within a constant factor. It is common to assume that the stations have no ID's or serial numbers – these have to assigned during a (non-trivial) initialization process.

In deterministic case, station ID's are necessary for symmetry breaking. So while designing deterministic leader election algorithms we assume that the stations have unique ID numbers from the range $[1 \dots n]$, where an arbitrary number of ID's might be used.

1.3 Complexity measures

There are two main complexity measures for RN algorithms: *time complexity* and *energy cost* [13]. Time complexity is the number of steps executed. Energy cost is the maximum over all stations of the number of steps in which the station is awake.

Low energy cost is a desired feature of RN algorithms: if a station is awake for a longer time, it may happen that it fails to operate due to batteries exhaustion. Thereby a station which is awake for much more steps than the other stations is likely to fail and for this reason the protocol may break down.

1.4 Previous results on leader election in RNs

An extremely simple algorithm for leader election in a network of n stations is in fact a core of Ethernet protocol [15]: each station sends with probability $\frac{1}{n}$. This step is repeated until some station succeeds. Expected runtime is $O(1)$, but it is necessary to know the number of active stations in advance. This has been improved by an algorithm [17], which runs a randomized algorithm to estimate the number of active stations. The expected runtime of this procedure is $\log\log n + O(1)$.

Leader election in RN received recently a lot of attention. Paper [6] describes a randomized algorithm that terminates in $O(\log^2 n)$ time with probability at least $1-\frac{1}{n}$ (the stations have no ID's). It is improved in [10]: the new algorithm terminates in time $O(\log n)$ with probability at least $1-\frac{1}{n}$, and in time $O(\log\log n)$ with probability $1-\frac{1}{\log n}$. Since the term $1-\frac{1}{\log n}$ becomes meaningless for small n, the same authors [11] show that with probability at least $1-\frac{1}{f}$ leader election can be terminated within time $O(\log\log n + \log f)$. Finally, in [14] the same bounds are obtained by a uniform algorithm, that is, an algorithm in which all active stations send with the same probability. In turn, paper [12] describes an oblivious leader election algorithm, i.e. such that these probabilities do not depend on computation history.

Energy efficient RN algorithms has been designed in [12]. A simple leader election deterministic algorithm is used there for RN initialization. It requires that the stations have unique ID numbers in the range $[1..n]$ (still, any subset of these ID's may correspond to active stations). The energy cost is $\log n + O(1)$, run time is $n + O(1)$. Further, they present a randomized algorithm that for n stations without ID's elects a leader in time $O(\log f)$ and energy $O(\log\log f + \frac{\log f}{\log n})$ with probability $1 - 1/f$ for any $f \geq 1$. Moreover, they present algorithms that work with the assumption that n is unknown and elect a leader within logarithmic energy cost.

1.5 New results

The first our algorithm is a kind of deterministic counterpart of the randomized algorithm [15]: we get energy cost only slightly bigger than the expected energy cost of the algorithm from [15]:

Theorem 1. *For no-CD RN, there is a deterministic leader election algorithm with energy cost* $\log^* n$ *that works for input configurations of the following form:*

- *the stations have unique names in the range* $1..n$,
- *the number of active stations is at least* $c \cdot n$ *for a fixed constant* $c > 0$.

(Recall that $\log^* n$ is a function defined by equality $\log^*(2^n) = \log^*(n) + 1$. It is a function that grows extremely slowly.) The techniques introduced in Theorem 1 are used to design a general leader election algorithm with sublogarithmic energy cost. This is the first known *deterministic* algorithm with such an energy cost.

Theorem 2. *For no-CD RN there is a deterministic leader election algorithm with energy cost* $o(\log^{\varepsilon} n)$ *for any* $\varepsilon > 0$.

Another application of Theorem 1 is an energy efficient randomized leader election algorithm that improves algorithm from [12] with $O(\log\log(n))$ energy cost:

Theorem 3. *Consider a no-CD RN that consists of* $N = \Theta(n)$ *anonymous stations (the number* n *is known to all active stations). For this RN, there is a randomized leader election algorithm that terminates after* $O(\log n)$ *steps with probability at least* $1-\frac{1}{n}$ *and has energy cost* $O(\log^* n)$.

In the second part of the paper we present lower bounds for energy cost of leader election in RNs. Almost no lower bounds have been previously known for RNs.

Theorem 4. *Every deterministic algorithm for leader election in no-CD RN consisting of stations with ID's from the set $\{1,\ldots,n\}$ where an arbitrary number of stations might be inactive requires $\Omega(\log\log n/\log\log\log n)$ energy cost. (The same holds for CD RNs.)*

This result can be extended to randomized algorithms in the following way:

Theorem 5. *The lower bound from Theorem 4 holds for randomized leader election algorithms that err with probability at most $\frac{1}{n^3}$.*

For completeness, we also consider time complexity of leader election:

Theorem 6. *Every deterministic algorithm for leader election in no-CD RN requires at least $n-1$ time in the worst case.*

2 Log-star randomized leader election algorithm for RN

2.1 Logarithmic energy cost solution

Let us recall a deterministic leader election algorithm from [14]. Consider a binary tree with n leaves labeled 1 through n. Leaf j corresponds to a station with ID j (provided that station j is active, otherwise there is no station corresponding to this leaf). We label also internal nodes of the tree so that if a node has label i and its sons have labels i_1, i_2, then $i_i, i_2 < i$. The algorithm computes the leader in a subtree rooted at node i at step $i-n$: the leader of the subtree rooted in the left son of node i sends a message and the leader of the subtree rooted in the right son of node i is listening. If such a message is actually sent, then the leader of the right subtree changes it status to non-leader and the leader of the left subtree becomes the leader of the subtree rooted at node i. If no message is sent, then it means that there is no active station corresponding to a leaf in the left subtree and the leader of the right subtree (if exists) becomes the leader of the subtree rooted at node i.

It is easy to see that the leader elected by the algorithm is awake for $\log n + O(1)$ steps and energy cost is $\log n + O(1)$. By doubling time and energy cost one can achieve that the leader knows all other active stations. For this purpose we attach two steps to each node i of the tree: first the leader of the subtree rooted at the left son of node i is sending a message informing which stations are active in its subtree and the leader of the right subtree is listening, in the second step the roles are reversed. (The information on active stations can be encoded as a binary string, each bit corresponding to one leaf, and equal to 1, if the corresponding station is active). This modification will be called *Basic Algorithm* henceforth.

2.2 Log-star deterministic leader election for special cases

In this section we rebuild Basic Algorithm in order to prove Theorem 1. The key point is that large energy consumption occurs only at these stations that come close to being a leader. The worst energy consumption occurs for the leader itself: it has to go along a path in a tree of logarithmic depth where each node means being awake for $O(1)$ steps. In our algorithm the stations that loose serve the stations that have won with them. Such stations, called *slaves*, take over the tasks of their *masters* and thereby reduce the number of steps when the masters are awake.

For the sake of presentation we assume that at the beginning each active station has $2/c$ slaves (these slaves can be virtual and emulated by the station itself, this increases the number of steps when the station is awake by a constant multiplicative factor only).
Our algorithm has $O(\log^* n)$ stages. The following invariant will hold at a stage i:

- there are x_i stations called *masters* participating in the phase, some of them called *active*, the rest is called *passive* (a passive master is either nonexisting leader or a leader with not enough slaves);
- there are at least $c_i \cdot x_i$ active masters;
- each active master has a group of at least s_i *slaves* - stations that serve this master;
- the slaves of the same master are labeled with consecutive numbers $1, 2, \ldots$, each slave knows its label and its master;
- the masters are split into disjoint groups of size 2^{s_i}, every group of masters elects a leader that becomes a master of the next stage of the algorithm;
- after electing a leader in a group all other masters and slaves of the group become slaves of the leader; the leaders elected with less than s_{i+1} slaves become passive.

Let $x_1 = n$, $s_1 = 2/c$, $c_1 = c$, and the following condition be fulfilled for each i (may be except the last phase):

$$c_{i+1} = c_i/2 \ , \quad x_{i+1} = x_i/2^{s_i} \ , \quad s_{i+1} = c_{i+1} \cdot s_i \cdot 2^{s_i} \ . \tag{1}$$

The way in which the slaves can help their master is the following. Assume that we perform Basic Algorithm in a group of 2^s stations, where each active station has at least s slaves. Roughly speaking, the step in which a master p competes to be a winner in a subtree of height i is emulated by the ith slave of p. More precisely, consider a stage of Basic Algorithm in which the winners of two subtrees L and R of height $i-1$ compete to become the leader in a tree of height i. The leaders of R and L are represented by their slaves with index $i-1$. First, the slave from R sends a message and the slave from L listens, in the next step the roles are reversed. Then the slave that has won for its master sends a message. All slaves from $L \cup R$ with label i listen to it. They fall asleep till the end of the phase except for the slave i of the leader. This slave takes over and represents the leader of $L \cup R$ during the next step of Basic Algorithm.

At the end of simulation of Basic Algorithm in a group of 2^s masters we have a leader E. The slave of E with index s knows all masters that have lost against E, encodes this information in a binary string of length 2^s and broadcasts while all masters and slaves of the group electing E are listening. They become the slaves of E for the next phase.

Now it is easy to establish the invariant of the algorithm. At the beginning of phase $i+1$ there are at least $c_i \cdot x_i$ active masters. So at least $\frac{c_i}{2} \cdot \frac{x_i}{2^{s_i}}$ groups considered at phase $i+1$ contain at least $c_i/2 \cdot 2^{s_i}$ active masters. Indeed, the worst case occurs, when we first put $c_i 2^{s_i}/2 - 1$ active masters in each group (so that it does not suffice to elect a leader with sufficiently many slaves) and put all remaining active masters into as few groups as possible (each of these group will elect a leader with enough slaves). Since the number of remaining active masters was at least $c_i/2 \cdot x_i$; the number of these groups will be $\frac{c_i \cdot x_i}{2 \cdot 2^{s_i}} = c_{i+1} x_{i+1}$.

An important point is that a lower bound on the average number of active masters in a group increases. Indeed, this average number is not less than $s_i \cdot c_i$ at phase $i+1$. In the next phase, the bound changes to $s_{i+1} \cdot c_{i+1} = c_i/2 \cdot s_i \cdot 2^{s_i} \cdot c_i/2 \geq 2^{s_i - 2} \cdot c_i \geq s_i \cdot c_i$. At the last phase there is one group, so the average number equals the number of active masters, and there is at least one such master at the last phase. So the leader will be elected.

It follows from equation (1) that s_i grows so that for $k = O(\log^* n)$ we end up with a single group and perform the last phase. Energy cost is $O(\log^* n)$, since each station is awake for $O(1)$ steps during each phase. The execution time of the algorithm is $O(n)$, since our algorithm emulates the Basic Algorithm (which needs time $O(n)$) and each step is simulated by $O(1)$ steps. This concludes the proof of Theorem 1.

2.3 General case deterministic sublogarithmic leader election

Our deterministic algorithm establishing Theorem 2 can be described as a recursive procedure $S_n(k)$. The parameter k (which is an increasing function $k : \mathbb{N} \to \mathbb{N}$) is responsible for *divide and conquer* feature of the algorithm and can be tuned separately. Below, the notions of slaves has similar meaning as in Section 2.2.

Algorithm $\mathbf{S_n(k)}$ The outcome of the algorithm is a leader. If active stations have slaves, then all slaves and the stations that have lost become slaves of the leader.
Phase 1. The stations are split into groups of size $k = k(n)$ based on the ID's of the stations. During step i every active station of group i sends a message. If there is an active station in a group, then there are two possible outcomes: (a) one station in a group is active and it can hear its own message, (b) at least two stations are active and they recognize that their messages have been scrambled.
Phase 2. In the groups, where the case (a) has occurred, the only active station is a leader. These leaders wait until phase 4.
Phase 3. Each group of size $k(n)$, where during phase 1 the case (b) has occurred, runs recursively algorithm $S_{k(n)}(k)$. When it is finished, then each leader elected has a slave that is used in Phase 4 for reducing the energy cost of its master by half.
Phase 4. A leader is elected from the leaders of Phases 2 and 3, by an execution of $S_{n/k(n)}(k)$ (with ID's equal to the numbers of the groups to which appropriate leaders belong).

It follows immediately from the description of the algorithm that the energy cost E of the algorithm S satisfies the following relation:

$$E(S_n(k)) = O(1) + \max\left(E(S_{n/k(n)}(k)), E(S_{k(n)}(k)) + \tfrac{1}{2}E(S_{n/k(n)}(k))\,.\right)$$

Let $t \in \mathbb{N}$ and $\delta = 2^{-t}$. For a parameter $k(n) = n^{\delta}$ we can check that a solution to this relation satisfies $E(S_n(k)) = O\left((\log n)^{1/t}\right)$. This concludes the proof of Theorem 2.

2.4 Fast randomized leader election

Theorem 3 follows easily from Theorem 1. The algorithm consists of two phases:

- the active stations "throw the balls into $k = c \cdot \log n$ bins". That is, there are k *trial steps*; during each trial step, each station that has not sent a message yet decides to send with probability $\frac{1}{n}$, otherwise it sleeps at this moment.
- Only the stations that have succeeded during trial steps participate in the second phase of the algorithm. Each of them has a temporary ID, which is the number of a trial step during which it was the only station sending a message. During this phase, the stations execute the algorithm from Theorem 1.

Each station is awake for $O(1)$ steps during the first phase and $O(\log^* n)$ steps during the second phase. So energy cost is $O(\log^* n)$.

The only problem is that the algorithm may fail to elect a leader. It may happen if the number of stations that participate in the second phase is too small. It is easy to see that the probability that some station succeeds during the first trial step equals $N \cdot \frac{1}{n} \cdot (1-\frac{1}{n})^{N-1} \geq \frac{N}{n} \cdot (1-\frac{1}{n})^{\max(N,n)} = \Theta(1)$. For the next trial steps this probability would be the same if we neglect the effect that some stations already tried to send a message. So, k trial steps resemble very much k independent Poisson trials with success probability $\Theta(1)$. Based on Chernoff Bounds [16] we can argue that the number of successes is $\Theta(k)$ with probability at least $1-\frac{1}{n}$.

We refer to [7] for a quite technical but rigorous probability analysis stating that the number of successes during k trial steps is $\Theta(k)$ with probability at least $1-\frac{1}{n}$ also in the case when the random experiments are stochastically dependent, as in our algorithm. Note that this dependency is absolutely necessary in order to achieve low energy cost.

3 Lower bounds

3.1 Lower bound for deterministic time complexity

In this section we prove Theorem 6. Let $\mathcal{A}$ be a leader election algorithm running in time T. Let $A_0 = \{1,\ldots,n\}$. We define a descending family of active sets $A_0 \supset A_1 \supset \ldots A_{n-1}$; with each $i = 0,\ldots,n-1$ we associate a time step $t_i \in [T] \cup \{\infty\}$ that corresponds to the change from the active set A_i to the active set A_{i+1}. We show that $t_i \neq t_j$ for $i \neq j$, so we get $T \geq n-1$, and thereby the claim of Theorem 6.

Consider execution of $\mathcal{A}$ for active set A_i. Let t_i be the minimal t such that during execution of $\mathcal{A}$ for active set A_i some station, say p_i, is successful at step t. If there is no successful step during this execution, then we put $t_i = \infty$ and p_i is defined as the leader elected for A_i. Then we set $A_{i+1} = A_i \setminus \{p_i\}$.

Now suppose that $t_i = t_j$ for $i < j$. If $t_i = \infty$, then during executions for A_i and for A_j, there is no successful step. So, for an active station from A_j, its computation history is the same for active sets A_i and A_j. This holds, in particular, for the leader elected by A_j. So this station would consider itself as a leader in the case that the active set equals A_i. However, according to the definition, the leader of A_i is not in A_s for any $s > i$.

Now assume that $t_i = t_j \neq \infty$. According to the definition, during the computation for active set A_i, active stations can hear only noise up to step $t_i - 1$. For the same reason, this is also true for A_j. Hence the state of station p_j after step $t_i - 1$ is the same for active sets A_i and A_j. So station p_j decides to send at step t_i during computation for active set A_i. But p_i does the same and since $p_i \notin A_{i+1}$, the stations p_i and p_j are different! We get a contradiction with the definition which states that there is no collision at step t_i for active set A_i.

3.2 Lower bound for deterministic energy cost

In this section we prove Theorem 6. The idea is to define sets $A_0, A_1, \ldots$ so that after considering step i we confine the set of active stations to all subsets of A_i; we shall assume that the active set consists of at least two stations. The key invariant we are aiming for is that for $j \in A_i$ station j has the same communication history up to step i for each active set $I \subseteq A_i$ with $|I| \geq 2$, $j \in I$. We also define weights $w_i(j)$ for elements $j \in A_i$ and $n_i = \sum_{j \in A_i} w_i(j)$. We start with $A_0 = \{1,\ldots n\}$, $n_0 = n$ and $w_0(j) = 1$ for $j \leq n$. Let $\pi_n(m) = m^{1/\log\log n}$.

Now consider step i and assume that A_{i-1} has been already defined. Let S_i denote the set of all stations $j \in A_{i-1}$ that would send at step i (provided that station j is active). Similarly, R_i denotes the set of stations that would listen at this step. Consider two cases:

Case (X): $|S_i \cup R_i| \le 2\pi_n(n_i)$. Then $A_i := A_{i-1} \setminus (S_i \cup R_i) \cup \{j\}$ where j is an element of $S_i \cup R_i$ such that $w_{i-1}(j) \ge w_{i-1}(m)$ for every $m \in S_i \cup R_i$. Let $n_i := n_{i-1}$, $w_i(l) := w_{i-1}(l)$ for $l \in A_i$, except that $w_i(j) = \sum_{l \in S_i \cup R_i} w_{i-1}(l)$.
Case (Y): $|S_i \cup R_i| > 2\pi_n(n_i)$. If $|S_i| > |R_i|$, then $A_i = S_i$, otherwise $A_i = R_i$. We put $n_i := |A_i|$, $w_i(j) = 1$ for every $j \in A_i$.

By construction, we get immediately that $n_i = \sum_{j \in A_i} w_i(j)$ for each i. Also a simple induction shows the following property: Let $i_0 < i$ be such that $i_0 = 0$ or rule (Y) is applied at step i_0, and the rule (X) is applied for steps $i_0+1, i_0+2, \ldots, i$. Let $m = |A_{i_0}|$. Then $w_i(j) \le (2\pi_n(|A_{i_0}|))^{l(j)}$ for every $j \in A_i$, where $l(j)$ is the number of steps among $i_0, i_0+1, \ldots, i$ when j is awake. The proof is based on the observation that we leave a station with the maximal weight in every application of (X) and add to its weight the weights of at most $2\pi_n(|A_i|) - 1$ other elements. The following proposition establishes other key properties:

Proposition 1. *(1) For every $j \in A_i$ and $I_1, I_2 \subseteq A_i$ such that $\{j\} \subsetneq I_1$ and $\{j\} \subsetneq I_2$, the station j is unable to distinguish between active sets I_1 and I_2 during the first i steps.*
(2) If (Y) is applied l times during the first i steps, then $n_i \ge n^{(1/\log\log n)^l}$ and for $j \in A_i$ station j is awake at least l times during the first i steps for every active set $I \subseteq A_i$ such that $j \in I$ and $|I| \ge 2$.

Proof. The proof of (1) is by induction. The case $i = 0$ is obvious. Assume that the claim holds for every $i' < i$. If we apply (X) at step i, then only one station from A_i is awake at step i, so no station from A_i gets any new information about other elements of A_i. There are two cases, when (Y) is applied. If $A_i = R_i$, then all elements of A_i are listening at step i so they gain no new information. If $A_i = S_i$, then all elements of A_i are sending at step i, so if there are at least two active stations in A_i there is a collision and the processors hear the same.

Part (2) follows directly from (1) and the construction. □

Observe that if $|A_i| > 2$ and the active set is a subset of A_i with at least two elements, then the algorithm cannot finish after step i. Indeed, let $a, b, c \in A_i$, let c be a leader if the active set is equal to $\{a, b, c\}$, and a is the leader if the active set is $\{a, b\}$. Then the station a cannot decide whether to become a leader since its communication history is the same in both cases.

We can continue the process of defining A_i's as long as $|A_i| > 2$. On the other hand, rule (X) may be applied as long as $2\pi_n(n_i) \ge 1$ and rule (Y) may be applied provided that $\pi_n(n_i) > 2$. So rule (Y) may be applied at most l times, where l is maximal such that $n^{(1/\log\log n)^l} > 2$. So $l = \lfloor \log\log n / (\log\log\log n) \rfloor$. If (Y) were applied l times up to the step i, then each element of A_i awake at least l times provided there are at least two active stations in A_i. So in this case the lower bound would hold. Assume now that the number of applications of rule (Y) is $k < l$. Let i be the step in which (Y) is applied for the last time. Then $n_{i'} = n_i \ge n^{(1/\log\log n)^k} \ge \log n$ for every $i' \ge i$. Starting with n_i stations we can apply (X) at least $n_i/(2\pi_n(n_i))$ times before be obtain A_l of size not larger than two. Finally, there are at most two stations in A_l and the sum of their weights is equal to n_i. So one of elements of A_l has weight at least $n_i/2$. On the other hand, its weight is not bigger than $(2\pi_n(n_i))^t$, where t denote its energy cost during steps $i, i+1, \ldots, i'$. So $n_i/2 \le (2\pi_n(n_i))^t$, and $t = \Omega(\log\log n)$ since $n_i \ge \log n$.

3.3 Lower bounds for randomized algorithms

Here we prove Theorem 5. We apply a technique from [9] and Yao's Min-Max Principle [18, 16]. First we define probability distribution $\mathcal{P}$ over all input configurations. Let $\mathcal{Z}$ be the set of all configurations such that two or three stations are active. We define $\mathcal{P}(\delta) = \frac{1}{h}$, where $h = \frac{1}{2}n(n-1) + \frac{1}{6}n(n-1)(n-2)$, if $\delta \in \mathcal{Z}$. Otherwise, $\mathcal{P}(\delta) = 0$.

Let $\mathcal{A}$ be a randomized leader election algorithm that errs with probability less than $\frac{1}{n^3}$. By Yao's Principle, there is a deterministic algorithm $\mathcal{A}'$ such that:

- $\mathcal{A}'$ has energy cost bounded by energy cost of $\mathcal{A}$;
- the fraction of inputs (according to probability distribution $\mathcal{P}$) for which $\mathcal{A}'$ does not terminate in a correct state is at most n^{-3}.

Now observe that since $\frac{1}{h} > \frac{1}{n^3}$, for each input configuration $\delta \in \mathcal{Z}$ the outcome of algorithm $\mathcal{A}'$ must be correct.

Now, for algorithm $\mathcal{A}'$ we apply the same proof as for Theorem 4. Indeed, the lower bounds argument was based on the algorithm behavior for situations in which two or 3 stations are active.

References

1. J.L. Bordim, J. Cui, T. Hayashi, K. Nakano, S. Olariu, *Energy efficient initialization protocols for ad-hoc radio networks*, ISAAC'99, LNCS 1741, Springer-Verlag, 215–224.
2. B. S. Chlebus, *Randomized communication in radio networks*, a chapter in „Handbook on Randomized Computing" P. M. Pardalos, S. Rajasekaran, J. H. Reif, J. D. P. Rolim, (Eds.), Kluwer Academic Publishers, to appear.
3. P. Fraigniaud, A. Pelc, D. Peleg, S. Perennes, *Assigning labels in unknown anonymous networks*, Distributed Computing 14(3), 163-183 (2001).
4. L. Gąsieniec, A.Lingas, *On adaptive deterministic gossiping in ad hoc radio networks*, in Proc. ACM-SIAM SODA '2002.
5. L. Gąsieniec, A. Pelc, D. Peleg, *The wakeup problem in synchronous broadcast systems*, PODC 2000, 113–121. Journal version: SIAM Journal on Computing 14(2) , 207–222 (2001).
6. T. Hayashi, K. Nakano, S. Olariu, *Randomized initialization protocols for Packet Radio Networks*, IPPS 1999 (International Parallel Processing Symposium), IEEE 1999, 544–548.
7. T. Jurdziński, M. Kutyłowski, J. Zatopiański, *Energy-Efficient Size Approximation for Radio Networks with no Collision Detection*, submitted.
8. E. Kushilevitz, Y. Mansour, *Computation in Noisy Radio Networks*, Proc. ACM-SIAM SODA '98, 236–243.
9. M. Kutyłowski, K. Loryś, *Limitations of the QRQW and EREW PRAM models*, Foundations of Software Technology and Theoretical Computer Science (FSTTCS)'96, LNCS 1180, Springer-Verlag, 310-321.
10. K. Nakano, S. Olariu, *Randomized $O(\log\log n)$-round leader election protocols in radio networks*, ISAAC'98, LNCS 1533, Springer-Verlag, 209-218.
11. K. Nakano, S. Olariu, *Randomized leader election protocols for ad-hoc networks*, SIROCCO'2000, Carleton Scientific, 253-267.
12. K. Nakano, S. Olariu, *Randomized Leader Election Protocols in Radio Networks with No Collision Detection.* ISAAC'2000, LNCS 1969, Springer-Verlag, 362–373.
13. K. Nakano, S. Olariu, *Energy Efficient Initialization Protocols for Radio Networks with no Collision Detection*, ICPP'2000, IEEE, 263–270.
14. K. Nakano, S. Olariu, *Uniform Leader Election Protocols for Radio Networks*, ICPP'2001, IEEE.

15. R. M. Metcalfe, D. R. Boggs, *Ethernet: distributed packet switching for local computer networks*, Communication of the ACM 19 (1976), 395-404.
16. R. Motwani, P. Raghavan, *Randomized Algorithms*, Cambridge University Press, 1995.
17. D.E. Willard, *Log-logarithmic selection resolution protocols in multiple access channel*, SIAM Journal on Computing 15 (1986) , 468-477.
18. A. C-C. Yao, *Probabilistic computations: towards a unified measure of complexity*, ACM-SIAM FOCS '77, 222-227.