Secure Initialization in Single-Hop Radio Networks*

Mirosław Kutyłowski[1] and Wojciech Rutkowski[1]
Miroslaw.Kutylowski@pwr.wroc.pl, Wojciech.Rutkowski@pwr.wroc.pl

Institute of Mathematics, Wrocław University of Technology

Abstract. We consider single-hop radio networks, where collisions in the shared channel cannot be detected (no-CD model). A radio channel can be accessed by an adversary trying to degrade functionality of the network, so we are interested in algorithms that work in the presence of an adversary, who knows the algorithm executed and may try make it faulty by injecting own messages. We also focus on algorithms that are time and energy efficient.
We propose a randomized initialization algorithm for a single-hop no-CD radio network. The algorithm has time complexity $O(N)$ and energy cost $O(\sqrt{\log N})$. This is not much worse than the best fragile algorithms constructed so far ($O(N)$ in time complexity and $O(\log\log N)$ energy cost). Our algorithm succeeds with probability $1-2^{-\Omega(\sqrt{\log N})}$ in presence of an adversary, who has energy cost $\Theta(\log N)$.

1 Introduction

Networks based on a radio communication channel have an Achilles' heel – they can be easily attacked by an adversary who causes transmission collisions. If communication algorithms do not take such attacks into account, it may happen that anybody can make algorithm execution faulty. These problems are particularly acute for algorithms that are used for self-organization of the network. One of the most important fundamental tasks is initialization – assigning consecutive labels to the stations in the network.

Our goal is to design an initialization algorithm that is both immune against an adversary and remains efficient in terms of execution time and energy cost.

Radio Network Model A radio network (RN for short) consists of processing units, called *stations*. The stations communicate through a single shared communication channel. Since a shared communication channel may be implemented by a radio channel, they are called radio networks. Since in general neither the number of stations nor their ID's are known, they fall into category of *ad hoc networks*.

There are many potential applications of such networks, sensor networks for collecting environment data [4] is just one of the examples.

In this paper we consider *single-hop* RN's: if a station is sending a message any other station receives it, provided that its receiver is on. So single-hop RN's are networks working within a small proximity. The main problem with RN's is that if two

* partially supported by Polish Committee for Scientific Research, grant 7 T11C 032 20 in years 2002-2003, and by grant 3 T11C 033 26 in year 2004. Copyright Springer-Verlag, LNCS 3113

stations are sending simultaneously, then a *collision* occurs. In such a case the messages get scrambled. We assume that the result is indistinguishable from a random noise – the model is called *no-CD* RN. This is a pessimistic assumption, since occurrence of a collision may also be a source of valuable information, particularly during the network self-organization phase. In accordance to physical reality and industrial standards, we also assume that a station cannot transmit and listen at the same time.

Communication between stations occur within *time slots* that are the same for all stations. This is possible, since the stations may be synchronized via a radio signal from a common clock. A GPS system can be used for this purpose as well. During a time slot a station either transmits or listens or works only internally. In the first two cases we say that the station is *awake*, in the last one – it is *asleep*.

There are different scenarios considered in the literature regarding the stations and their knowledge: either the number of stations is known, or the number of stations is known up to some multiplicative constant, or the stations have no idea about the number of other active stations. In this paper we design an initialization algorithm, so we assume that stations have no IDs, or they have IDs in a big range $1..R$ where $N = o(R)$.

Further information concerning ad hoc networks can be found in a handbook [8].

Complexity measures Given a single shared communication channel two factors may become a bottleneck: execution time and energy cost of transmission. While the issue of time complexity is obvious, energy cost needs some explanation. Many networks consist of battery operated devices. Re-charging or exchanging these batteries might cause many problems – sometimes it is impossible. Even if battery can be replaced or recharged, an interruption due to battery exhaustion might cause problems. So there is a high motivation to design energy efficient algorithms. It turns out that the most costly (in terms of energy consumption) is listening and transmitting messages. On the other hand, energy consumption of processor and sensors can be neglected in such systems.

We define the *energy cost of a station* as the number of time slots, during which a station was awake. *Energy cost of an algorithm* is the maximum energy cost over all stations involved in algorithm execution. For a randomized algorithm $\mathcal{A}$, we say that with probability p the energy cost is $f(n)$, if an execution of $\mathcal{A}$ has energy cost bounded by $f(n)$ with probability p.

Initialization Problem Some algorithms running on RN's in a multiprocessor environment require that all stations are labeled with consecutive numbers. So we have to solve the following *Initialization Problem*:

> A RN network is given consisting of n active stations. Each active station has no knowledge which stations are active (except itself) and does not know n. To each active station assign a number in the range $[1,\ldots,n]$ so that each number $i \in [1,\ldots,n]$ is used exactly once. After this procedure each active station knows the number assigned to it.

In this paper we may assume that the stations know a number N such that $cN \leq n \leq N$ with high probability for some constant c. Let us remark that there is an efficient size approximation algorithms which finds such an N (see [2]). It seems that they can be tuned to an adversary immune version.

Previous results There are many initialization protocols in the literature so far. As we work in the no-CD RN model, we can limit to few ones.

The first algorithm on the no-CD model was designed by Nakano and Olariu [6]. However this method does not succeed when the sending station cannot check the channel status (according to IEEE 802.11 standard). They relied on the tree leader election algorithm, which help them build a sparse table. Then they run a prefix sum algorithm to count the number of alive stations. The algorithm was divided into phases. First, all the stations try to get a temp-ID in the range of 1 to $8n$. Later, all unsuccessful stations try to do it in the range $[8n+1..12n]$, geometrically decreasing the length of the interval range. Then, after shuffling the first $8n$ temp-ID with the rest we get a dense distribution of ID's (in the sense that every interval of length $O(\log n)$ has at least one station with temp ID assigned). So the tree prefix sum algorithm is performed in each of $\frac{n}{\log n}$ groups. Then all the groups perform again this tree prefix sum algorithm and calculate their own ID. As it is obvious that tree algorithms are very fragile to adversary attacks, this algorithm cannot be considered as adversary immune.

A modification of the previous algorithm was presented by Jurdziński, Kutyłowski and Zatopiański in [3] in the no-CD model. The purpose of [3] was partial initialization in which stations are organized in pairs, so that listening while transmitting a message can be emulated. In the first phase authors tried to pair the stations and hence run the initialization algorithm from [6]. Of course that could not initiate all the network, however a vital part of it. Then these stations were used as the echo for the rest of the algorithm and thus a CD model algorithm could be performed, starting from some ID, next after the maximum ID already assigned in the first phase. This algorithm cannot also be regarded as adversary immune.

Another paper [5] presented an adversary immune algorithm that initializes a small number of stations $O(\log N)$. We extend this result to full initialization.

Security issues Practical applications of RN's must take into account various dangers resulting from the communication model. Although in a wired network an adversary attacking the system might inject packets with manipulated data concerning their origin, it is possible to trace their origin to some extent. In wireless networks it is much harder. As long as no special physical equipment is deployed, the mobile intruder is safe. Almost all algorithms built so far for RN's disregard this issue. This is a severe drawback, since for many applications ad hoc networks must offer a certain security level.

New Result Our main goal is to design an initialization algorithm that is energy efficient and tolerates an adversary having limited energy resources. The secondary goal is to preserve linear execution time. (Obviously, $\Omega(n)$ steps are necessary for initialization of n stations in our model.) We get the following result:

Theorem 1. *Consider a single-hop no-CD radio network consisting of $n = \Theta(N)$, $n \leq N$ stations sharing a secret key. Assume that the stations are not initialized with any ID's. Then it is possible to initialize all stations with energy cost $O(\sqrt{\log N})$ within time $O(N)$, so that the outcome is faulty with probability $O(2^{-\sqrt{\log N}})$ in a presence of an adversary which has energy resources of $O(\log N)$.*

So we want to extend the leader election result from [5] using the same energy resources. Initialization is a much harder task, since the consecutive numbers have to be assigned to all stations and not only to a single one.

2 Basic Techniques

In this section we introduce some "building blocks" of our algorithm. We put them separately in order to improve readability of the main part of algorithm description. These techniques are simple, some of them are well-known, standard tools in this area.

Cryptographic Assumptions We assume that the stations participating in the initialization procedure share a secret s unknown to the adversary, deployed during initialization of these stations. The secret s can be used to derive further keys used for protecting communication. Let us say that at time t the stations sharing s may use key $s_t = F(s,t)$, where F is an appropriate cryptographic pseudorandom one-way string generator. (This protects also from replay attack - repeating by an adversary intercepted transmissions.)

We assume that all messages sent during the initialization procedure are encrypted with such keys so that the messages transmitted cannot be distinguished from a random noise by an adversary. Even on weak devices encrypting with stream ciphers seems to be plausible, since information volume is low and pseudorandom bit streams can be prepared in advance.

Since encryption hides not only communication contents but also communication occurrence from the adversary, we may assume that the knowledge of the adversary is confined to the knowledge of the algorithm executed, its starting point and the approximate number of participating stations. The adversary cannot fake or modify messages of the protocol; the adversary may only try to cause collisions and in this way to break down the protocol. This assumption models also a broad family of transmission faults.

Initial selection As one of the building blocks we use the following simple procedure well known from many communication protocols [6, 7]. By a *participant* we mean any station which performs the protocol. Let us assume that there are $n = \Theta(N)$ participants and N is known to all stations (including the adversary); b is a protocol parameter. First each participant P tosses a coin and with probability 0.5 it becomes a *sender*, otherwise it becomes a *receiver*. Then P participates in b rounds. During a round a sender P first decides to be active with probability $1/N$, if it is not active it skips the rest of the round. During a round an active sender P executes the following steps:

step 1: P transmits a random message,
step 2: P listens,
step 3: if P has received its own message in step 2, then it transmits it again.

During a round an active receiver P executes the following steps:

step 1: P listens,
step 2: P transmits the message received in the previous step,

step 3: P listens.

If the sender gets its own message, it knows that there was no collision and exactly one receiver has responded. At step 3 the receiver checks whether there was no collision. In this case we say that a pair of participants, the sender and the receiver, succeeds, and we assign them the round number as an ID number. Such an event occurs with probability approximately $\frac{1}{e^2}$.

In order to keep energy cost $O(1)$ we also assume that a participant remains inactive after the first round in which it was active (no matter whether this round ended with a success or not). Despite that the process is no more a sequence of independent Bernoulli trials, one can prove that the number of successes does not change substantially [2].

Note that initial selection is resistant against an adversary: we assume that its energy capacity is $O(\log N)$, so for $d \gg \log N$, the adversary can attack only a small fraction of rounds.

Relay Procedure Assume that some stations have unique labels in the range $[1..d]$, say $s_1, \ldots, s_g$, where $s_i < s_j$ for $i < j$. *Relay procedure* informs the station with label $s = s_i$ about number i in the following way: at step $2s-1$ a station P with label s listens – if it receives a message $i-1$, then it means that $s = s_i$. At step $2s$ the station responds by sending $i-1$ again. During the following steps $2s+1$, $2s+3$, $\ldots$ station P keeps sending i until at some step $2s+2t$ it receives response i. This means that $s_{i+1} = s+t$ and that the station with label $s+t$ takes over.

Relay procedure may be used to initialize the stations that have succeeded during initial selection. The procedure has two major drawbacks: First, if $s_{i+1} - s_i$ is large, then the station with label s_i consumes a lot of energy. Second, the adversary can make a collision in order to prevent delivery of an acknowledgment. After that the whole computation would be faulty.

Time Windows [5] Suppose that stations A and B share a common secret s and that A has to sent a message to station B at some step i of the algorithm executed. Assume that a time window consisting of r consecutive steps is reserved for this purpose. The problem is that an adversary may try to make a collision and destroy the transmission in this way. In order to avoid such a collision the moment of transmission for step i is determined as $t = f(s,i)$ for some secure pseudo-random function f. For stations A and B the choice is deterministic, while the adversary cannot determine t.

If the adversary detects a transmission, then it is too late for disturbing it. So the adversary may only attack by transmitting at random moments. So, with energy cost m, probability of collision equals $\frac{m}{r}$. So by having a large time window r we may keep collision probability low.

The main drawback of this method is a trade-off between security and time complexity: executing each step of a protocol with a time window of size r increases time complexity by a factor of r. So it is not directly suited to achieve time optimal solutions.

Interleaving Technique Now we introduce a simple but very useful trick that contributes a lot to efficiency of our solution. Assume we have to perform the same algo-

rithm $\mathcal{A}$ in k independent groups. We can run them concurrently as in *time division multiple access* (TDMA mode): the ith step of group j will be executed in step $(i-1)\cdot k+j$. If $\mathcal{A}$ tolerates a small number of collisions caused by an adversary, then we can modify TDMA in order to achieve robustness against the adversary without applying time windows technique. Namely, we use a secure pseudorandom permutation generator π: for step number i and secret s it outputs a permutation $\pi(i,s) \in \mathbb{S}_k$. Then step i of algorithm $\mathcal{A}$ in group j is executed at step $(i-1)\cdot k+\pi(i,s)(j)$. So, from a point of view of a single run of $\mathcal{A}$, its every step is executed with time window of size k.

Interleaving technique has one major advantage over time windows: the same effect of confusing an adversary attacking a single groups is achieved, while we do not waste time for leaving most time slots unused for communication. Of course, now the adversary knows that it always hits somebody, but the chances that the same group is attacked repeatedly are bounded.

Adversary Immune Leader Election Now we describe some features of the leader election algorithm presented in [5]. It has time complexity $O(\log^3 N)$, but essentially the number of steps executed is $O(\log^{3/2} N)$, where each step is executed in a window of size $O(\log^{3/2} N)$. An introductory phase of this algorithm is initial selection executed with parameter $\Phi = \Theta(\log^{1.5} N)$. Later phases are executed by the stations that have succeeded during initial selection.

The main part of the algorithm consists of *group election* executed sequentially in $\Theta(\sqrt{\log N})$ groups, each working on a group of $d = \Theta(\log N)$ consecutive ID's. Once a group elects a leader it prevents all later groups from electing a leader – in this way at most one group elects a leader that becomes the final leader elected by the algorithm. Disabling later groups is achieved by appropriate collisions. (This seems to contradict the assumption that the algorithm is collisions immune; however, the group that has elected a leader knows a secret value used by the following groups to make secret pseudorandom choices of transmission times.)

Group election procedure consists of two phases: the first one is called *building chains*, the second one – *merging chains*. The first phase is a modified relay procedure. The change is that once a station seeks long for the next active station ($\Theta(\sqrt{\log N})$ times), it stops the search. On the other hand, if a station i does not hear any predecessor at step $2i-1$, it assumes that either there is no predecessor or it has stopped due to high energy usage. In this way, the modified relay procedure might fail to link all active stations, and a number of *chains* of linked stations may emerge. Each chain consists of all active stations with ID's from some interval and the station with the highest ID in a chain knows all ID's of active stations in this interval. The second modification in the relay procedure is that one can make it immune against collisions – instead of faulty results (due to a collision caused by an adversary), the worst that can happen is stopping a chain at the moment of attack. This goal is achieved by a simple modification in the relay procedure. The stations work in pairs (as given by the initial selection). Instead of sending a message once, it is sent twice by both stations of a pair. By a simple case inspection one can show that no inconsistency would appear provided that the stations apply a decision procedure described in [5].

During the merging phase the goal is to join the chains. We assume that before the chains are built, the ID's are permuted according to some pseudorandom permutation computed with the secret shared by the stations. So even if the adversary caused a certain number of collisions during the initial selection, the ID's excluded by him would be dispersed over all possible ID's. So the allocation of unused ID's (due to adversary and to lack of success) can be treated as random from an external point of view. Probability that a chain gets broken due to a gap of size $\Omega(\sqrt{\log N})$ is $2^{-\Omega(\sqrt{\log N})}$. So the main reason for which a chain can be broken are collisions caused by an adversary during execution of the relay procedure – this occurs with probability $O(1/\sqrt{\log N})$. In this case one chain stops exactly where the next one starts. So in order to connect these chains it suffices that the last station in one chain contacts the last station in the next chain. The time moment for this message can be determined based on the ID of the beginning of the next chain – this value is known to both stations. Well, the adversary knows this moment, too, so the next trick is that during merging phase the stations electing the group leader shift the ID's by a pseudorandom value based again on the shared secret.

The outcome of the group selection is either no leader or a set of stations (a chain) in which every station knows all ID's of active stations in the chain. With high probability the number of active stations in a such a chain is $\Omega(d)$. For our purposes we may modify the procedure so that if some boundary number $c \cdot d$ of active stations is not reached (for some constant c), the group switches off. So:

> if the group is not disabled by another group, then with probability $1 - o(1)$ we get a set of $\Omega(d)$ active stations (a chain) such that each station knows all ID's of the active stations in the group. So we can easily initialize the active stations in the group with consecutive numbers.

We see that the leader election procedure from [5] does slightly more than electing leader. In fact, a set of $\Theta(\log N)$ stations are initialized. However simple extensions of this algorithm to an initialization procedure yields a poor solution. The reason is time complexity $\Theta(\log^3 N)$ of the leader election. So for instance applying the leader election algorithm repeatedly on the set of yet uninitialized stations would provide an algorithm with time complexity $\Omega(N \log^2 N)$, while our target is a linear time.

Joining procedure Assume that we are faced with the following situation: there are K groups of stations; within each group the stations are initialized and there are at least $r+1$ of them; the groups are labeled with numbers 1 through K and each group member is aware of its group label. We show how we can *join* the groups so that the set of stations of all groups except a few stations becomes initialized.

The joining procedure works as follows: group i would like to learn how many stations are in groups 1 through $i-1$. For this purpose the groups communicate like in relay procedure: at phase i a group $\mathcal{B}$ of r stations work in behalf of groups 1 through $i-1$. First, each station from $\mathcal{B}$ sends a message with the number of stations initialized in groups 1 through $i-1$; the messages are sent in different time slots in time windows of size s_1. The first station of group i chooses at random $l = O(\sqrt{\log N})$ moments out of all transmission moments and listens. If it succeeds at least once, then it retransmits the message received in a **single** moment known to members of all groups in a time window

of size s_2. All stations from group i and of $\mathcal{B}$ listen. If the message comes through, then the stations of group i with labels 2 through $r+1$ become the set $\mathcal{B}$. Otherwise, $\mathcal{B}$ remains unchanged and we discard from the initialization all stations from group i with labels higher than $r+1$.

Let $w(i)$ denote the number of stations in groups 1 through i (we take into account the fact of discarding some stations from initialization). Finally, the station of group i with label j gets label $w(i-1)+j$. The number of stations that get discarded from initialization in this procedure and the energy cost depend on the parameters r, s_1, s_2.

We also consider a *modified joining procedure*: we may assume that a group contains either at least $r+1$ stations or no station. The algorithm works almost as before.

3 Adversary Resistant Initialization

Overview The initialization algorithm consists of the following phases:

Phase 1: initialization performed concurrently in $k = \Theta(N/\log^3 N)$ groups of polylogarithmic size;

Phase 2: joining the groups from Phase 1 into a set of $D = \Theta(n/\log^2 N)$ initialized stations;

Phase 3: it consists of 4 subphases, each of them increase the number of initialized stations by a factor of $\Theta(\sqrt{\log N})$ with high probability;
each subphase consists of two parts: during Subphase 3a we form *collection groups* served by already initialized stations (so called *servants*), assign some of the remaining stations to the collection groups and initialize these stations within collection groups; during Subphase 3b we join collection groups;

Phase 4: having already $\Omega(N)$ stations initialized we use them to initialize the remaining stations in a way analogous as in Phase 3 but with modified parameters.

Phase 1 First, every active station chooses independently at random a number in the range $[1,k]$, where $k = \Theta(N/\log^3 N)$. We say that a station that have chosen number i belongs to *election group* i. Then each election group runs independently the leader election algorithm from [5]. However, instead of using time windows of size $O(\log^{3/2} n)$ in each run of the leader election algorithm we interleave k elections. Then each of k election groups initializes at least $c\log N$ stations, for some constant c, or no station, if leader election fails. Each run requires $O(\log^{3/2} N)$ communication steps, so after interleaving them we use together $O(k\cdot\log^{3/2} N)$ steps, which is $o(N)$. Energy cost is $O(\sqrt{\log N})$. Each run succeeds with probability $1-2^{-f}$, where $f = \Omega(\sqrt{\log N})$, hence with high probability at least a constant fraction of groups succeeds. Moreover, the probability of failure of $2\sqrt{\log N}$ consecutive groups is $O(N^{-2})$.

The stations that become initialized within the election groups are called *local leaders*. Each such local leader knows the index of its election group, the number of stations initialized in this group, and its index within the group. For the sake of simplicity we may assume that each successful group elects exactly $c\log N$ local leaders - simply we disregard the stations with higher index.

Phase 2 We apply *modified joining procedure* from Section 2 for the election groups from Phase 1. The parameters used are: $r = c\log N$, $s_1 = \log^2 N$, $s_2 = \log^3 N$. Then computation time is $O(N)$, energy cost is $O(\sqrt{\log N})$. Let D denote the number of stations initialized during Phase 2. Then with probability $p = 1 - 2^{-\Omega(\sqrt{\log N})}$ (derived from Stirling formula) $D = \Theta(k \cdot \log N) = \Theta(N/\log^2 N)$.

Phase 3 The goal of this Phase is to get $\Omega(N)$ initialized stations. For this purpose we repeat four times Subphase 3a and 3b – the number of initialized stations increase by a factor of $\sqrt{\log N}$ in each subphase. We start a subphase by splitting already initialized stations into *collection groups* of size $2h$ for $h = 2^{3\sqrt{\log N}}$. Then they help to assign yet un-initialized stations to collection groups. Each group should get $\Theta(h\sqrt{\log N})$ such stations. Then in Subphase 3b we join collection groups.

Subphase 3a Assume we have $G = \Theta(\frac{N}{\log^t N})$ initialized stations ($t = 2, 1.5, 1, 0.5$). We split them into $H = G/2h$ groups of size $2h$. The stations assigned to group j will be called *servants* of *collection group* j and are indexed 1 through $2h$.

First, each station that is not a servant chooses independently at random a number in the range $[1, N]$. Let us consider a station A that has chosen a number t. If $t > G \cdot \sqrt{\log N}$, then A remains idle during this subphase. Let us consider the opposite case. Let $t = m \cdot H + j$, where $j < H$. Station A listens at step $3t - 2$ of Phase 3. At this moment, one of the servants of group j is sending – the servant with index $2u - 1$, where $u = m \bmod (\log N)$. The message transmitted is the number of stations v already assigned to collection group j. Station A responds with some control message at step $3t - 1$. If there is no collision, then servant $2u$ of group j confirms at step $3t$ that everything went fine. If station A receives this acknowledgment, it regards itself as the station with index $v + 1$ among those stations that have joined collection group j and waits till the end of the subphase. At steps $3t - 2$ and $3t$, the servants from group j with indexes $2u + 1$ and $2u + 2$ listen, too. They learn v and recognize whether a new station has joined collection group t.

Consider what happens, if an adversary disturbs communication. If a collision occurs at step $3t - 1$, then no station joins the collection group at this moment. If the adversary makes a collision at step $3t - 2$ or $3t$, then the servant $2u - 1$ cannot guarantee that the servant $2u + 1$ is aware of v. In this case the servant $2u - 1$ temporarily takes over the next step(s) that would be executed by servant $2u + 1$. In this way the adversary cannot cause any inconsistency. The only problem is that it can force some servant to work in behalf of other servants and terminate execution due to energy limit. However, the adversary cannot influence more than $O(\log N)$ collection groups, so it does not reduce the number of assigned stations significantly.

At the end of Phase 3 in each collection group the current number of stations that have joined this group is broadcast. An appropriate message is sent by the servant that was responsible for the last step in which a station could join this collection group. The remaining servants of this collection group and the stations that have joined this group listen. If an adversary collides this message, then joining this collection group is canceled for all stations. This can be repeated $O(\sqrt{\log N})$ times to decrease adversary chances. We can do it without violating energy cost $O(\sqrt{\log N})$ and linear time. In

this way all stations assigned to the same collection group get the same view of the situation (except the broadcasting stations – these H stations are excluded from the further computation and finally they are assigned some initial set of labels).

Subphase 3a requires time $\Theta(G \cdot \sqrt{\log N})$ which is $O(N)$. A limitation on energy cost of each servant is $O(\sqrt{\log N})$, the other stations have energy cost $O(1)$. With high probability the number of stations that have joined the collection groups is $\Theta(G \cdot \sqrt{\log N})$.

Subphase 3b We perform *joining procedure* of the collection groups. The parameters used are: $r = 2h$, $s_1 = 1$, $s_2 = 2h$. Then computation time is $O(N)$, energy cost is $O(\sqrt{\log N})$. Probability of preventing message passing from group i to group $i+1$ is $O((\log N/h)^{\sqrt{\log N}})$ which is $o(N^{-2})$. Colliding broadcast in group $i+1$ succeeds with probability $O(\log N/h)$. Probability that the adversary manages to do it for $\sqrt{\log N}$ consecutive groups (that would interrupt the joining procedure) is $O((\log N/h)^{\sqrt{\log N}}$ which is $o(N^{-2})$.

Phase 4 After Phase 3 we have a set $\mathcal{L}$ of $l = \Omega(N)$ initialized stations and a set $\mathcal{M}$ of $m = O(N)$ yet not initialized stations. Our goal is to initialize stations from $\mathcal{M}$ using stations from $\mathcal{L}$. For this purpose we follow the algorithm of Subphase 3a with slight changes.

The main problem now is not energy cost of servants, but energy cost of stations from $\mathcal{M}$. Our first goal is to assign each station from $\mathcal{M}$ a unique temporary label in the range $[1, \alpha N]$ for some constant α. We assign also the stations from $\mathcal{L}$ to these labels – so that each station is responsible for $O(1)$ labels.

The assignment of stations from $\mathcal{M}$ is performed in $O(\log\log N)$ rounds (the idea is borrowed from [6]): in the first round labels 1 through βN are chosen independently at random by stations from $\mathcal{M}$. A station A that has chosen j sends at step $2j-1$, at step $2j$ the member of $\mathcal{L}$ assigned to j responds with the message received at step j. If there is no response, then there was a collision at step $2j-1$ or $2j$ and A goes to the next round. Since with high probability the number of stations that go to round 2 is at most δN, we reduce the number of labels used in round 2 to $\gamma\beta N$ (with $\gamma > \delta$) and repeat the same procedure. Following the calculations from [6] one can show that after the last round all but $O(\log N)$ stations from $\mathcal{M}$ have their temporary labels in the range $[1, \alpha N]$. As in Subphase 3a, we use collection groups of size $2h$, but now at most $2h$ stations from $\mathcal{M}$ may join a collection group i, namely the stations with temporary labels in the interval $[(i-1) \cdot 2h - 1, i \cdot 2h]$. After initializing collection groups, we run Subphase 3b (which fails with probability $o(N^{-2})$).

Again, the adversary may disturb communication, but only $O(\log N)$ groups might be influenced, making together $O(h \log N)$ stations uninitialized. To get rid of this problem we simply repeat Phase 4 once more. Now probability that an adversary hits any collection group that contains joining stations is $O(h \log N/N)$ which is $o(2^{-\sqrt{\log N}})$.

Conclusions and future work

We concentrated ourselves on asymptotic behaviour of the initialization procedure. The choice of parameters was adapted to this goal. In the case of networks of realistic size, the design can be fine tuned. The point is that the same tricks may be used there.

The algorithm proposed works also for the case when an adversary can detect a transmission related to the protocol execution as well as collision that has occurred at such a moment.

References

1. Bordim, J. L., Cui, J., Ishii, N., Nakano, K.: Doubly-Logarithmic Energy-Efficient Initialization Protocols for Single-Hop Radio Networks. IEICE Transactions on Fundamentals '2002, 5 , 967-976
2. Jurdziński, T., Kutyłowski, M., Zatopiański, J.: Energy-Efficient Size Approximation for Radio Networks with no Collision Detection. COCOON'2002, LNCS 2387, Springer-Verlag, 279-289
3. Jurdziński, T., Kutyłowski, M., Zatopiański, J.: Weak Communication in Single-Hop Radio Networks – Adjusting Algorithms to Industrial Standards. Concurrency and Computation: Practice & Experience, 15 (2003), 1117-1131. Preliminary version in: Weak Communication in Radio Networks. Euro-Par'2002, LNCS 2400, Springer-Verlag, 965-972
4. Estrin, D.: Sensor Networks Research: in Search of Principles. invited talk at PODC'2002, `www.podc.org/podc2002/estrin.ppt`
5. Kutyłowski, M., Rutkowski, W.: Adversary Immune Leader Election in ad hoc Radio Networks. ESA'2003, LNCS 2832, Springer-Verlag, 397-408
6. Nakano, K., Olariu, S.: Energy Efficient Initialization Protocols in Radio Networks with no Collision Detection. ICPP'2000, IEEE 263-270
7. Nakano, K., Olariu, S.: Randomized Initialization Protocols for Ad-hoc Networks. Transactions on Parallel and Distributed Systems '11 (2000), IEEE 749–759
8. Stojmenović, I. (Ed.): Handbook of Wireless Networks and Mobile Computing. Wiley 2002