On λ-Alert Problem

Marek Klonowski, Dominik Pajak

On λ-Alert Problem

Marek Klonowski
Wrocław University of Technology
Faculty of Fundamental Problems of Technology
Wrocław, Poland
Email: Marek.Klonowski@pwr.wroc.pl

Dominik Pajak
INRIA Bordeaux Sud-Ouest
LaBRI
Talence, France
Email: Dominik.Pajak@labri.fr

***Abstract*—In this paper we introduce and analyse the λ-Alert problem: in a single hop radio network a subset of stations is activated. The aim of the protocol is to decide if the number of activated stations is greater or equal to λ. This problem is similar to the k-Selection problem. It can also be seen as an extension of the standard Alert problem.**

In our paper we consider the λ-Alert problem in various settings. We describe characteristics of oblivious and adaptive deterministic algorithms for the model with and without collision detection. We also show some results for randomized algorithms. In particular, we present a very efficient Las Vegas-type algorithm which is immune to an adversary.

***Keywords*-Alert, k-Selection problem, ad hoc network**

I. Introduction

We consider a network of sensors spread on a large area that can be used to monitor threats, for example fire or earthquake. In the case of detection of symptoms such as smoke or earthquakes the sensor should raise an alarm and notify the whole network of danger. But in practice we may want to raise the alarm only when at least a certain number, say $\lambda > 1$, of sensors detect the threat. Consider, for example, a situation where some small fraction of sensors experiences decreased humidity conditions, which does not necessarily mean that a fire has broken out. In this paper we define and discuss the λ-*Alert* problem in a Wireless Sensor Network. Informally, the problem can be stated as follows : we have a set of n stations. Some of them are *activated*. Our aim is to say if at least λ of them are activated. Special case, when $\lambda = 1$ is called the *Alert problem* and was discussed, in [17].

A. Model description

We consider a Wireless Sensor Network working in ad hoc mode. Let N be the set of all stations. We assume that $|N| = n$. Let K be the set of all *activated* sensors. We denote $|K| = k$. Moreover, stations have unique labels from the set $[n] = \{0, \ldots, n-1\}$. Time is divided into slots. In each slot each station may broadcast or listen. If exactly one station broadcasts in a single slot, all other stations can hear the message and we say that the message is *successfully broadcast*. Sensors are synchronized as they have access to a global clock.

Each sensor is aware if its message was *successfully broadcast* or not. If more than one station is broadcasting none of the broadcast messages can be heard and we say that *Collision* has occurred. Herein we consider two settings - the model *with collision detection* (CD) (when stations may distinguish the slot with broadcasting more than one station from the slot without broadcasting stations) and *without collision detection* (noCD). In the latter case the communication channel can have only two states - BROADCASTING and noBROADCASTING (*Silence* or *Collision*).

B. Problem definition

An instance of the λ-*Alert* problem consists of values of n, λ and set K. Each sensor knows values n, λ and its own state (*activated* or not). Of course, set K and value k are not known to any sensor. At the end of the algorithm each sensor finishes with the state $ALERT$ or $NOALERT$. The result $ALERT$ is correct if and only if $|K| \geq \lambda$. otherwise *NO ALERT* is correct. We say that an algorithm returns a correct answer for a given instance of the λ-Alert problem if all sensors have the correct result after the execution of the protocol. An algorithm is correct if it is correct for every problem instance.

C. Notation

We assume that $0 < \lambda \leq n$. We denote by $r_i(K)$ the result of transmission in the i-th time slot, $r_i \in \{Silence, Signal, Collision\}$. We call an active sensor which has not yet successfully transmitted a *participant*. We will also denote by $r(K) \in \{ALARM, NOALARM\}$ the output of the algorithm.

D. Relations to other problems

To the best of our knowledge, the λ-*Alert* problem was not considered before. However, many similar problems were studied.

Conflict resolution: In the *conflict resolution* problem (known also as k-Selection), all k activated sensors have to transmit. More precisely, we have set K of activated sensors (or processes). For each $s \in K$ there must exist time slot t, that only s broadcasts in t. This problem is similar to our λ-*Alert*, but there is a key difference between these

problems. First of all, in λ-*Alert*, sensors do not have to get exclusive access to the channel. Moreover, in our paper we try to find procedures that are universal with respect to the number of activated stations (i.e. parameter k). Indeed, our procedure has to be correct for any k. There is rich literature devoted to the selection problem. In the case with *collision detection*, the k-Selection problem can be solved in time $O(k \log \frac{n}{k})$ using an oblivious protocol described in [18]. In this algorithm it is assumed that sensors are deactivated after successful transmission. In the strict oblivious model, when sensors must participate even after successful transmission, the best known algorithms are based on superimposed codes introduced in [16] and run in time $O(k^2 \log n)$. An adaptive algorithm with complexity $O(k \log \frac{n}{k})$ was presented by Capetanakis [3]. A lower bound on time complexity of $\Omega(k \log \frac{n}{k})$ for the oblivious model was showed by Clementi, Monti and Silvestri [7]. In the adaptive model Greenberg and Winograd in [13] showed a $\Omega(k \log_k n)$ lower bound. Another important paper related to the k-Selection problem is [19].

Alert: The *Alert* problem is simply λ-*Alert* with $\lambda = 1$. In other words, we only ask if there exists any activated sensor. An energy efficient randomized algorithm solving *Alert* in the model without *collision detection*, working in time $\text{polylog}\, n$, is presented by Klonowski, Kutyłowski and Zatopiański in [17]. Similar problems in different models have been investigated in [6] and [21].

Group testing: This is the problem of identifying at most d infected individuals from set N. A single test consists in choosing any subset $S \subset N$ and testing if any object from S is infected. An oblivious algorithm solving the group testing problem, based on superimposed codes with length $O(d^2 \log n)$, where $n = |N|$, is presented in [1], [2].

Wake-up problem: The alert problem is also related to the *Wake-up* problem [4], [5], [11], [14], wherein a single activated has to contact (wake-up) all other stations.

E. Organization of this paper

In this paper we investigate λ-Alert in various settings. In Section II we present and analyse deterministic algorithms - we show effective algorithms as well as some lower bounds. Section III is devoted to randomized algorithms. We give two solutions. Apart from theoretical analysis we also present some simulations.

A significant number of presented solutions and proofs are based on modifications of previous approaches, as marked, in the text. In particular we very often use methods from [23] and [13].

II. Deterministic algorithms

For each deterministic algorithm $\mathcal{A}$ we can denote by $T_{\mathcal{A}}(K)$ the runtime of the algorithm for given n, λ, K. We also define $T_{\mathcal{A}} = \max_K T_{\mathcal{A}}(K)$. When it is obvious which algorithm is considered, we omit the subscript $\mathcal{A}$ and write just T. Each algorithm defines a sequence of sets $Q_1, Q_2, \ldots, Q_T$. In time slot i, only all active sensors from set Q_i are transmitting. So, the result of transmission depends on the cardinality of set $K \cap Q_i$. If the algorithm is adaptive, then set Q_t can depend on result of transmission in time slots $1, 2, \ldots, t-1$. If the algorithm is oblivious, all sets Q_i are defined before the execution of the algorithm. We assume that inactive stations are not broadcasting in any slot, but only listen.

A. No collision detection

The simplest algorithm for problem is to assign to each sensor a unique time slot. Then each *activated* sensor broadcasts in its time slot and we can count number of the active sensors. This algorithm needs time n. We show that in the model without *collision detection* this algorithm is optimal.

Lemma 2.1: In the model without *collision detection* the runtime of any algorithm in the worst case must satisfy $T > n - \lambda$.

Proof: Let us assume that the time of execution of the procedure is T. For each sensor s we can define a function $p : N \to \{0,1\}^T$, such that p returns a vector for each sensor. We denote by $p(x)_t$, the t-th coordinate of vector $p(x)$. We define p in following way:

- $p(x)_t = 1$, if sensor x broadcasts in time slot t, provided that, it heard *Silence* in all previous slots,
- $p(x)_t = 0$, otherwise.

We can define such function for both oblivious and adaptive algorithms. Then we can construct a set X using following procedure. Initially let $X := N$ (the set of all sensors). We repeat following procedure as long as possible.

- $t \leftarrow \min\left\{u \in \{1, 2, \ldots, T\} : \sum_{s \in X} p(s)_u = 1\right\}$,
- let x be the only sensor broadcasting in time slot t,
- $X \leftarrow X \setminus \{x\}$.

In other words t is the first time slot in which only one sensor x broadcasts. Next we remove x from X. We show that the set X has at least $n - T$ sensors. Indeed, we can only remove sensors from X, thus $h_t = \sum_{s \in X} p(s)_t$ for any t is non-increasing. So we can only once change h_t from 1 to 0. Thus we can remove at most T sensors and for the final set X, it holds that $|X| \geq n - T$.

We can prove using simple induction that if the set of activated stations is X (i.e., $K = X$) then the state of the channel is exactly the same as in the case of the empty set (i.e., $K = \emptyset$). That is, $r(X) = r(\emptyset)$. If the algorithm is correct then $|X| < \lambda$. Finally, $\lambda > |X| \geq n - T$. and $T > n - \lambda$. ∎

Using a similar approach we show following lemma.

Lemma 2.2: For any deterministic algorithm solving the λ-*Alert* problem, we have $T \geq \frac{\lambda}{2}$.

Proof: Let us consider a deterministic algorithm defined by the sequence $Q_1, Q_2, \ldots, Q_T$ if all sensors are activated

(i.e. $K = N$). We can construct a set X satisfying the following conditions:

- $\forall_{i=1}^{T} |X \cap Q_i| \geq \min(2, |Q_i|)$,
- $|X| < 2T$.

It is clear that $r(X) = r(N)$ and all sensors at the end should be in the state $ALERT$. This implies $2T \geq |X| \geq \lambda$. ■

Theorem 2.3: In the model without *collision detection* any deterministic algorithm has time complexity $\Omega(n)$.

Proof: The above theorem follows directly from the above lemmas. That is, $T \geq n - \lambda$ (Lemma 2.1) and $2T \geq \lambda$ (consequence of Lemma 2.2). This implies $3T \geq n$. ■

From the above theorem we see that the simplest algorithm is asymptotically optimal.

B. Collision detection

1) Oblivious algorithms: First of all, let us note that for $\lambda = 2$, we can solve the problem in time 1, independently on the parameter n. If all activated sensors will broadcast in time slot 1, they are able to distinguish the cases $|K| = 0$, $|K| = 1$ and $|K| \geq 2$, and return an alarm only in the last case (i.e., if a *Collision* occurs). For other values of λ, analysis turns to be much more complicated. Fortunately, in many cases the analysis can be reduced to other results and techniques used in other problems - in particular in conflict resolution protocols.

Let us recall some constructions. Let family $\mathcal{C} = \{c_1, c_2, \ldots, c_n\}$ be a set of binary words with fixed length t. The number of vectors n is the size of code. Given k words $c_{i_1}, c_{i_2}, \ldots, c_{i_k}$, we define the sum of vectors $c_{i_1} \vee c_{i_2} \vee \cdots \vee c_{i_k}$ as bitwise Boolean sum. We say that binary vector v covers vector w if for each coordinate with value 1 in w, the corresponding coordinate in v is also 1.

Definition 1: Let r be a positive integer. We say that set of binary words $\mathcal{C} = \{c_1, c_2, \ldots, c_n\}$ is r-superimposed if for any distinct words $c_{i_0}, c_{i_1}, c_{i_2}, \ldots, c_{i_r}$, the word c_{i_0} is not covered by $c_{i_1} \vee c_{i_2} \vee \cdots \vee c_{i_r}$.

Superimposed codes were introduced by Kautz and Singleton in [16]. Now take any λ-superimposed code with length t. Assign one unique codeword to each sensor. Now if sensor is *activated* codeword defines its behaviour (1 – transmit, 0 – listen) during procedure of time t. From the definition, we can see, that if we have at most λ *activated* sensors, then each *activated* sensor will have unique time slot in which it will transmit. Thus each superimposed codes is defining algorithm solving both k-Selection and λ-Alert. Note, that also each algorithm solving k-Selection defines some superimposed code (but this does not work necessarily for λ-*Alert*). Based on these observations we will want to prove lower and upper bound on oblivious λ-*Alert* algorithm. Below we recall an upper bound on the length of codewords t, proven in [8].

Theorem 2.4 (see [8, Theorem 3.1]): There exists an r-superimposed code $\mathcal{C}$ with size n, and codeword length t, such that

$$t = O(r^2 \log n).$$

Using superimposed codes we can solve both k-Selection ([18] [2]), and λ-*Alert*.

Below we show how to modify the k-Selection protocol based on superimposed codes in order to get an algorithm solving the λ-Alert problem.

Lemma 2.5: An algorithm based on λ-superimposed code with length t and the number of codewords at least n solves the λ-*Alert* problem in time t.

Proof: Take any such λ-superimposed code $\mathcal{C}$ and assign one codeword to each sensor. Take any sets $X, Y \subset N$, such that $|X| < \lambda$, and $|Y| \geq \lambda$. Denote by C_X, C_Y sets of codewords assigned to sensors from set X and Y respectively. We want to show that $r(X) \neq r(Y)$. Take any $y \in Y \setminus X$. We know that $\mathcal{C}$ is a superimposed code, so codeword c_y is not covered by $\bigvee_{c \in C_X} c$. So, there exists a time slot t_0 such that y will broadcast (as long as it is activated) and no sensor from X is allowed to broadcast. So

$$r(X) \neq r(Y).$$

We can now construct sets

$$R_1 = \{r(X) : |X| < \lambda\},$$

$$R_2 = \{r(Y) : |Y| \geq \lambda\}.$$

We have just proven that $R_1 \cap R_2 = \emptyset$. Finally, a sensor after the execution of the algorithm can check if $r(K)$ is in R_1 or R_2 and return *NO ALARM* or *ALARM*, respectively. So, such an algorithm solves the λ-*Alert* problem. ■

Theorem 2.6: There exists an oblivious deterministic algorithm solving the λ-*Alert* problem in time $O(\lambda^2 \log n)$.

Proof: We have proven that an algorithm based on superimposed codes solves the problem. We need to show that such codes exists. But constructions of superimposed codes can be found in literature for example in [2]. ■

Lower bound: In this section we prove a lower bound on the time complexity of oblivious deterministic algorithms solving λ-*Alert*. Since in λ-*Alert* it is not necessary, that each *activated* sensor transmits successfully, it is not necessary, that algorithm is based on λ- superimposed code (however it is sufficient). To prove the lower bounds, we need to define different combinatorial structure which will be necessary (but not sufficient). Presented proof has nature similar to lower bounds for k-selection problem. However we need to introduce a different underlying combinatorial structure that we call **r-double superimposed code**.

First, let us define a binary operator $\oplus$ that works on elements from set $\{0, 1, 2\}$ as follows:

$\oplus$	0	1	2
0	0	1	2
1	1	2	2
2	2	2	2

When we add two vectors using operation $\oplus$, we simply use this operator on each position of the summed vectors.

Definition 2: Let r be a positive integer. We say that a set of binary words $\mathcal{C} = \{c_1, c_2, \ldots, c_n\}$ is r**-double superimposed** if for any distinct words $c_{i_0}, c_{i_1}, c_{i_2}, \ldots, c_{i_r}$, $c_{i_0} \oplus c_{i_1} \oplus c_{i_2} \oplus \cdots \oplus c_{i_r} \neq c_{i_1} \oplus c_{i_2} \oplus \cdots \oplus c_{i_r}$.

It easy to see that any oblivious algorithm solving λ-*Alert* is a λ-double superimposed code. Indeed, if we treat 0, 1, and 2 as the states of the channel when none, one or at lest two stations transmit, respectively. We want to prove that a λ-double superimposed code must have length at least $\Omega(\lambda \log n)$. Let us introduce some further definitions.

Definition 3: Family $\mathcal{F} = \{F_1, F_2, \ldots\}$ of subsets of X is r-double-cover-free if

$$(\forall_{F_0, F_1 \ldots, F_r \in \mathcal{F}} \exists_{x \in F_0}) \sum_{i=1}^{r} |F_i \cap \{x\}| \leq 1.$$

First we note that there is a natural correspondence between λ-double-cover-free families $\mathcal{F}$ and λ-double superimposed codes $\mathcal{P}$. For any λ-double-cover-free family we can define a λ-double superimposed code of length $|X|$. For each set $F_i \in \mathcal{F}$ we define c_i as follows. For each $x \in F_i$ we set 1 on x-th position of vector c_i, an all other positions we set 0. Finally, note that the set X represents time slots of some λ-Alert algorithm. Thus, we denote $|X| = t$.

First we will prove the lemma for λ-double-cover-free families with size restriction. Let us denote $g_\lambda(t, k)$ as the maximum cardinality of a λ-double-cover-free family $\mathcal{F}$, where $\forall F \in \mathcal{F}\ |F| = k$.

Lemma 2.7: Let $s = \lceil \frac{2k}{\lambda} \rceil$. If λ is even, then:

$$g_\lambda(t, k) \leq 2 \frac{\binom{t}{s}}{\binom{k-1}{s-1}}.$$

Proof: Below we use ideas of Proposition 2.1 from [8]. We will call a set of cardinality s a *small set*. First we define the following family of *small sets*:

$$\mathcal{N}(F) = \{T \subset F : |T| = s, (\exists F', F'' \neq F; F', F'' \in \mathcal{F}) \\ T \subset F', T \subset F''\}$$

which are *small subsets* of F contained in at least two other sets from $\mathcal{F}$. Note that for any $F \in \mathcal{F}$ and any $T_1, T_2, \ldots, T_{\frac{\lambda}{2}} \in \mathcal{N}(F)$ we have

$$\left| \bigcup_{i=1}^{\frac{\lambda}{2}} T_i \right| < k.$$

Indeed, otherwise F could be doubly covered by λ sets from other sets from $\mathcal{F}$. This, is however, impossible since $\mathcal{F}$ is a double-cover-free family. Up to now we know that $\mathcal{N}(F)$ fulfills the following conditions:

- $\forall_{T \in \mathcal{N}(F)} T \subset F, |T| = s$;
- $\frac{\lambda}{2} s \geq |F| = k$;
- $\left(\forall T_1, T_2, \ldots, T_{\frac{\lambda}{2}} \in \mathcal{N}(F)\right) T_1 \cup T_2 \cup \cdots \cup T_{\frac{\lambda}{2}} \neq F$.

Thus, by Lemma 4.2 from [10], we know that

$$|\mathcal{N}(F)| \leq \binom{k-1}{s}.$$

Now, consider all *small subsets* of some $F \in \mathcal{F}$. We proved, that at most $\binom{k-1}{s}$ are contained in two other sets from $\mathcal{F}$. Thus for any F, we have at least

$$\binom{k}{s} - \binom{k-1}{s} = \binom{k-1}{s-1},$$

different *small subsets* contained in at most one other set from $\mathcal{F}$. Thus

$$\frac{|\mathcal{F}| \binom{k-1}{s-1}}{2} \leq \binom{t}{s}.$$

■

Note that a $(\lambda - 1)$-double-cover-free family is also λ-double-cover-free. Thus, $g_\lambda(t, k) \leq g_{\lambda-1}(t, k)$. For any (no necessarily even) λ, we note the following lemma.

Lemma 2.8: Let $s = \left\lceil \frac{2k}{\lambda - 1} \right\rceil$. For any $\lambda > 1$

$$g_\lambda(t, k) \leq 2 \frac{\binom{t}{s}}{\binom{k-1}{s-1}}.$$

Theorem 2.9: If $g_\lambda(n)$ is the maximum cardinality of a λ-double-cover-free family $\mathcal{F} \subset 2^X$, then

$$g_\lambda(t) \leq 2e^{\frac{2t}{\lambda - 1}}.$$

Proof: One can see that

$$g_\lambda(t) \leq \sum_{k=1}^{t} g_\lambda(t, k) \leq \sum_{k=1}^{t} 2 \frac{\binom{t}{s}}{\binom{k-1}{s-1}}.$$

Indeed, let us consider any λ-double-cover-free family $\mathcal{F}$ and the set of families $\mathcal{F}_1, \mathcal{F}_2, \ldots, \mathcal{F}_t$, where $\mathcal{F}_i = \{F \in \mathcal{F} : |F| = i\}$. Any family $\mathcal{F}_i$ must be λ-double-cover-free, so $|\mathcal{F}| = \sum_{k=1}^{t} |\mathcal{F}_k| \leq \sum_{k=1}^{t} g_\lambda(t, k)$. One can see that

$$\begin{aligned} \binom{k-1}{s-1} &= \frac{k-1}{k-s} \binom{k}{s-1} = \\ \binom{k}{\left\lceil \frac{2k}{\lambda-1} \right\rceil - 1} &\geq \left(\frac{k}{\frac{2k}{\lambda-1}} \right)^{s-1} \geq \frac{(\lambda-1)^{s-1}}{2^{s-1}}, \end{aligned}$$

Thus

$$g_\lambda(t) \leq 4 \sum_{k=1}^{t} \frac{t^s}{s! \left(\frac{\lambda-1}{2}\right)^s} \frac{1}{\lambda - 1} \leq 4 \sum_{k=0}^{\infty} \frac{t^s}{s! \left(\frac{\lambda-1}{2}\right)^s} \frac{1}{\lambda - 1}.$$

Since $s = \left\lceil \frac{2k}{\lambda-1} \right\rceil$, then we have at most $\lceil \frac{\lambda-1}{2} \rceil \leq \lambda - 1$ identical elements in above sum, and

$$g_\lambda(t) \leq 4 \sum_{s=0}^{\infty} \frac{t^s}{s! \left(\frac{\lambda-1}{2}\right)^s} = 4e^{\frac{2t}{\lambda-1}}.$$

■

Now we are ready to prove the main result of this part.

Theorem 2.10: If $\lambda > 2$, and $n \geq 3$, then for all deterministic oblivious algorithms solving the λ-*Alert* problem we have $T = \Omega(\lambda \log n)$.

Proof: Since every oblivious deterministic algorithm solving λ-*Alert* problem corresponds to a λ-double-cover-free family we can use the previously obtained bound. Take any oblivious deterministic algorithm solving λ-*Alert*. As we already discussed, we can assign a set of slots to each sensor representing the slots in which it will transmit in case of being activated. If an algorithm solves λ-*Alert*, this set of slots is a λ-double-cover-free family. Since we have limited the size of a family for given λ and t, we have $n \leq g_\lambda(t)$. From previous lemma we have $n \leq 4e^{\frac{2t}{\lambda-1}}$, and finally

$$t \geq \frac{(\lambda - 1)\log\left(\frac{n}{4}\right)}{2}.$$

■

2) Adaptive algorithms: In this section we show an adaptive algorithm running in time $O(\lambda \log(\frac{n}{\lambda}))$. Then we state a lower bound on the running time of the λ-*Alert* algorithm adapted from [13].

Algorithm: We identify each sensor with its identifier. The algorithm works in rounds. Each round has $\kappa = 3\left\lfloor\frac{\lambda-1}{2}\right\rfloor$ time slots. Initially, each sensor sets local variable $signals := 0$. The variable $collisions$ is reset to 0 at the beginning of each round. A single round works as follows.

- $collisions := 0$
- Divide set N into κ groups
$$N_i = \{x \in N : \kappa | (x - i)\},$$
where $\kappa|(x-i)$ means that κ divides $(x-i)$.
- If sensor fulfills all the following conditions
  - it belongs to set N_i;
  - it is active;
  - it has not broadcasted successfully yet;

  then the sensor broadcasts in the i-th slot of this round and listens in the other slots. Let us denote the result of transmission in the i-th slot by $r(i)$.
- $signals := signals + |\{i : r(i) = Signal\}|$, $collisions := |\{i : r(i) = Collision\}|$.
- If $\lambda \geq signals + 2collisions$ then *ALARM*.
- If $collisions = 0$ and $signals < \lambda$ then *NO ALARM*.
- Else
$$N' = \bigcup_{r(i)=Collision} N_i.$$
- If $|N'| + signals < \lambda$, then *NO ALARM*.
- Assign new identifiers to sensors from set N' (identifiers from set $\{1, 2, \ldots, |N'|\}$).
- $N := N'$.

The algorithm repeats this procedure until a result (*ALARM* or *NO ALARM*) is obtained. We want to show that this algorithm is correct and works in time $O(\lambda \log(\frac{n}{\lambda}))$. First notice that each *participant* belongs to set N'. If an active sensor transmits successfully, it is removed from N, but the number of such sensors is counted by each sensor (i.e., the variable $signals$ is incremented). We want to show that if the algorithm returns *ALARM*, then $|K| \geq \lambda$. Note that

$$N_i \cap N_j = \emptyset, \quad \text{for } i \neq j.$$

If *ALARM* is returned, then since the sets N_i are disjoint, we have at least $2 \cdot collisions$ of *participants*. We also have exactly $signals$ sensors that have made successful transmissions. Thus $|K| \geq \lambda$, and the result is correct. Now, if the result is *NO ALARM*, then there were less than λ $signals$ and no *Collision*, so there are no more *participants*. In such a case, result is also correct. We need to show that the algorithm always finishes its work. We will do this by bounding the time complexity. Assigning new identifiers is simple because sensors are aware which N_i sets are removed from N, and can compute offsets of identifiers. Set N' and new identifiers are computed locally by each sensor. We omit local computations in complexity analysis, because they are fast compared to time of transmitting and receiving messages. Below we show a bound on the running time of the algorithm.

Theorem 2.11: For every input $K \subset N$, and $1 < \lambda < n$, the algorithm terminates in $O(\lambda \log\left(\frac{n}{\lambda}\right))$ time slots.

Proof: The key observation is that if in round i the result is not returned, then

$$|N^{(i+1)}| \leq \frac{|N^{(i)}|}{3} + \left\lfloor\frac{\lambda - 1}{2}\right\rfloor,$$

where $N^{(i)}$ denotes variable $N^{'}$ in round i. Indeed, because the result *ALARM* is not returned, we have at most $\frac{\lambda-1}{2}$ *Collisions* in this round. Since we remove set N_i if *Signal* or *Silence* appears in the i-th slot in round, we have

$$|N_i| \leq \left\lceil\frac{N^{(i)}}{\kappa}\right\rceil \leq \frac{|N^{(i)}|}{\kappa} + 1,$$

$$|N^{(i+1)}| \leq \left\lfloor\frac{\lambda-1}{2}\right\rfloor\left(\frac{|N^{(i)}|}{\kappa} + 1\right) \leq \frac{|N^{(i)}|}{3} + \left\lfloor\frac{\lambda-1}{2}\right\rfloor.$$

We denote $n^{(i)} = |N^{(i)}|$. We have already proven that

$$n^{(t+1)} \leq \frac{n^{(t)}}{3} + \left\lfloor\frac{\lambda-1}{2}\right\rfloor.$$

One can see that sequence $n^{(1)}, n^{(2)}, n^{(3)}, \ldots, n^{(t)}$ is decreasing if $n^{(t)} > \kappa$. Finally,

$$n^{\left(\log_3\left(\frac{n}{\lambda}\right)\right)} \leq \frac{n^{(1)}}{3^{\log_3\left(\frac{n}{\lambda}\right)}} + \sum_{i=0}^{\log_3\left(\frac{n}{\lambda}\right)} \frac{\left\lfloor\frac{\lambda-1}{2}\right\rfloor}{3^i} \leq \frac{n}{\frac{n}{\lambda}} + \frac{\lambda}{2}\frac{3}{2} \leq 2\lambda.$$

This means that if in rounds $1, 2, \ldots, \log_3\left(\frac{n}{\lambda}\right)$ there was no result, then in the next round there are less than 2κ sensors

in set N'. Thus, sets N_i will have 1 or 2 sensor each. In this case the algorithm finishes in the next round. Thus, the algorithm will always terminate. Its runtime is $O(\lambda \log\left(\frac{n}{\lambda}\right))$, because each round lasts $O(\lambda)$ slots. ■

Below we discuss lower bounds on the execution time of adaptive algorithms solving the λ-Alert problem. All theorems can be proved using straightforward modifications of proofs from [13] due to Greenberg and Winograd. The only trick is to find appropriate "worst case" sets K of activated stations.

Theorem 2.12 ([13, Theorem 1]): For any n and λ $(3 \leq \lambda \leq n)$, in the worst case at least $\frac{\lambda}{2} + \log(\frac{n}{\lambda})$ time slots are needed to solve λ-*Alert* problem.

Theorem 2.13 ([13, Theorem 2]): For $8 \leq \lambda < \frac{2}{3}n$, in the worst case, at least $\Omega((\lambda/\log\lambda)(\log\frac{n}{\lambda}))$ time is needed to solve λ-*Alert*

Combining the two previous theorems gives us the following result.

Theorem 2.14: For any $\lambda > 2$, any adaptive deterministic algorithm needs in the worst case $\Omega\left(\lambda \log_\lambda n\right)$ time to solve λ-*Alert*.

The table below summarizes results for λ-Alert problem discussed in this section.

Model	Deterministic lower bound	Deterministic known algorithm
no-CD	$\Omega(n)$	$O(n)$
CD oblivious	$\Omega\left(\lambda \log n\right)$	$O(\lambda^2 \log n)$
CD adaptive	$\Omega\left(\lambda \frac{\log n}{\log \lambda}\right)$	$O\left(\lambda \log\left(\frac{n}{\lambda}\right)\right)$

Table I
TABLE FOR λ-*Alert*

III. RANDOMIZED ALGORITHMS

In this section we switch to randomized, adaptive algorithms. In subsection III-B we present the Election Alert Algorithm (EAA) algorithm that can be regarded as an extension of *Uniform–election* from [23]. Subsection III-C is devoted to a short description of another randomized protocol called Oracle Algorithm. In both discussed algorithms, a labelling of nodes, very important in deterministic algorithms, is not required. Both algorithms are of Las Vegas type. That is, they always return the correct answer, however the time of execution may differ in different executions. Although the first algorithm is better in asymptotic sense, in many cases Oracle Algorithm seem to be more efficient. This intuition is supported by experimental results given in III-D. As before, activated sensors that have not transmitted successfully yet are called *participants*.

Since the time of execution of each protocol depends on many parameters, we recall and set some notation.

Let random variable $T_{n,\lambda,K}$ be the runtime of an algorithm for given n, λ, K.

Definition 4: We say, that a randomized algorithm works in expected time $O(f(n,\lambda))$, if

$$\exists_{c,n_0,\lambda_0} \forall n > n_0 \forall_{\lambda>\lambda_0} \forall_{K\subset N} \mathbf{E}\left(T_{n,\lambda,K}\right) < cf(n,\lambda).$$

Definition 5: A randomized algorithm works in time $O(f(n,\lambda))$ with probability at least p, if

$$\exists_{c,n_0,\lambda_0} \forall n > n_0 \forall_{\lambda>\lambda_0} \forall_{K\subset N} \Pr\left(T_{n,\lambda,K} < cf(n,\lambda)\right) \geq 1-p.$$

A. Common sub-procedures

Below we describe some auxiliary procedures used later.

Test procedure: The first common procedure is called *test*. It takes a single time slot. It works as follows:

Function test(c)

1 Each currently *activated* sensor transmits with probability $\frac{1}{c}$. *Silence*, *Signal* or *Collision* is returned depending on the number of sensors which transmit.

Deactivation: Activated sensor that have made successful transmission (we assume that sensor which is making a successful transmission is aware of this) can deactivate and behave from this moment as an inactive one. They are just notified of the algorithm's result at the end.

Signals counting: Each *activated* sensor counts the number of *Signals*, and returns *ALARM* if the counter reaches λ. When *Signal* appears, the following procedure is executed.

Procedure incrementObserved

1 $signalsObserved \leftarrow signalsObserved + 1$
2 **if** $signalsObserved \geq \lambda$ **then**
3 | *ALARM*
4 **end**

Control slots: Another idea implemented in all our algorithms in this section are *control slots*. The value returned by $test(1)$ can tell us if all *activated* sensors have already broadcast. Note that for $k < \lambda$, a Las Vegas algorithm can end only in a control slot.

Below we also recall a theorem that we use in the analysis.

Theorem 3.1 (see [22, Theorem 4.4]): Let $X_1, \ldots, X_n$ be independent Poisson trials such that $\Pr(X_i = 1) = p_i$. Let $X = \sum_{i=1}^{n} X_i$ and $\mu = \mathbb{E}[X]$. Then the following Chernoff bounds hold:

Procedure controlSlot

```
result ←test (1)
if result ← Silence then
    NO ALARM
else if result ← Signal then
    incrementObserved
    controlSlot
end
```

1) for any $\delta > 0$,

$$\Pr\left(X \geq (1+\delta)\mu\right) \leq \left(\frac{e^{\delta}}{(1+\delta)^{1+\delta}}\right)^{\mu}; \quad (1)$$

2) for $0 < \delta \leq 1$,

$$\Pr\left(X \geq (1+\delta)\mu\right) \leq e^{\frac{-\mu\delta^2}{3}}; \quad (2)$$

3) for $R \geq 6\mu$,

$$\Pr\left(X \geq R\right) \leq 2^{-R}. \quad (3)$$

B. Election Alert Algorithm (EAA)

Below we present Election Alert Algorithm (EAA) based on a leader election protocol called *Uniform-election*, described by Nakano and Olariu in [23]. Our protocol works in the same way as *Uniform-election*, however it does not stop after one *Signal*, but will continue until λ *Signals* appear, or there are no more *participants* left (to detect such a case, we use control slots). The EAA is uniform - i.e. each node executes the same algorithm. Execution of the protocol depends on parameter $f \geq 1$. Election Alert Algorithm works as follows. The pseudocode of EEA is similar to *Uniform-election*, but its analysis is different. In particular, we need to take into account that more than one *Signal* is expected. To solve the λ-*Alert* problem with this procedure, each activated sensor must count the number of *Signals*, and stop the procedure when the counter reaches λ. We also use here control slots described in subsection III-A. Inactive sensors are notified about the result using the procedure *result broadcast* also defined in III-A. Let $f \geq 1$ be arbitrary, and $s = \lceil \log\log(4kf) \rceil$. The first lemma bounds the time complexity of Phases 1 and 2. We did not modify Phases 1 and 2, thus proofs of next two lemmas are exactly the same as in [23].

Lemma 3.2 (Nakano, Olariu, see [23, Lemma 3.1]): With probability exceeding $1 - \frac{1}{4f}$, Phase 1 and Phase 2 combined take at most $2\log\log n + O(\log\log f)$ time slots. The second lemma proves the correctness of u after phase 2. The proof of this lemma can be found in [23].

Lemma 3.3 (Nakano, Olariu, see [23, Lemma 3.2]): With probability exceeding $1 - \frac{1}{2f}$, when Phase 2 terminates, u satisfies the double inequality $\frac{k}{\ln(4(s+1)f)} \leq 2^u \leq 4(s+1)fk$.

Algorithm 2: Election Alert Algorithm

```
Phase 1:
signalsObserved ← 0
i ← 1
repeat
    i ← i + 1
    result ← test (2^(2^i))
    if result = Signal then
        incrementObserved
        controlSlot
    end
until result = Silence;
Phase 2:
l ← 1
u ← 2^i
while l + 1 < u do
    m ← ⌈(l+u)/2⌉
    result ← test (2^m)
    if result = Signal then
        incrementObserved
        controlSlot
    end
    if result = Silence then
        u ← m
    else
        l ← m
    end
end
Phase 3:
while true do
    result ← test (2^u)
    if result = Silence then
        u ← max{u − 1, 1}
    else if result = Collision then
        u ← u + 1
    else if result = Signal then
        incrementObserved
        controlSlot
    end
end
```

Lemma 3.4: Protocol Election Alert Algorithm terminates with probability at least $1 - \frac{1}{f}$, in at most $2\log\log n + o(\log\log n) + O(\log f) + O(\lambda)$ time slots.

Proof: This proof is a modification of the proof of lemma 3.3, from [23]. The main difference is that we expect multiple *Signals*, thus the number of *participants* (which is estimated by 2^u) changes in time. Let us denote $k^{(i)}$ as the number of *participants* in the i-th step of execution. We can also define $v^{(i)}$ as an integer, such that:

$$2^{v^{(i)}-1} < k^{(i)} \leq 2^{v^{(i)}}.$$

Of course, $k^{(0)} = k$, and $v^{(0)} = \lceil \log k \rceil$. We say that $\texttt{test}(2^u)$ performed in Phase 3 *fails to decrease* if $u \geq v^{(i)} + 2$ and the result of *Test* is *Collision*. If the result is *Silence*, we say that it *succeeds to decrease*. When $u \leq v^{(i)} - 2$, and the status of the channel is *Collision*, than this call *fails to increase*, when status is *Silence*, than this call *succeeds to increase* the estimate.

The call $\texttt{test}(2^u)$ is *good* if $v^{(i)} - 2 \leq u \leq v^{(i)} + 2$. Note, that in this case the probability of *Signal* is:

$$\begin{aligned} & \binom{k^{(i)}}{1} \frac{1}{2^u} \left(1 - \frac{1}{2^u}\right)^{k^{(i)}-1} > \frac{k^{(i)}}{2^u} e^{-\frac{k^{(i)}}{2^u}} \\ > & \min\left\{ \frac{2^{u+2}}{2^u} e^{-\frac{2^{u+2}}{2^u}}, \frac{2^{u-3}}{2^u} e^{-\frac{2^{u-3}}{2^u}} \right\} \\ > & \min\left\{ \frac{4}{e^4}, \frac{1}{8e^{\frac{1}{4}}} \right\} = \frac{4}{e^4}. \end{aligned}$$

We note that probability is bounded by a constant independent of u. Thus, if we have at least $\frac{e^4}{4}(\lambda + \ln(4f))$ *good* slots, we can compute the probability of at least λ successes. More precisely, we have at most $\min\{k, \lambda\}$ Poisson trials. Each has expected value at most $\frac{e^4}{4}$. Let X denote the sum of all successes. Using Chernoff bound for $R = \frac{3e^4}{2}(\lambda + \ln(4f))$ we get:

$$\Pr(X > R) \leq 2^{-R} < 2^{-\min\{\lambda,k\}} \frac{1}{4f} \leq \frac{1}{4f}.$$

Note that $R = 6\frac{e^4}{4}(\min\{k, \lambda\} + \ln(4f)) > \min\{k, \lambda\} + \log(4f)$.

Finally, if we have during the whole execution at least $\frac{3e^4}{2}(\lambda + \ln(4f))$ good slots, than we are certain that λ-*Alert* problem will be solved with probability at least $1 - \frac{1}{4f}$. Now we want to show that *good* calls occur quite frequently in Phase 3. First we will bound the probability that a call $\texttt{Test}(2^u)$ fails. Let Z denote the number of sensors transmitting in a particular time slot. Clearly, $\mathbf{E}[Z] = \frac{k^{(i)}}{2^u}$. So, if $u \geq v^{(i)} + 2$, than the call $\texttt{Test}(2^u)$ fails fo decrease with probability at most:

$$\begin{aligned} \Pr[Z > 1] &= \Pr\left[Z > \frac{2^u}{k^{(i)}} \mathbf{E}[Z]\right] \\ &< \Pr\left[Z > \frac{2^u}{2^{v^{(i)}}} \mathbf{E}[Z]\right] \quad (\text{from } k^{(i)} \leq 2^{v^{(i)}}) \\ &< \Pr\left[Z > 4\mathbf{E}[Z]\right] \quad (\text{from } u \geq v^{(i)} + 2) \\ &< \frac{1}{4.} \quad (\text{by Markov's inequality}) \end{aligned}$$

On the other hand, if $u \leq v^{(i)} - 2$, then the probability that $\texttt{Test}(2^u)$ fails to increase is at most:

$$\begin{aligned} \Pr[Z = 0] &= \left(1 - \frac{1}{2^u}\right)^{k^{(i)}} \\ &< e^{-\frac{k^{(i)}}{2^u}} \\ &< e^{-\frac{2^{v^{(i)}-1}}{2^u}} \quad (\text{from } 2^{v^{(i)}-1} < k^{(i)}) \\ &< e^{-2} \quad (\text{from } u \leq v^{(i)} - 2) \\ &< \frac{1}{4}. \end{aligned}$$

Finally, the call $\texttt{Test}(2^u)$ fails with probability at most $\frac{1}{4}$. Now, suppose that we execute $\texttt{test}(2^u)$ $\frac{8}{3}e^4(\ln(4f) + \log\log\log k + \lambda)$ times in Phase 3. Let N_s,N_f,N_g be the number of times $\texttt{Test}(2^u)$, succeeds, fails, and is good among these $\frac{8}{3}e^4(\ln(4f) + \log\log\log k + \lambda)$ calls, respectively. Clearly:

$$N_s + N_f + N_g = \frac{8}{3}e^4(\ln(4f) + \log\log\log k + \lambda).$$

If at the end of Phase 2, u satisfies the double inequality of Lemma 3.3, we have:

$$\begin{aligned} u &\geq \log\left(\frac{k}{4(s+1)f}\right) \\ &> \log k - \log(s+1) - \log\log f - 2 \\ &> v^{(0)} - \log\log f - \log\log\log k - \log\log\log f - 4 \end{aligned}$$

and similarly

$$\begin{aligned} u &\leq \log(k(\ln(4(s+1)f))) \\ &< \log k + \log\ln(s+1) + \log\ln f + 2 \\ &< v^{(0)} + \log\log f + \log\log\log\log k + \log\log\log\log f + 3. \end{aligned}$$

Thus we have,

$$\left|u - v^{(0)}\right| < 2\log\log f + \log\log\log k + 4.$$

If equation 4 holds at the end of Phase 2, we have

$$N_s < N_f + 2\log\log f + \log\log\log k + 2 + \log\lambda. \quad (4)$$

Since a particular call $\texttt{Test}(2^u)$ falls with probability at most $\frac{1}{4}$, we have:

$$\mathbb{E}[N_f] \leq \frac{2e^4}{3}(\ln(4f) + \log\log\log k + \lambda).$$

Now using Chernoff bound we can limit the probability that N_f will exceed $e^4(\ln(4f) + \log\log\log n + \lambda)$.

$$\begin{aligned} &\Pr[N_f > e^4(\ln(4f) + \log\log\log k + \lambda)] < \\ &< \Pr\left[N_f > \left(1 + \frac{1}{2}\right)\mathbb{E}[N_f]\right] \\ &< e^{-\frac{1}{2^2 3}\mathbb{E}[N_f]} \\ &< e^{\frac{e^4}{24}(\ln(4f) + \log\log\log k + \lambda)} \\ &< \frac{1}{4f}. \end{aligned}$$

Suppose that $N_f < e^4(\ln(4f) + \log\log\log k + \lambda)$ is satisfied. We have already proven that this happens with probability at least $1 - \frac{1}{4f}$. Then, we have:

$$\begin{aligned} N_g &= \frac{8}{3}e^4(\ln(4f) + \log\log\log n + \lambda) - (N_s + N_f) \\ &\geq \frac{8}{3}e^4(\ln(4f) + \log\log\log n + \lambda) \\ &\quad -2N_f - (2\log\log f + \log\log\log k + 2 + \log\lambda) \\ &> \frac{e^4}{4}(\ln(4f) + \lambda). \end{aligned}$$

So, with probability at least $1 - \frac{1}{4f}$, among $\frac{8}{3}e^4(\ln(4f) + \log\log\log k + \lambda)$ calls $\texttt{Test}(2^u)$, there are at least $\frac{e^4}{4}(\ln(4f) + \lambda)$ good ones. But then with probability at least $\frac{1}{4f}$ we succeed in finishing algorithm. So, if at the end of Phase 2, u satisfies the double inequality in Lemma 3.3, then with probability $1 - \frac{1}{2f}$, Phase 3 terminates in at most $\frac{8}{3}e^4(\ln(4f) + \log\log\log k + \lambda)$ time slots. By Lemma 3.2, with probability at least $\frac{1}{2f}$, the combined time of Phases 1 and 2 is at most $2\log\log k + O(\log\log f)$. Finally, with probability at least

$$\left(1 - \frac{1}{2f}\right)\left(1 - \frac{1}{4f}\right)^2 > 1 - \frac{1}{f},$$

the algorithm terminates in time $2\log\log n + o(\log\log n) + O(\log f) + O(\lambda)$. ∎

Now we can formulate the final results of this subsection.

Theorem 3.5: There exists a uniform randomized algorithm solving λ-*Alert* without knowledge of n, with probability exceeding $1 - \frac{1}{f}$, in time $O(\log\log n + \lambda + \log f)$.

Proof: The time complexity follows directly from the previous lemma. We need to prove its correctness. But since we will terminate only when at least λ *Signals* appear, or there are no *participants*, we cannot return the wrong result. We can see in the pseudocode that $u > 0$ during whole algorithm. Thus, if the number of *participants* is positive, then in each time slot the probability of successful transmission is positive. If there are no *participants*, we will end in the next control slot. Thus, Election Alert Algorithm always solves λ-*Alert*. ∎

C. Oracle Algorithm

In this section we introduce another approach to the construction of λ-*Alert* that we call *Oracle Algorithm*. The idea is as follows. Each round consists of two phases. In the first phase we run a very precise size approximation protocol ("oracle") that gives us an approximation with constant factor of accuracy w.h.p. The second phase lasts until a fixed number of *signals* appears. Each signal is followed by a control slot. Consecutive rounds are repeated until the total number of signals is equal to λ or there are no other participants. The last condition can be detected in the control slot procedure.

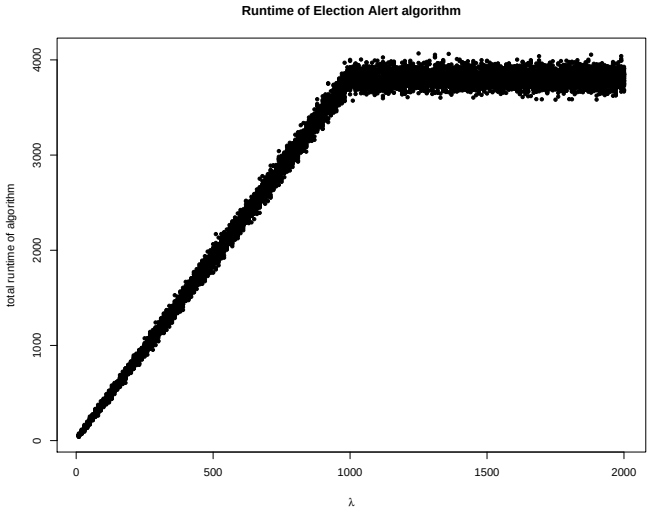
We have proposed an implementation of the above idea using HyperLogLog algorithm ([9]). Of course, it had to be modified to work in an ad hoc single-hop network. HyperLogLog works in time $O(\log\log n)$ and returns a very precise approximation of the number of activated sensors. We can prove that having an oracle returning an approximation of the number of *participants* allows us to construct a λ-*Alert* algorithm working in time $O(\lambda)$. Then we applied HyperLogLog as an oracle and we obtained an algorithm working in time $O(\log n \log\log n \log\lambda + \lambda)$ with probability

- at least $1 - \left(2^{-\lambda} + \frac{1}{n^3}\right)$, if n is large enough, but $n < M$,
- at least $1 - \left(2^{-\lambda} + \frac{1}{M^3}\right)$, if n is large enough, and $n \geq M$,

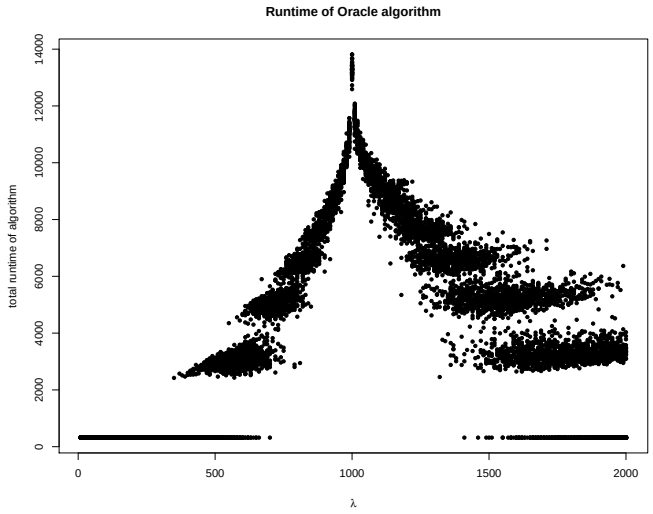
where $M = 2^{10^7}$. We skip the analysis and details of description due to space limitations. This algorithm seems to be much worse than for example EAA described before (exponential difference !). However, in practice it may be very useful and extremely fast in some particular cases as shown in the next subsection.

D. Experiments

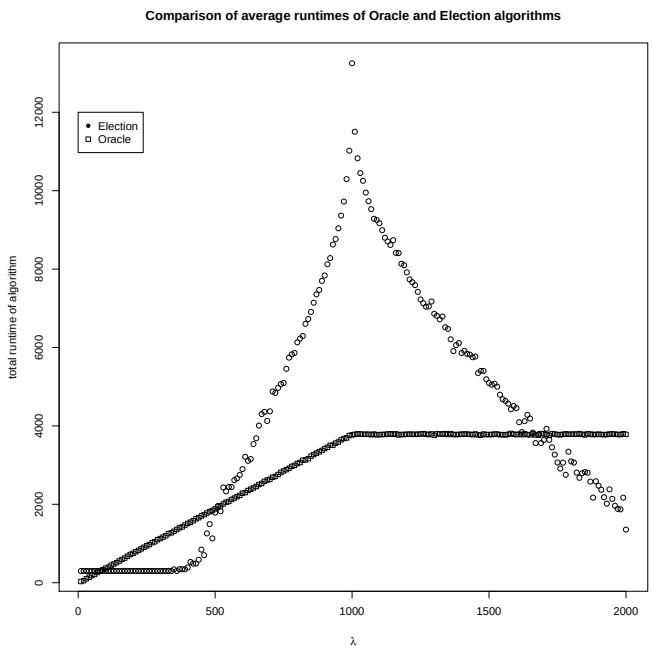
We have made tests comparing Election and Oracle Algorithm. Tests were done for $n = 10000, k = 1000, m = 16$. We have done 50 tests for each value of λ from 10 to 1000, divisible by 10. The first plot shows the execution time of each test using EAA algorithm.



Runtime of Election algorithm is approximately equal to $3\min\{k, \lambda\}$.



The next chart shows the runtime of the Oracle Algorithm. We can see that the algorithm worked for the longest time in the case when k was close to λ. This happened because in this case the number of executions of the HyperLogLog subroutine was the largest.



Finally, we compare both algorithms. We can see that the Oracle Algorithm works faster in the case when $k \gg \lambda$, or $k \ll \lambda$. But the variance of its runtime is bigger when compared to the Election Alert Algorithm. On the other hand, EAA works much faster in the case when $k \approx \lambda$.

IV. Conclusions

In this paper we discussed the λ-Alert algorithm. Although we have shown results for many of the most natural settings, several important questions are left unanswered. In particular, it is not clear how to design energy-efficient protocols that solve the λ-Alert problem. In analysis, we always discussed the worst–case scenario. It seems that the proposed algorithms are very far from optimal if we have some knowledge about K (i.e., the set of activated nodes), for example, if we know distribution of K. Such a model can be very natural, in particular, if nodes are activated independently with the same (possibly unknown) probability.

Acknowledgement

Partially supported by Polish Ministry of Science and Higher Education, grant NN206 2573 35, by the ANR projects ALADDIN and DISPLEXITY, and by the INRIA project CEPAGE.

References

[1] Annalisa De Bonis, Leszek Gasieniec, and Ugo Vaccaro. Optimal two-stage algorithms for group testing problems. *SIAM J. Comput.*, 34(5):1253–1270, 2005.

[2] Annalisa De Bonis and Ugo Vaccaro. Efficient constructions of generalized superimposed codes with applications to group testing and conflict resolution in multiple access channels. In *ESA '02: Proceedings of the 10th Annual European Symposium on Algorithms*, pages 335–347, London, UK, 2002. Springer-Verlag.

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

[4] Bogdan S. Chlebus and Dariusz R. Kowalski. A better wake-up in radio networks. In Soma Chaudhuri and Shay Kutten, editors, *PODC*, pages 266–274. ACM, 2004.

[5] Marek Chrobak, Leszek Gasieniec, and Dariusz R. Kowalski. The wake-up problem in multihop radio networks. *SIAM J. Comput.*, 36(5):1453–1471, 2007.

[6] Jacek Cichoń, Rafal Kapelko, Jakub Lemiesz, and Marcin Zawada. On alarm protocol in wireless sensor networks. In Ioanis Nikolaidis and Kui Wu, editors, *ADHOC-NOW*, volume 6288 of *Lecture Notes in Computer Science*, pages 43–52. Springer, 2010.

[7] Andrea E. F. Clementi, Angelo Monti, and Riccardo Silvestri. Selective families, superimposed codes, and broadcasting on unknown radio networks. In *Proceedings of the twelfth annual ACM-SIAM symposium on Discrete algorithms*, SODA '01, pages 709–718, Philadelphia, PA, USA, 2001. Society for Industrial and Applied Mathematics.

[8] P. Erdös, P. Frankl, and Z. Füredi. Families of finite sets in which no set is covered by the union of r others. *Israel Journal of Mathematics*, 51:79–89, 1985. 10.1007/BF02772959.

[9] Philippe Flajolet, Éric Fusy, Olivier Gandouet, and et al. Hyperloglog: The analysis of a near-optimal cardinality estimation algorithm. In *IN AOFA 07: PROCEEDINGS OF THE 2007 INTERNATIONAL CONFERENCE ON ANALYSIS OF ALGORITHMS*, 2007.

[10] P. Frankl. On sperner families satisfying an additional condition. *Journal of Combinatorial Theory, Series A*, 20(1):1 – 11, 1976.

[11] Leszek Gasieniec, Andrzej Pelc, and David Peleg. The wakeup problem in synchronous broadcast systems. *SIAM J. Discrete Math.*, 14(2):207–222, 2001.

[12] Zbigniew Golebiewski, Marek Klonowski, Michal Koza, and Miroslaw Kutylowski. Towards fair leader election in wireless networks. In Pedro M. Ruiz and Jose Joaquin Garcia-Luna-Aceves, editors, *ADHOC-NOW*, volume 5793 of *Lecture Notes in Computer Science*, pages 166–179. Springer, 2009.

[13] Albert G. Greenberg and Schmuel Winograd. A lower bound on the time needed in the worst case to resolve conflicts deterministically in multiple access channels. *J. ACM*, 32(3):589–596, 1985.

[14] Tomasz Jurdzinski and Grzegorz Stachowiak. Probabilistic algorithms for the wake-up problem in single-hop radio networks. *Theory Comput. Syst.*, 38(3):347–367, 2005.

[15] Jedrzej Kabarowski, Miroslaw Kutylowski, and Wojciech Rutkowski. Adversary immune size approximation of single-hop radio networks. In Jin yi Cai, S. Barry Cooper, and Angsheng Li, editors, *TAMC*, volume 3959 of *Lecture Notes in Computer Science*, pages 148–158. Springer, 2006.

[16] W. Kautz and R. Singleton. Nonrandom binary superimposed codes. *Information Theory, IEEE Transactions on*, 10(4):363 – 377, oct. 1964.

[17] Marek Klonowski, Miroslaw Kutyłowski, and Jan Zatopiański. Energy efficient alert in single-hop networks of extremely weak devices. In Shlomi Dolev, editor, *Algorithmic Aspects of Wireless Sensor Networks*, volume 5804 of *Lecture Notes in Computer Science*, pages 139–150. Springer Berlin / Heidelberg, 2009.

[18] J. Komlos and A. Greenberg. An asymptotically fast nonadaptive algorithm for conflict resolution in multiple-access channels. *Information Theory, IEEE Transactions on*, 31(2):302 – 306, March 1985.

[19] Dariusz R. Kowalski. On selection problem in radio networks. In *Proceedings of the twenty-fourth annual ACM symposium on Principles of distributed computing*, PODC '05, pages 158–166, New York, NY, USA, 2005. ACM.

[20] Miroslaw Kutylowski and Wojciech Rutkowski. Secure initialization in single-hop radio networks. In Claude Castelluccia, Hannes Hartenstein, Christof Paar, and Dirk Westhoff, editors, *ESAS*, volume 3313 of *Lecture Notes in Computer Science*, pages 31–41. Springer, 2004.

[21] Bernard Mans, Stefan Schmid, and Roger Wattenhofer. Distributed disaster disclosure. In Joachim Gudmundsson, editor, *SWAT*, volume 5124 of *Lecture Notes in Computer Science*, pages 246–257. Springer, 2008.

[22] Michael Mitzenmacher and Eli Upfal. *Probability and Computing: Randomized Algorithms and Probabilistic Analysis*. Cambridge University Press, New York, NY, USA, 2005.

[23] Koji Nikano and Stephan Olariu. Uniform leader election protocols for radio networks. *IEEE Trans. Parallel Distrib. Syst.*, 13:516–526, May 2002.