Probabilistic Algorithms for the Wakeup Problem in Single-Hop Radio Networks ⋆

Tomasz Jurdziński†∗ and Grzegorz Stachowiak†

†Institute of Computer Science, Wrocław University
∗ Institute of Computer Science, Technical University of Chemnitz

Abstract. We consider the problem of waking up n processors in a completely broadcast system. We analyze this problem in both globally and locally synchronous models, with or without n being known to processors and with or without labeling of processors. The main question we answer is: how fast we can wake all the processors up with probability $1-\varepsilon$ in each of these eight models. In [11] a logarithmic waking algorithm for the strongest set of assumptions is described, while for weaker models only linear and quadratic algorithms were obtained. We prove that in the weakest model (local synchronization, no knowledge of n or labeling) the best waking time is $O(n/\log n)$. We also show logarithmic or polylogarithmic waking algorithms for all stronger models, which in some cases gives an exponential improvement over previous results.

1 Introduction

We concentrate on the effects of synchronization level in broadcast systems such as the Ethernet or radio networks (RN). The system is assumed to be synchronous, i.e. processors send messages in rounds. It is assumed that the processors succeed in hearing a message in a given round if and only if exactly one processor sends a message in that round (the situations in which more than one processor or none of the processors send a message are indistinguishable). Hence the communication model is equivalent to the radio model [1, 3, 4, 7, 15, 17], in a complete graph. This case in which every processor is directly accessible by all the other ones is called a *single-hop* radio network (general *multi-hop* networks are defined by graphs, where stations denote nodes and a node (u,v) denotes that the station v is reachable from the station u). More precisely, the model considered here is called single-hop radio networks without collision detection (no-CD RN), as opposite to the model in which processors can distinguish between the situations when more than one processor or none of the processors sends a message (CD RN). Because of diversity of technical capabilities of existing technologies, both models (noCD RN and CD RN) are investigated in the literature. However, noCD RN are more general, all positive results obtained for this model are also applicable in CD RN (see e.g. [17, 7, 2, 3]).

A central problem in such systems is to establish the pattern of access to the shared communication media that allows messages to go through with a small delay, i.e., avoid or efficiently resolve message collisions. Another important problem considered is to design algorithms working without complete information about the network (for example topology, size) and/or an information which makes possible to distinguish its elements (e.g. different labels) [7, 17, 10, 3, 6]. This direction is motivated by the mobility of radio networks, changes of the topology and the set of participants.

As shown in [2], algorithms designed for single-hop RNs can be efficiently emulated on multi-hop RNs. Thus, algorithmic results concerning single-hop RN have also some significance for the multi-hop model.

⋆ The first author was supported by DFG grant GO 493/1-1, and the second by KBN grant 8T11C 04419.

2 Problem statement

In this paper, we consider the fundamental problem of waking up all of n processors (or „radio stations") in a completely broadcast system or a single-hop radio network [11] (see [11] for motivations). We distinguish the cases when processors are labeled by the numbers $1,\ldots,n$ or unlabeled. Some processors wake up spontaneously, in different rounds, while others have to be woken up. Only awake processors can send messages. All sleeping processors wake up non-spontaneously upon hearing a message. This happens in the first round when exactly one processor sends a message. Time complexity of an algorithm in a radio network is the number of rounds executed. We consider here the worst case time complexity of the wakeup problem, measured by the number of rounds elapsing from the time the first processor wakes up (spontaneously) to the time all processors are woken up (i.e. up to the first round in which exactly one processor sends a message). The notion „worst case complexity" means here that an adversary controls which processors and when wake up spontaneously, but does not know in advance the values of random choices made by processors.

Following [11], we consider two modes of synchronization. In *globally synchronous* systems, all processors have access to a global clock showing the current round number. In *locally synchronous* systems, all clocks tick at the same rate, one tick per round. However, no global counter is available, the local clock of each processor starts counting rounds when it wakes up. In both models, we distinguish the following assumptions concerning the knowledge of processors (see e.g. [17, 11, 19] for motivations):

- the size of the system, n, is known or unknown to processors,
- processors are labeled by different numbers from the set $\{1,\ldots,n\}$ or they are unlabeled
- the acceptable probability of error for randomized algorithms, ε, is given explicitly as a constant parameter, or only as a function of n.

The first two distinctions were considered in many aspects in research on algorithms for radio networks (e.g. [17, 11, 19]). Let us explain the motivation for the last distinction (observe, that it makes sense only when n is unknown to processors). In designing Monte Carlo algorithms, the consideration of time complexity is often made with the requirement that the error probability should go to zero (polynomially or exponentially) when n is growing. It seems to be reasonable to take such assumptions into account also in the case of distributed algorithms (note that the crash of „large" distributed environment affects the work of „large" number of its „clients"). So, we also consider the case of ε being given as a function of n, which in case of „n unknown" is an additional difficulty for an algorithm's designer.

In [6], authors pointed out that algorithms for RN should be as simple as possible (because „processors" are very often small hand-held devices with limited resources). Therefore they defined a notion of *uniform* algorithms in the model of radio networks. A randomized algorithm for a model with a global clock is called *uniform* if in every round all awake processors send a message with the same probability (independently on the history of communication). Our goal is also to construct efficient uniform algorithms for the wakeup problem.

Efficient solutions of the wake up problem may be also used as a tool improving algorithms for more sophisticated communication tasks in *unknown* multi-hop radio networks (i.e. where none of processors knows a graph that defines reachability in the network). For example, if the stations do not know the graph of the multi-hop network then the arrival of the message to some particular station may be seen as a time needed for the wake-up problem executed on the set of all its in-neighbors, starting from the round when at least one in-neighbor of this station receives the message.

Previous results			
n known	Labels	Global clock	Local clock
Yes	Yes	$O(\log n \log(1/\varepsilon))$	$O(n \log(1/\varepsilon))$
Yes	No	$O(\log n \log(1/\varepsilon))$	$O(n \log(1/\varepsilon))$
No	Yes	–	$O(n^2 \log(1/\varepsilon))$
No	No	not considered	$O(n^2 \log(1/\varepsilon))$

3 Previous and related work

The wakeup problem has been studied in other contexts of distributed computations [8, 9, 16]. Collision detection and resolution, and access management algorithms for the model considered here were studied mainly assuming some probability distribution on the arrival rate of messages at the different processors, cf. [13, 12]. Unlike in these papers we consider worst-case analysis of the noCD RN model, introduced in [11]. Both randomized and deterministic algorithms for wake-up problems were considered. All deterministic algorithms require labeling of processors. It was proved ([11]) that deterministic algorithms for the wake-up problem require the time $\Omega(n)$. For globally synchronized model, the upper bound matching the lower bound was obtained. For local synchronization, it was shown that there exists a deterministic algorithm achieving time $O(n \log^2 n)$. A constructive version of this result (with slightly worse time performance) was presented by Indyk [14]. Results (from [11]) concerning probabilistic algorithms are summarized and compared with our work in the next section.

Recently many other problems in *multi-hop* and *single-hop* radio networks (as broadcasting, gossiping, initialization i.e. labeling, leader election) were considered [7, 6, 3, 1, 17] (one can find more pointers to previous work on this topic in [5]). In most cases, this research concentrated on the model with global synchronization. As pointed out in [5] exploration of differences in complexity of problems in globally and locally synchronized networks is an interesting and promising direction of research (see also [18]).

4 Our results

In this paper, we concentrate on probabilistic algorithms. Previous authors consider the model with an allowed error probability ε given as a constant parameter. They obtained an algorithm waking up all processors in time $O(\log n \log(1/\varepsilon))$ with probability $1-\varepsilon$ in this model, but only in case of global synchronization, and n known for all the processors. When a global clock or the knowledge of n is missing, their algorithms require at least linear time. We present algorithms waking up all processors in time $O(\log n \log(1/\varepsilon))$ with probability $1-\varepsilon$ even for the weakest assumptions considered in [11] (i.e. with local synchronization, unknown n, but labeled processors). These algorithms show that also in this case randomization gives almost exponential improvement in time complexity in comparison to determinism. Moreover, in the model with global synchronization, unknown n and no processor labeling we obtain a polylogarithmic algorithm. This result establishes an almost exponential gap between global and local synchronization, because a lower bound $\Omega(n/\log n)$ holds in the weakest model (with local synchronization, without labels, and with unknown n). Further, the lower bound $\Omega(n/\log n)$ compared with our upper bounds show that in locally synchronous environment the knowledge of the size of the network (or at least its approximation given by labels) is crucial for the complexity of the problem. In the tables presented above we make more detailed comparison with previous results. It is assumed here that ε is given explicitly as a parameter.

Our results			
n known	Labels	Global clock	Local clock
Yes	Yes	$O(\log n \log(1/\varepsilon))$ [uniform]	$O(\log n \log(1/\varepsilon))$
Yes	No	$O(\log n \log(1/\varepsilon))$ [uniform]	$O(\log n \log(1/\varepsilon))$
No	Yes	$O(\log n \log(1/\varepsilon))$	$O(\log n \log(1/\varepsilon))$
No	No	$O(\log^2 n (\log\log n)^3 \log(1/\varepsilon(n)))$ [uniform]	$O\left(\frac{n\log(1/\varepsilon))}{\log n}\right)^*$

$\star$ – matches the lower bound

Algorithm's name	**Minimal requirements**				Time
	n known	Global	Labels	ε explicit	
Repeated Prob. Decrease	Yes	Yes	No	No	$O(\log n \log(1/\varepsilon))$
Probability Increase	Yes	No	No	No	$O(\log n \log(1/\varepsilon))$
Increase from Square	No	No	Yes	Yes	$O(\log n \log(1/\varepsilon))$
Use Factorial Representation	No	Yes	No	No	$O(f(n,\varepsilon))$
Decrease From Half	No	No	No	Yes	$O(n\log(1/\varepsilon)/\log n)$
Decrease Slowly	No	No	No	No	$O(n\log(1/\varepsilon))$

where $f(n,\varepsilon) = \log^2 n(\log\log n)^3 \log(1/\varepsilon)$

We also show a lower bound $\Omega\left(\frac{\log n \log(1/\varepsilon)}{\log\log n + \log\log(1/\varepsilon)}\right)$ for uniform algorithms working with known n and a global clock. This bound almost matches our upper bound for these assumptions: $O(\log n \log(1/\varepsilon))$.

Further on we consider the case when ε is not given as a constant, but as a function of n only. Note that this assumption makes difference only when n is unknown for processors. If there is a global clock, mentioned earlier polylogarithmic algorithm for unlabeled networks can work with unknown ε. When the global clock and the value of n are not available, we can wake up all processors with probability $1-\varepsilon$ in time $O(n\log(1/\varepsilon))$, even without labels (but without labels the function ε should be polynomially decreasing). Thus, for the weakest scenario (i.e. without labels and without knowledge of ε) our solution significantly improves the bound $O(n^2 \log(1/\varepsilon)$ from [11] (where constant ε given explicitly was needed).

Our upper bounds are obtained by a sequence of algorithms, presented in the order of decreasing amount of knowledge and synchronization needed. The above table (see page 3) summarizes time bounds and minimal requirements of these algorithms.

Aside from the fact that our results establish exponential gaps between global and local synchronization, known and unknown (approximation of) n as well as determinism and randomization, they may be helpful in construction of efficient probabilistic algorithms for more general problems on multi-hop networks without global synchronization. This is because of large time efficiency of presented solutions for the wake-up problem, which maybe used as a tool in more general settings. From a technical point of view, maybe the most interesting result was obtained in Section 9 where some interesting properties of a so called *factorial representation* of natural numbers were showed and used for the construction of a very simple but efficient algorithm. The additional advantage of this algorithm is that it has also small expected waking time (opposite to some other algorithms presented here).

In the next section we present some basic mathematical facts and a very useful Sums Lemma. Sections 6–10 are devoted to our randomized algorithms and corresponding lower bounds. We present them in the order of decreasing amount of resources needed. In some cases, our algorithms need an explicit knowledge of allowed error probability, ε. Then, we present also other solutions that work if ε is given as a function of n only.

5 Basic Lemmata

We say that a round of computation is *successful* if and only if exactly one processor is broadcasting in this round. Let *success probability* in a round be a probability that the round is successful. The *broadcast probability* of a processor i in its round r is the probability that i broadcasts a message in the round r. The *broadcast sum* of a round is a sum of broadcast probabilities (in this round) of all processors that are awake in that round.

Let us recall some basic mathematical facts:

Lemma 5.1. *1.* $(1-p)^{1/p} \geq 1/4$ *for every* $p, 0 < p \leq 1/2$.
2. $(1-p)^{1/p} \leq 1/e$ *for every* $p, 0 < p \leq 1$.
3. $\log n! = \Theta(n \log n)$.
4. $\sum_{i=1}^{\infty} 1/i^2 = \pi^2/6$.
5. $\frac{\log n}{2} \leq \sum_{i=2}^{n} \frac{1}{i} \leq \log n$ *and* $\frac{p \log n}{4} \leq p \sum_{i=1}^{n} \frac{1}{2p+i} \leq p \log(2n)$ *for every* $n > 16$ *that is a natural power of two and* $p, 1 \leq p < n^{1/4}$.

Lemma 5.2 (Sums Lemma). *Assume that there are k awake processors in a round, with broadcast probabilities* $p_1, \ldots, p_k$ *such that* $p_i \leq 1/2$ *for* $i = 1, \ldots, k$. *Then, the probability that the round is successful is not smaller than* $(\sum_{i=1}^{k} p_i) \left(\frac{1}{4}\right)^{\sum_{i=1}^{k} p_i}$.

Proof. The probability that exactly one processor is broadcasting is $\sum_{i=1}^{k}(p_i \prod_{j=1; j \neq i}^{k}(1 - p_j)) \geq \sum_{i=1}^{k}(p_i \prod_{j=1}^{k}(1-p_j)) = (\sum_{i=1}^{k} p_i)(\prod_{j=1}^{k}(1-p_j))$. Observe that, by Fact 5.1, $(1 - p_j) = \left((1-p_j)^{1/p_j}\right)^{p_j} \geq (1/4)^{p_j}$. Thus, $\prod_{j=1}^{k}(1-p_j) = \prod_{j=1}^{k}((1-p_j)^{1/p_j})^{p_j} \geq \prod_{j=1}^{k}(1/4)^{p_j} = (1/4)^{\sum_{j=1}^{k} p_j}$. So, the success probability is not smaller than $(\sum_{i=1}^{k} p_i) \cdot (1/4)^{\sum_{j=1}^{k} p_j}$. □

Corollary 5.3. *Assume that there are k awake processors in a round, with broadcast probabilities* $p_1, \ldots, p_k$ *such that* $p_i \leq 1/2$ *for* $i = 1, \ldots, k$.
1. *If* $1/2 \leq \sum_{i=1}^{k} p_i \leq y$ *then the success probability is not smaller than* $\frac{1}{2}\left(\frac{1}{4}\right)^y \geq \left(\frac{1}{16}\right)^y$.
2. *If* $1/2 \leq \sum_{i=1}^{k} p_i \leq 2$ *then the success probability is not smaller than* $1/32$.
3. *If* $y \leq \sum_{i=1}^{k} p_i \leq 1/2$ *then the success probability is not smaller than* $y/2$.

6 The globally synchronous model with known n

In this section we consider the globally synchronous model where the number of processors, n, is known to all of them. Recall that in the case when n is known, there is no difference whether ε is given explicitly or as a function. Gąsieniec et al. [11] presented algorithm Repeated-Decay (based on the algorithm Decay from [3]) for this model. This algorithm achieves time complexity $O(\log n \log(1/\varepsilon))$ with probability $1-\varepsilon$. We present a modification of this algorithm that is *uniform* and achieves the same time complexity.

Algorithm Repeated Probability Decrease (RPD)
Let $l = 2\lceil \log n \rceil$. Each processor which wakes up spontaneously broadcasts a wakeup message in the round s (for every s) with probability $2^{-1-(s \mod l)}$.

Theorem 6.1. *The algorithm Repeated Probability Decrease succeeds in waking up the system in time* $O(\log n \log(1/\varepsilon))$ *with probability* $1-\varepsilon$.

Proof. Consider rounds $s, s+1, \ldots, s+l-1$ for any s such that $s \bmod l = 0$ and at least one processor is awake in the round s. So, at least one processor is awake in the round s and at most n are awake in the round $s+l-1$. It means that there exists $1 \leq i \leq l$ such that the number of awake processors in round $s+i+1$ is greater or equal to 2^{i-1} and smaller or equal to 2^i. For any woken up processor the probability of broadcasting in round $s+i+1$ is equal to $1/2^i$, so the broadcast sum is not smaller than $2^{i-1} \cdot \frac{1}{2^i} = \frac{1}{2}$ and not bigger than $2^i \frac{1}{2^i} = 1$. By Corollary 5.3, the success probability in this round is not smaller than $1/32$. Let a *phase* be a part of the computation which consists of rounds $s, s+1, \ldots, s+l-1$ for any s such that ($s \bmod l = 0$) and at least one processor is awake in the round s. Thus, the probability of success in one phase is at least $1/32$. Hence, the probability that none of the first $\log(1/\varepsilon(n))/\log(32/31)$ phases succeeds is at most $(31/32)^{\log(1/\varepsilon)/\log(32/31)} = \varepsilon$. $\square$

Kushilevitz and Mansour showed ([15], Lemma 3) that if l labeled processors are woken up spontaneously in the same round (for $l \in \{2^0, 2^1, \ldots 2^{\lceil \log n \rceil}\}$) then the expected number of rounds until the successful round is $\Omega(\log n)$ (the expectation is taken over l and probabilistic choices), even in case of a nonuniform algorithm, with known n and global clock. Thus, our algorithms that work with probability $1-\varepsilon$ in time $O(\log n \log(1/\varepsilon))$ are optimal for constant ε and their time bound differ only by a factor $\log(1/\varepsilon)$ from this lower bound in general case.

In the *uniform* case, the above algorithm almost matches the following lower bound that for $1/\varepsilon$ growing with n is tighter than the bound $\Omega(\log n)$ from [15].

Theorem 6.2. *Any uniform probabilistic algorithm (for globally synchronous model and known system size) requires* $\Omega\left(\frac{\log n \log(1/\varepsilon)}{\log\log n + \log\log(1/\varepsilon)}\right)$ *rounds to succeed in waking up the system with probability at least* $1-\varepsilon$*.*

Proof. Assume that $\varepsilon = \varepsilon(n) < 1/2 - c$ for any n large enough and some constant c (otherwise, we can apply Lemma 3 from [15]). We analyze only scenarios in which $m \geq 10$ processors are woken up spontaneously in the first step and no processor is woken up spontaneously later (all considerations below concern this scenario). Because of uniformity, in every round of the computation the broadcast probabilities of all awake processors are equal. Let m be the number of woken-up processors and p be the broadcast probability in a round. Then, the success probability in that round is $mp(1-p)^{m-1} \leq 3/4$, what we show by considering two cases:

Case 1: $p \leq 1/2$.

We have $mp(1-p)^{m-1} \leq (1/(1-p))mp(1/e)^{mp} \leq 2 \cdot mp(1/e)^{mp}$. Observe that $0 < mp \leq m/2$. For a function $f(x) = x(1/e)^x$ we have $f'(x) = (1/e)^x(1-x)$. So, the only local extremum of $f(x)$ is obtained for $x = 1$. Thus, by simple calculations one can see that the maximal value of the success probability is smaller than $2 \cdot 1 \cdot (1/e)^1 < 3/4$.

Case 2: $p > 1/2$.

We have $mp(1-p)^{m-1} \leq mp(1/2)^{m-1} \leq m(1/2)^{m-1} < 1/2$ for $m > 10$.

We say that a given round is *lost* if the success probability in this round is smaller than $1/(\log n \log(1/\varepsilon))$. Observe that the probability of unsuccessful work in $(1/2)\log n \log(1/\varepsilon)$ lost rounds is not smaller than $\left(1 - \frac{1}{\log n \log(1/\varepsilon)}\right)^{(1/2)\log n \log(1/\varepsilon)} \geq \left(\frac{1}{4}\right)^{1/2} = \frac{1}{2}$. Thus, in order to wake up all processors with probability $1-\varepsilon$, the probability of unsuccessful work in rounds that are not lost should be at most 2ε. Recall, the success probability is in every round not bigger than $3/4$. So, in order to get the probability of unsuccessful work smaller than 2ε, we need at least $\log_{3/4}(2\varepsilon) = \Theta(\log(1/\varepsilon))$ rounds that are not lost.

Let $x = \lceil 2(\log\log n + \log\log(1/\varepsilon)) \rceil$, $m_i = \lfloor x/2 \rfloor + (i-1)x$ for $i = 1, 2, \ldots, \lfloor \log n / x \rfloor - 1$. As shown in the claim that follows, there is no round r in the uniform computation such that r is not the lost round when 2^{m_i} processors are woken-up and r is not the lost round when 2^{m_j} processors are awake for $i \neq j$.

Claim. Let p be a broadcast probability in a round, let m be a number of awake processors. If $|\log m - \log(1/p)| \geq x$ and $m \geq \max\{10, \lceil \log\log n + \log\log(1/\varepsilon) \rceil\}$ then the round is lost.

Proof: Let p_s be a success probability in a round considered. First, assume that $p \geq 1/2$. Then $p_s = mp(1-p)^{m-1} \leq m(1/2)^{m-1} \leq (1/2)^{m/2} \leq 1/(\log n \log(1/\varepsilon))$ for every $m \geq \max\{10, \lceil \log\log n + \log\log(1/\varepsilon)\rceil\}$.

Now, consider the case $p \leq 1/2$. Then, $p_s = mp((1-p)^{1/p})^{mp}/(1-p) \leq 2mp(1/e)^{mp}$. Note that $mp = 2^{\log m + \log p} = 2^{\log m - \log(1/p)}$. If $\log m - \log(1/p) > x$ then $mp > 2^x = (\log n \log(1/\varepsilon))^2$, so $p_s \leq 2mp \cdot (1/e)^{mp} \leq 2 \cdot (1/e)^{mp/2} \leq 1/(\log n \log(1/\varepsilon))$. If $\log(1/p) - \log m > x$ then $mp < 2^{-x} \leq \frac{1}{(\log n \log(1/\varepsilon))^2}$. Thus, $p_s \leq 2mp(1/e)^{mp} \leq 2mp \leq 1/(\log n \log(1/\varepsilon))$. □ *Claim*

Thus, for every $i = 1, 2, \ldots, \lfloor \log n/x \rfloor - 1$ we need $\Theta(\log(1/\varepsilon))$ rounds that are not lost. On the other hand, each round is not lost for not more than one value of i. So the time needed in order to obtain the success probability $1-\varepsilon$ is $\Omega(\log n \log(1/\varepsilon)/x) = \Omega\left(\frac{\log n \log(1/\varepsilon)}{\log\log n + \log\log(1/\varepsilon)}\right)$. □

7 Only the number of processors is known

In this section we consider the case when the number of processors, n, is known but processors are unlabeled and no global clock is available (recall that there is no difference between ε given as a constant parameter and as a known function of n in this case). As shown in following algorithm, one can obtain efficient solutions for these conditions, despite of the lack of the global synchronization.

Algorithm Probability Increase
Let $k = \lceil \log(1/\varepsilon(n))/\log(32/31)\rceil$, $l = \lceil \log n \rceil + 1$. After waking up spontaneously, every processor works in l phases numbered in decreasing order $l, l-1, \ldots, 1$. Each phase lasts k rounds. In each of these rounds the processor broadcasts a wakeup message with probability $1/2^j$, where j is the number of the phase.

Theorem 7.1. *Algorithm Probability Increase succeeds in waking up the system in time* $O(\log n \log(1/\varepsilon))$ *with probability at least* $1-\varepsilon$.

Proof. Consider the first round in which at least one processor is woken up. The sum of broadcast probabilities of all woken up processors is equal to at most $n \cdot \frac{1}{2^{\lceil \log n \rceil}} = \frac{1}{2}$ in this round. Starting from this round, the sum of broadcast probabilities of all woken up processors is growing. And, after at most $k\lceil \log n \rceil$ rounds, it is not smaller than $1/2$. Let r be the last round in which this sum is smaller than $1/2$. Note that the sum of broadcast probabilities of all processors woken up in rounds $r+1, r+2, \ldots, r+k$ is not greater than $1/2$ in each of these rounds. The sum of broadcast probabilities of all processors woken up till r is not bigger than 1 and not smaller than $1/2$ in each of these rounds. So, in each of these rounds, the sum of broadcast probabilities belongs to the interval $\langle 1/2; 3/2\rangle$. By Corollary 5.3, the success probability in each of these rounds is not smaller than $1/32$. Thus, the probability that every round in the sequence $r, r+1, \ldots, r+k$ is unsuccessful is not bigger than $(1-1/32)^k = \varepsilon$. □

8 Only labels are available

In this section we consider the model with labeled processors, without a global clock, where the number of processors n is unknown to them. We present an efficient algorithm that requires a direct access to a parameter ε. The main observation that enables to achieve efficient solution in this model is that one can use labels of processors as „local approximations" of the size of the network.

Algorithm Increase From Square (IFS)
Let $k = \lceil \log(1/\varepsilon)/\log(32/31)\rceil$, let $y = \pi^2/6$. Upon waking up spontaneously, the processor i performs the following:

1. $p \leftarrow \frac{1}{2^{\lceil \log(2y(i+1)^2) \rceil}}$
2. if $p \leq 1/2$ then in k consecutive rounds the processor works as follows: randomly set a bit b with probabilities $\mathbf{P}[b=1]=p$, $\mathbf{P}[b=0]=1-p$; if $b=1$, broadcast a wakeup message.
3. $p \leftarrow 2p$
4. Goto 2.

Theorem 8.1. *The algorithm Increase From Square succeeds in waking up all processors in time $O(\log n \log(1/\varepsilon))$ with probability at least $1-\varepsilon$.*

Proof. Consider the first round in which at least one processor is awake. The broadcast sum is not bigger than $\sum_{i=1}^{\infty} 1/(2^{\lceil \log(2yi^2) \rceil}) \leq \sum_{i=1}^{\infty} 1/(2yi^2) = 1/(2y)\sum_{i=1}^{\infty} 1/i^2 = 1/2$ in this round. Starting from this round, the broadcast sum is growing. Let r be the last round in which this sum is smaller than $1/2$. Let $s \leq 1/2$ be the value of this sum in the round r. Observe that the sum of probabilities of all processors that were woken up in the round r is not greater than $2s \leq 1$ and not smaller than $1/2$ in every round in the sequence $r+1, r+2, \ldots, r+k$. Moreover, the sum of broadcast probabilities of all processors woken up spontaneously in rounds $r+1, r+2, \ldots, r+k$ is not greater than $\sum_{i=1}^{\infty} 1/(2^{\lceil \log(2yi^2) \rceil}) \leq 1/2$ in all these rounds. So, the broadcast sum is in rounds $r+1, r+2, \ldots, r+k$ not smaller than $1/2$ and not bigger than $3/2$. By Corollary 5.3, the probability of success in each of these rounds is at least $1/32$. So, the probability that every round in this sequence was unsuccessful is at most $(1/32)^k \leq \varepsilon$.

Finally, observe that the largest possible distance between the first round in which at least one processor is woken up (spontaneously) and the last round in which the broadcast sum is smaller than $1/2$, is $O(k \log(2^{\lceil \log(2y(n+1)^2) \rceil})) = O(\log n \log(1/\varepsilon))$.

$\square$

9 Only a global clock is available

Now, we consider the model with a global clock, unlabeled processors, where the number of processors, n, is unknown to them. We present a fast algorithm for this scenario that does not require an explicit information about ε. First, we introduce a very useful notion of a factorial representation of natural numbers.

Definition 9.1 (Factorial representation). *Let $t > 0$ be a natural number. The factorial representation of t is equal to the sequence $t_1, t_2, \ldots$ such that $t = \sum_{i=1}^{\infty} t_i \cdot i!$ and $0 \leq t_i \leq i$ for every $i \in \mathbb{N}$ (and only finite number of t_i's are not equal to 0).*

Lemma 9.2. *For every natural number t there exists exactly one factorial representation of t.*

Definition 9.3. *Let t be a natural number, $\{t_i\}_{i \in \mathbb{N}}$ be a factorial representation of t. Let $j(t) = \min\{i \mid t_i = 0\}$, and:*

- $y(t) = 1$ *for* $t \in \{0,1\}$,
- $y(t) = y((j(t)-1)!-1) + 1 + \sum_{l=2}^{j(t)-1} (t_l - 1)(l-1)!$ *for* $t > 1$.

Before the analysis of this algorithm, we show some properties of factorial representations:

Lemma 9.4. *The functions y and j satisfy following conditions:*

1. $y((j(t)-1)!-1) \leq y(t) \leq y(j(t)!-1)$ *for every natural number* $t > 1$,
2. $y(k!-1) = 1! + 2! + \ldots + (k-1)!$ *for every* $k > 1$.

Proof. 1. Let $\{t_i\}_{i\in\mathbb{N}}$ be a factorial representation of t. The inequality $y((j(t)-1)!-1) \leq y(t)$ follows directly from the definition. So, we concentrate on the second inequality. Let $j(t)=k$. Observe that the factorial representation of $k!-1$ is equal to $1,2,3,\ldots,(k-1),0,0,0\ldots$ Thus, $j(t)=j(k!-1)=k$ and further $y(k!-1)=y((k-1)!-1)+1+\sum_{l=2}^{k-1}(l-1)(l-1)! \geq y((k-1)!-1)+1+\sum_{l=2}^{k-1}(t_l-1)(l-1)! = y(t)$, because $t_l \leq l$ for every l.
2. We prove this property by induction. For the base step, observe that $y(2!-1)=1=1!$ Now, assume that the property is true for $k-1$. Recall that $j(k!-1)=k$ and the factorial representation of $k!-1$ is equal to $1,2,3,\ldots,(k-1),0,0,0,\ldots$ Then, $y(k!-1)=y((k-1)!-1)+1+\sum_{l=2}^{k-1}(l-1)(l-1)!$ Thus, using inductive hypothesis, $y(k!-1)=(1!+2!+\ldots+(k-2)!)+1+\sum_{l=2}^{k-1}(l-1)(l-1)! = 1!+2!+\ldots+(k-2)!+(k-1)!$ □

We propose very simple but efficient algorithm based on properties of factorial representations:

Algorithm Use Factorial Representations (UFR)
Each woken processor performs the following in round t (for every t): Randomly set a bit b with probabilities $\mathbf{P}[b=1]=1/2^{y(t)}$, $\mathbf{P}[b=0]=1-1/2^{y(t)}$. If $b=1$, broadcast a wakeup message.

The following theorem establishes the complexity of Algorithm UFR.

Theorem 9.5. *Algorithm UFR is uniform and succeeds in waking up a globally synchronous system with probability* $1-\varepsilon$ *in time* $O(\log^2 n(\log\log n)^3\log(1/\varepsilon))$*, even if the number n and the parameter* ε *are unknown and processors are unlabeled.*

Proof. First, we show some useful facts that are crucial for the analysis of our algorithm:
Claim 1. For any $y\in\mathbb{N}$, there exists a natural number t such that $y(t)=y$; moreover, the smallest t such that $y(t)=y$ is not greater than $c_1 y\log y$ for some constant c_1.
Proof: Let y be a natural number, $(y_i)_{i\in\mathbb{N}}$ be a sequence denoting the factorial representation of $y-1$, k be the largest index such that $y_k\neq 0$. First, we show that there exists a *modified factorial* representation $(y'_1,\ldots,y'_{k'})$ of $y-1$ such that $y-1=\sum_{l=1}^{k'} y'_l l!$, where $0<y'_i\leq i+1$ for each $i\in\{1,\ldots,k'\}$, and $k'\leq k$. If $y_i>0$ for each $i=1,\ldots,k$ then the sequence $y_1,\ldots,y_k$ satisfies conditions of „modified" representation. Otherwise, we can transform the sequence $y_1,\ldots,y_k$ in order to obtain appropriate modified factorial representation. To this aim we can use the following algorithm that given a factorial representation $y_1,\ldots y_k$ (where $y_i=0$ for $i>k$), generates a modified factorial representation $y'_1,\ldots,y'_{k'}$.

```
y_0 ← 1
sub ← false;
For l = 1,2,...,k do
    if (y_l > 0) and (not sub) then y'_l ← y_l
    else if y_l = 0 then begin
            sub ← true;
            if y_{l-1} > 0 then y'_l ← l+1 else y'_l ← l
      end
      else begin{i.e.when y_l > 0 and sub =true}
          if (y_l > 1) or (l = k) then
          (y'_l ← y_l − 1; sub ← false)
          else (y'_l ← l+1; y_l ← 0; sub ← true)
      end
```

In other words, every subsequence of zeroes, say $y_a,\ldots,y_b$ (such that $y_{a-1}\neq 0$ and $y_{b+1}\neq 0$) is replaced into a sequence $y'_a,\ldots,y'_b$ such that $y'_a=a+1$ and $y'_c=c$ for $c=a+1,\ldots,b$. This is done by „borrowing" one occurrence of $(b+1)!$ from y_{b+1}, so then

$y'_{b+1} = y_{b+1} - 1$ (and possibly consecutive „borrowing" is needed if $y_{b+1} = 1$ and $b+1 < k$).

Using Lemma 9.4, we show that $y = y(t)$ for $t = 1 + \sum_{l=1}^{k'} y'_l (l+1)!$. The factorial representation of t is equal to $1, y'_1, y'_2, \ldots, y'_{k'}, 0, 0, \ldots$ Thus, $j(t) = k' + 2$ and $y(t) = y((k'+1)! - 1) + 1 + \sum_{l=1}^{k'} (y'_l - 1) l! (\sum_{l=1}^{k'} l!) + 1 + (\sum_{l=1}^{k'} (y'_l - 1)(l)!)(\sum_{l=1}^{k'} y'_l l!) + 1 = (y-1) + 1 = y$. For the second part of the claim, observe that $t = 1 + \sum_{l=1}^{k'} y'_l (l+1)! \le 1 + (k'+1) \sum_{l=1}^{k'} y'_l l! \le 1 + (k'+1)(y-1) \le (k'+1)y$, and $y \ge (k')!$. Thus, $k' \le d \log y$ (for some constant d, see Fact 5.1.3), so $t \le (k'+1)y \le 2dy\log y$. $\square$ *Claim 1*

Claim 2. For each y, the difference $t_2 - t_1$ between every two consecutive numbers $t_1 < t_2$ such that $y(t_1) = y(t_2) = y$ is not bigger than $c_2 y \log^3 y$ (for a constant c_2).

Proof: Let y be a natural number, let t be any number that satisfies $y(t) = y$, let $j = j(t)$. First, observe that $y(t) = y(t + (j+1)!)$. Indeed, first j numbers in the factorial representation of $t' = t + (j+1)!$ are equal to the first j numbers in factorial representation of t. So, $j(t') = j(t) = j$ and $y(t') = y(t)$. Now, we must only show that $(j+1)! = O(y \log^3 y)$. By Lemma 9.4, $y((j-1)! - 1) \le y \le y(j! - 1)$. Thus (also by Lemma 9.4), $1! + 2! + \ldots + (j-2)! \le y \le 1! + 2! + \ldots + (j-1)!$ It means that $y \ge (j-2)!$ and $\log y \ge e(j-2)\log(j-2)$ for some constant e. In consequence, $j + 1 \le f \log y$ (for a constant f). Finally, $(j+1)! \le (j-2)!(j+1)^3 \le f^3 y \log^3 y$. $\square$ *Claim 2*

Now, we apply these properties to the analysis of our algorithm. Observe that in every round all awake processors have the same broadcast probability. If this probability is $1/2^y$ and the number of awake processors is in the range $\langle 2^{y-1}, 2^{y+1} \rangle$ then the broadcast sum is not smaller than $2^{y-1} \cdot \frac{1}{2^y} = \frac{1}{2}$ and not bigger than $2^{y+1} \cdot \frac{1}{2^y} = 2$. Thus, the probability that the round is successful is not smaller than $1/32$, by Corollary 5.3. After $\Theta(\log(1/\varepsilon(n))$ such rounds we obtain appropriate probability of success. Let $y = \lceil \log n \rceil$. For any $m = 1, 2, \ldots, \lceil \log n \rceil$ and for every $y \log^3 y = O(\log n (\log\log n)^3)$ consecutive rounds there is a round in which each (awake) processor has broadcast probability $1/2^m$ (by Claims 1 and 2). Let us split the computation into blocks of length $y \log^3 y$ (starting from the first round in which at least one processor is woken up). In each block, at least one of the following two conditions is satisfied:

(1) There is at least one round t in the block such that $y(t) = x$ and the number of woken up processors in t is in the range $\langle 2^{x-1}, 2^{x+1} \rangle$.
(2) The number of awake processors at the end of the block is at least twice larger than the number of awake processors at the beginning of the block.

There are at most $\lceil \log n \rceil$ blocks of type (2). Moreover, the probability that none of b blocks of type (1) is successful is at most $(1 - 1/32)^b$. So, we get required probability of success after $\log n + \log_{32/31}(1/\varepsilon(n)) = O(\log n + \log(1/\varepsilon(n))$ blocks. Time consumed by these blocks is $O(\log n (\log\log n)^3 (\log n + \log(1/\varepsilon(n)))) O(\log^2 n (\log\log n)^3 \log(1/\varepsilon(n)))$. $\square$

10 The weakest model

In this section we consider the weakest model, without global clock, with unlabeled processors, where the number of processors, n, is unknown to them. First, we show an almost linear lower bound which contrasts to polylogarithmic algorithms obtained for stronger models.

Theorem 10.1. *If a global clock is not available, the size n of the system is not known to processors and processors do not have labels then every probabilistic algorithm needs at least $\Omega(n/\log n)$ rounds in order to wake up the system of n processors with probability $1 - \varepsilon$.*

Proof. In this model, the algorithms of all processors are identical and may be described by a (infinite) sequence $p_1, p_2, p_3, \ldots$ such that p_i is the broadcast probability in the ith round after spontaneous wake-up of the processor. If $p_i = 0$ for every $i \in \mathbb{N}$ then the algorithm is incorrect (an adversary may never wake-up all processors). Let $j = \min\{i \mid p_i > 0\}$, let $p_j = 1/p$ (n is unknown, thus we can assume that p is constant with respect to n). For a natural number n, let $r_n = \lfloor n/(\lceil 4p\log n \rceil) \rfloor$. We take any (large enough) n that satisfies inequalities $(1/n)^4 \leq 1/(2n)$, $8\log n/n^2 \leq 1/(2n)$ and $r_n \leq n\log(1/\varepsilon)/2$. Let us call a round in which first processor is woken up spontaneously as the round 1. Assume that the adversary wakes up spontaneously $x = \lceil 4p\log n \rceil$ processors in each round in the sequence $1, 2, \ldots, \lfloor n/x \rfloor = \Theta(n/\log n)$. The probability of success in rounds $1, \ldots, j-1$ is equal to zero. Let U_k be a set of processors woken up in the round k. We show that for any $k \in \{j, j+1, \ldots, \lfloor n/x \rfloor\}$, the probability that at least two processors from U_{k-j+1} send broadcast message in the round k is not smaller than $1 - 1/n$. First, observe that the probability that none of the processors from U_{k-j+1} broadcasts is $(1-1/p)^x \leq (1-1/p)^{4p\log n} \leq (1/2)^{4\log n} \leq (1/n)^4 \leq 1/(2n)$. Moreover, the probability that exactly one element of U_{k-j+1} broadcasts is $\frac{x}{p}(1-1/p)^{x-1} \leq \frac{8p\log n}{p}(1-1/p)^{2p\log n} \leq (8\log n)/n^2 \leq 1/(2n)$. Concluding, the probability that more than one processor from U_{k-j+1} is broadcasting in the round k is at least $1 - 1/(2n) - 1/(2n) = 1 - 1/n$.

Thus, the probability of success in each round from the sequence $j, j+1, \ldots, \lfloor n/x \rfloor$ is smaller than $1/n$. The probability of unsuccessful work in rounds $1, \ldots, \lfloor n/x \rfloor$ is not smaller than $(1-1/n)^{r_n} > (1-1/n)^{n\log(1/\varepsilon)/2} \geq (1/4)^{log(1/\varepsilon)/2}\varepsilon$. □

Now, we present an algorithm working in the model with known and constant ε that matches above lower bound.

Algorithm Decrease From Half

Let $k = \lceil \log(1/\varepsilon)/\log(32/31) \rceil$. Upon waking up spontaneously, each processor performs the following (until all processors are woken up):

1. $p \leftarrow 1/2$
2. In k consecutive rounds: randomly set a bit b with probabilities $\mathbf{P}[b=1] = p$, $\mathbf{P}[b=0] = 1-p$; if $b = 1$, broadcast a wakeup message.
3. $p \leftarrow p/2$
4. Goto 2.

Theorem 10.2. *The algorithm Decrease From Half succeeds in waking up all processors in time* $O\left(\frac{n\log(1/\varepsilon)}{\log n}\right)$ *with probability* $1-\varepsilon$.

Proof. Note that for each processor the sum of broadcast probabilities in all phases during the work of the algorithm is not larger than $k = \sum_{i=1}^{\infty} k \cdot \frac{1}{2^i}$. Thus, the sum of broadcast probabilities of all processors during the work of the algorithm is not bigger than kn. Let us concentrate on the work of the protocol in first $\frac{4kn}{\log_{16} n}$ rounds, starting in the round in which the first processor is woken up spontaneously. It follows from above discussion that in at least half of these rounds, i.e. $\frac{2kn}{\log_{16} n}$, the sum of broadcast probabilities of woken up processors is not bigger than $2 \cdot \frac{kn}{4kn/\log_{16} n} = \frac{1}{2}\log_{16} n$. We call rounds that satisfy this condition *light* . Consider two cases:

Case 1. There exists a light round j such that the sum of broadcast probabilities of processors that are awake in this round is not bigger than $1/2$. Then, for each of k rounds that directly precede j, the sum of broadcast probabilities is not smaller than $1/2$ and not bigger than 1 (because the sum of broadcast probabilities can become smaller at most twice in k rounds). By Corollary 5.3, the probability of success in each of these rounds is not smaller than $1/32$. Thus, error probability is not bigger than $(1-1/32)^k = \varepsilon$.

Case 2. In each of light rounds the sum of broadcast probabilities of awake processors is bigger or equal to $1/2$. Recall that in each of light rounds the sum of broadcast probabilities is smaller or equal to $\frac{1}{2}\log_{16} n$. Thus, by Corollary 5.3, the probability of success in light rounds is not smaller than $(1/16)^{\frac{1}{2}\log_{16} n} = 1/\sqrt{n}$. Thus, the probability of unsuccessful work is not bigger than $(1-1/\sqrt{n})^{\frac{2kn}{\log_{16} n}} \leq (1-1/\sqrt{n})^{k\sqrt{n}} \leq \varepsilon$. □

Finally, we present an algorithm for the weaker scenario, when ε is given as a function of n, not as a constant parameter. This algorithm works for any $\varepsilon(n)$ that polynomially goes to zero as n is growing.

Algorithm Decrease Slowly
Assume that $\varepsilon(n) \geq 1/n^r$ for a constant r. Let $q = 16r\ln 2$. Any processor, after waking up spontaneously performs the following:

1. $i \leftarrow 0$
2. $p \leftarrow q \cdot \frac{1}{2q+i}$
3. In one round: randomly set a bit b with probabilities $\mathbf{P}[b=1] = p$, $\mathbf{P}[b=0] = 1-p$; if $b=1$, broadcast a wakeup message.
4. $i \leftarrow i+1$
5. Goto 2.

Theorem 10.3. *Algorithm Decrease Slowly wakes up a system of n processors in time* $O(nlog(1/\varepsilon(n)))$ *with probability at least* $1-\varepsilon(n)$.

Proof. Assume that n is a natural power of two. Let us analyze the work of the algorithm during the first $2m$ rounds (starting from the first round in which at least one processor is woken up), for $16qn \leq m < n^2/2$. Note that the sum of broadcast probabilities of one processor in all these $2m$ rounds is not bigger than $q\sum_{i=0}^{n^2} 1/(2q+i) \leq q\log(2n^2) \leq q(2\log n+1) \leq 4q\log n$. Thus the sum of broadcast probabilities of all processors in first $2m$ rounds of the algorithm is not bigger than $4qn\log n$. This implies that in at least m of the first $2m$ rounds, the broadcast sum is not bigger than $\frac{4qn\log n}{m} \leq \frac{\log n}{4}$ (indeed, otherwise the sum over all rounds would be bigger than $4qn\log n$). Let rounds with broadcast sum smaller than $\log n/4$ be called *light*. Let us split *light* rounds into two categories:
Category 1: The rounds with broadcast sum bigger than $1/2$. By Corollary 5.3.1, the success probability in each such round is bigger than $(1/16)^{(\log n)/4} = 1/n$. If the number of rounds of category 1 is bigger than $n\ln(1/\varepsilon(n))$ then the probability of unsuccessful work is smaller than $(1-1/n)^{n\ln(1/\varepsilon(n))} \leq (1/e)^{\ln(1/\varepsilon(n))} = \varepsilon(n)$.
Category 2: The *light* rounds with broadcast sums not bigger than $1/2$. First note that:
Claim 1
(a) Assume that the broadcast sum in the round j is not smaller than $q/(2q+i)$. Then the broadcast sum in the round $j+1$ is not smaller than $q/(2q+i+1)$.
(b) The broadcast sum of the first light round of Category 2 is not smaller than $q/(2q+1)$.
Proof: (a) Let U_j be the set of processors that are awake in the round j, let s_j be the broadcast sum in the round j. Then $s_j = q\sum_{l\in U_j} 1/(2q+i_l)$, where i_l are natural numbers. So, $s_{j+1} \geq q\sum_{l\in U_j} 1/(2q+i_l+1)$. If there exists $k \in U_j$ such that $i_k \leq i$ then $s_{j+1} \geq q/(2q+i_k+1) \geq q/(2q+i+1)$. Otherwise, $s_{j+1} \geq q\sum_{l\in U_j} \frac{1}{2q+i_l}\cdot\frac{2q+i_l}{2q+i_l+1} \geq q\sum_{l\in U_j}\frac{1}{2q+i_l}\cdot\frac{2q+i}{2q+i+1} = \frac{2q+i}{2q+i+1}s_j \geq \frac{2q+i}{2q+i+i}\cdot\frac{q}{2q+i} = \frac{q}{(2q+i+1)}$.
(b) Note that the broadcast sum of the first round during the work of the algorithm is not smaller than $q\cdot\frac{1}{2q}$. The rest follows from (a). □ *Claim 1*

One can show by induction that the broadcast sum of the ith *light* round of Category 2 is not smaller than $q/(2q+i)$. By above claim, it is true for $i=0$. For the inductive step, let us only comment the case when the ith and the $(i+1)$st *light* round of Category 2 are not consecutive in the execution of the algorithm. Note that then the broadcast sums of the rounds between them are bigger than the broadcast sum in the round i. So, our statement is satisfied by the application of Claim 1.

Thus by Corollary 5.3.3 the broadcast probability in the ith *light* round of Category 2 is not smaller than $\frac{1}{2} \cdot \frac{q}{2q+i}$. So the probability of unsuccessful work in first x *light* rounds of Category 2 is not bigger than $\prod_{i=1}^{x} \left(1 - \frac{1}{2} \cdot \frac{q}{2q+i}\right) \leq \left(\frac{1}{e}\right)^{\frac{q}{2} \sum_{i=1}^{x} \frac{1}{2q+i}} \leq \left(\frac{1}{e}\right)^{\frac{1}{2} \cdot \frac{q \log x}{4}} = x^{-2r}$. In order to bound this probability by $\varepsilon(n)$ we need $x = (\varepsilon(n))^{-1/(2r)} = \sqrt{n}$ *light* rounds of Category 2.

Finally, if $m \geq 2 \cdot \max(n \ln(1/\varepsilon(n)) + \sqrt{n}, 16qn)$ then we get required success probability. □

11 Conclusions

We presented efficient probabilistic algorithms waking up all processors in polylogarithmic time with probability $1 - \varepsilon$ if global clock or knowledge of n or labeling is accessible. These algorithms substantially improve previous results, and show that efficient solutions are possible even without the knowledge of the basic parameters of the network. Further, we showed an almost linear lower bound for the weakest model and we presented an algorithm that matches this bound. We think that our results may be helpful in designing algorithms for other problems in (not only single hop) radio networks working without global synchronization (or without labels).

Some interesting problems remain open. First, there is a gap of size $\log(1/\varepsilon)$ between lower bounds and our algorithms for the „standard" model with labels and ε given as a parameter. Second, is it possible to construct polylogarithmic (or even sublinear) randomized algorithm for the model with local clocks, n unknown, known labels and ε given as a function? The best solution known for us is a protocol in which each processor broadcasts in each step with probability equal to $1/l$, where l is the label of the processor. However, this protocol requires $\Theta(n \log(1/\varepsilon))$ steps for waking up all processors with probability $1 - \varepsilon(n)$. (note that that the broadcast sum in each round belongs to the interval $\langle 1/(2n), \lceil \log(n+1) \rceil \rangle$).

Finally, observe that some of our protocols (e.g. ....) have a disadvantage that their expected time is infinite, because they do not wake-up all processors with some small probability. An interesting question is whether good solutions with small expected time are possible in these cases (note that, for globally synchronous environment, the algorithm UFR seems to be quite efficient when expected time is considered).

References

1. N. Alon, A. Bar-Noy, N. Linial, D. Peleg, *A Lower Bound for Radio Broadcast*, Journal of Computer Systems Sciences 43, (1991), 290–298.
2. R. Bar-Yehuda, O. Goldreich, A. Itai, *Efficient Emulation of Single-Hop Radio Network with Collision Detection on Multi-Hop Radio Network with no Collision Detection*, Distributed Computing 5 (1991), 67-71.
3. R. Bar-Yehuda, O. Goldreich, A. Itai, *On the Time-Complexity of Broadcast in Multi-hop Radio Networks: An Exponential Gap Between Determinism and Randomization*. JCSS 45(1), (1992), 104-126.
4. I. Chlamtac, S. Kutten, *On Broadcasting in Radio Networks – Problem Analysis and Protocol Design*, IEEE Trans. on Communications 33, 1985.
5. 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.
6. B. Chlebus, L. Gąsieniec, A. Lingas, A.T. Pagourtzis, *Oblivious gossiping in ad-hoc radio networks*, in Proc., DIALM for Mobility: 5-th Int. Workshop on Discrete Algorithms and Methods for Mobile Computing and Communications, 2001, Rome, Italy, 44–51.

7. M. Chrobak, L. Gąsieniec, W. Rytter, *Fast Broadcasting and Gossiping in Radio Networks*, in Proc. FOCS, 2000, 575-581.
8. S. Even, S. Rajsbaum, *Unison, canon, and sluggish clocks in networks controlled by a synchronizer*, Mathematical Systems Theory 28, (1995), 421–435.
9. M.J. Fischer, S. Moran, S. Rudich, G. Taubenfeld, *The Wakeup Problem*, SIAM Journal of Computing 25, (1996), 1332–1357.
10. P. Fraigniaud, A. Pelc, D. Peleg, S. Perennes, *Assigning labels in unknown anonymous networks*, Distributed Computing 14(3), 163-183 (2001).
11. L. Gąsieniec, A. Pelc, D. Peleg, *The wakeup problem in synchronous broadcast systems*, PODC 2000, 113–121. Journal version: *The Wakeup Problem in Synchronous Broadcast Systems*, SIAM Journal on Discrete Mathematics 14(2): 207-222 (2001).
12. J. Goodman, A. G. Greenberg, N. Madras, P. March, *On the stability of Ethernet*, 17th ACM-STOC, 1985, 379–387.
13. J. Hastad, T. Leighton, B. Rogoff, *Analysis of Backoff Protocols for Multiple Access Channels*, 19th ACM-STOC, 1987, 241–253.
14. P. Indyk, *Explicit constructions of selectors and related combinatorial structures, with applications*, Proc. SODA, 2002.
15. E. Kushilevitz, Y. Mansour, *An* $\Omega(Dlog(N/D))$ *Lower Bound for Broadcast in Radio Networks*, SIAM J. Comput. 27(3), (1998), 702–712.
16. J. Mazoyer, *On optimal solutions to the firing squad synchronization problem*, Theoretical Computer Science 168, (1996), 367–404.
17. K. Nakano, S. Olariu, *Energy-Efficient Initialization Protocols for Single-Hop Radio Networks with no Collision Detection*, IEEE Trans. on Parallel and Distributed Systems, Vol. 11, No. 8, pp. 851–863, Aug. 2000.
18. D. Peleg, *Deterministic radio broadcast with no topological knowledge*, a manuscript, 2000.
19. D. E. Willard, *Log-logarithmic selection resolution protocols in multiple access channel*, SIAM Journal on Computing 15 (1986), 468–477.