The Wake-Up Problem in Multi-Hop Radio Networks *

Marek Chrobak[†] Leszek Gąsieniec[‡] Dariusz Kowalski[§]

April 6, 2004

Abstract

We study the problem of waking up a collection of n processors connected by a multi-hop *ad-hoc* ratio network with unknown topology, no access to a global clock, and no collision detection mechanism available. Each node in the network wakes-up spontaneously, or it is activated by receiving a wake-up signal from another node. All active nodes transmit the wake-up signals according to a given protocol $\mathcal{W}$. The running time of $\mathcal{W}$ is the number of steps counted from the first spontaneous wake-up, until all nodes become activated.

We provide two protocols for this problem. The first one is a deterministic protocol with running time $O(n^{5/3} \log n)$. Our protocol is based on a novel concept of a rotation-tolerant selector to which we refer as a (radio) *synchronizer*. The second protocol is randomized, and its expected running time is $O(D \log^2 n)$, where D is the diameter of the network.

Subsequently we show how to employ our wake-up protocols to solve two other communication primitives: leader election and clock synchronization.

1 Introduction

We define an *ad-hoc* multi-hop radio network as a directed, strongly-connected graph with n vertices, whose nodes represent processors and outgoing edges represent their transmission ranges. The execution time is divided into discrete time steps, same for all processors. We assume that each processor is assigned a unique integer identifier from a range of size $O(n)$. The processors know the range of the identifiers, but they do not know the network's topology.

We consider the fundamental problem of waking up the processors of an ad-hoc multi-hop radio network. Initially, all nodes are assumed to be asleep. Each node in the network can wake-up spontaneously, or can be activated by receiving a wake-up signal from another node. Once a node v is activated, it starts executing its wake-up protocol $\mathcal{W}$. This protocol dictates during which time steps v will transmit a wake-up signal. In our model, the network does not have a global clock, so $\mathcal{W}$ depends only on the local clock of v (the number of steps since activation), and possibly additional information received earlier from other nodes. The wake-up signal from v is sent, during the same time step, to all out-neighbors of v. However, an out-neighbor u of v receives this signal only if no collision occurred, that is, if no other

*A preliminary version of this paper appeared in Proceedings of Symposium on Discrete Algorithms (SODA'04), pages 985-993.

[†]Department of Computer Science, University of California, Riverside, CA 92521, USA. Email: `marek@cs.ucr.edu`. Research supported by NSF grants CCR-9988360 and CCR-0208856. (Contact author.)

[‡]Dept. of Computer Science, University of Liverpool, Liverpool L69 7ZF, UK. Email: `leszek@csc.liv.ac.uk`. Research supported by the EPSRC grant GR/R84917/01.

[§]Max-Planck-Institut fuer Informatik, Stuhlsatzenhausweg 85, 66123 Saarbruecken, Germany, and Instytut Informatyki, Uniwersytet Warszawski, Banacha 2, 02–097, Warszawa, Poland. Email: `darek@mpi-sb.mpg.de`. Research supported in part by KBN grant 4T11C04425.

in-neighbor of u transmitted at the same time. Furthermore, we do not assume any collision detection capability, so if u does not receive a signal at time t, it has no way to tell whether a collision occurred or whether none of its neighbors transmitted. The running time of $\mathcal{W}$ is the number of time steps, counting from the first spontaneous wake-up, until all nodes become activated.

While other important communication primitives, such as *broadcasting* (one-to-all communication), *gossiping* (all-to all communication) and *multicast* (some-to-some communication) have been explored extensively in the context of ad-hoc multi-hop radio networks, e.g., see [3, 4, 5, 6, 7, 8, 9, 11, 12, 15, 18, 19, 20, 23], the wake-up and synchronization problems were studied primarily for *one-hop* networks (i.e., in the presence of a complete graph of connections). For one-hop networks, Gąsieniec *et al.* [14] presented a deterministic protocol with running time $O(n^2 \log n)$. They also gave a probabilistic argument showing that there exists a wake-up protocol with running time $O(n \log^2 n)$. Later, Indyk [15] showed that the same proof can be turned into a constructive argument, with only a slight increase in the running time. In the randomized case, in [14] one can find an algorithm that, for any given $\epsilon > 0$, completes the wake-up process in time $O(n \log(1/\epsilon))$ with probability at least $1 - \epsilon$. This result was recently improved by Jurdziński and Stachowiak [17] who gave a protocol with running time $O(\log n \log(1/\epsilon))$.

Note that broadcasting can be reduced to wake-up, in the following sense. Given a wake-up protocol $\mathcal{W}$, we can use it to perform broadcasting by waking-up only the node s that wants to initiate the broadcast, and then using the transmissions of $\mathcal{W}$ to transmit the broadcast message instead of the wake-up signal. Therefore the wake-up problem can be thought of as a generalization of broadcasting.

Our results. We study the wake-up problem in arbitrary ad-hoc multi-hop radio networks (far more general than the one considered in [14, 17]), when the network has an arbitrary topology and, further, this topology is not known to the algorithm. We provide two protocols for this problem.

First, we give a probabilistic construction of a deterministic wake-up protocol with running time $O(n^{5/3} \log n)$. Our construction works in several stages. In the first stage, we construct certain combinatorial structures called (radio) *synchronizers*. Later, we take a random protocol $\mathcal{P}$ that consists of a synchronizer followed by a random sequence of transmissions, where each transmission is executed with probability $1/n$. We prove that, with very high probability, this random protocol completes the wake-up task in time $O(n^{5/3} \log n)$ in so-called *path graphs*. This implies, via the probabilistic method, that there exists a deterministic protocol $\mathcal{D}$ for path graphs with the same running time. Finally, we prove that $\mathcal{D}$ works, in fact, for arbitrary strongly-connected graphs.

As our second result, we give a randomized wake-up protocol $\mathcal{R}$ whose expected running time is $O(D \log^2 n)$, where D is the diameter of the network.

Completing the wake-up alone is necessary but not sufficient for effective communication between the processors. Thus, we subsequently study two other communication primitives: leader election and clock synchronization. In the *leader election* problem, we want to designate one node, typically the one with minimum label, as the leader. All nodes have to be informed about the leader identity (label). In the *clock synchronization* problem, we require that, after the completion of the protocol, all nodes agree on some integer value (typically, the local time of one selected node) as the current global time. Assuming that all node labels are integers drawn from an $O(n)$-size range, we show that both tasks, leader election and clock synchronization, can be achieved deterministically in time $O(n^{5/3} \log^2 n)$, and in time $O(D \log^3 n)$ with randomization.

Other related work. Wake-up problem in ad-hoc wireless networks was considered by Zheng *et al.* [24]. They designed energy efficient protocol in bounded-asynchrony environment. A variant of a wakeup problem in asynchronous shared memory was studied by Fischer *et al.* [13].

The leader election problem has been widely studied in the area of distributed computing [22]. Jurdziński *et al.* [16] investigated randomized algorithms for leader election in one-hop radio networks.

Awerbuch *et al.* [2] considered synchronization-like problems in asynchronous message-passing distributed system. We also point out that the term "synchronizer" was used in other contexts in distributed computing (see [1]). Our definition of synchronizers is different than those used in the setting of asynchronous distributed message-passing systems.

2 Preliminaries

Radio networks. A *radio network* is a directed, strongly-connected graph G with n vertices that represent processors. If there is an edge from u to v, then we say that v is an *out-neighbor* of u and u is an *in-neighbor* of v. The set of all in-neighbors of v is denoted by $N_G(v)$, or simply $N(v)$ if G is understood from context.

To simplify presentation, throughout the paper we assume, without loss of generality, that n is a power of 2. Further, we often use other functions of n, say $n^{1/3}$, $n^{5/3}$, etc., as if their values were integer. All the calculations can be easily formalized by using the appropriately rounded values instead, say $\lfloor n^{1/3} \rfloor$, $\lceil n^{1/3} \rceil$, etc., although we chose not to do it in order to avoid clutter.

Each node v is assigned a unique identifier ℓ_v, also called a *label*. The labels are integer numbers from an interval $\{\ell_{\text{min}}, \ldots, \ell_{\text{max}}\}$, where $\ell_{\text{max}} - \ell_{\text{min}} = O(n)$. Initially, each node knows only its label and the bounds ℓ_{min}, ℓ_{max}. In the paper we assume that $\ell_{\text{min}} = 0$, since otherwise a node v instead of label ℓ_v can use an auxiliary label $\ell_v - \ell_{\text{min}}$ for execution. In the construction of synchronizers and wake-up protocols, we further simplify the notation by assuming, without loss of generality, that the label set is $V = \{0, 1, \ldots, n-1\}$, so that we can identify nodes by consecutive integers. (Note that we cannot make this assumption in the leader election and synchronization protocols, because then node 0 could be always selected as the leader, making both problems trivial.)

The execution time is divided into discrete time steps. For the sake of clarity, we make a distinction between the terms "time" and "time step". A *time* t is the point t on the time axis. A *time step* t is the unit time interval $[t, t+1)$. Any actions (activations, transmissions, etc.) take place during time steps. Properties of the network (for example, the set of active nodes) are observed at time points. If some action takes place at time step t, its effect will be reflected by the change of the state of the network from time t to $t+1$.

Since our focus is on communication, we assume that processors have unlimited computing power and can perform arbitrary computations within one time step. However, only one transmission or message receipt is allowed in one time step.

By a *wake-up schedule* we mean any vector $\omega = (\omega_x)_{x \in V}$, where ω_x denotes the time step in which x wakes-up spontaneously. For any set $X \subseteq V$, by ω_X we denote the earliest wake-up time step in X, that is $\omega_X = \min_{x \in X} \omega_x$. For our convenience, we will be assuming that $\omega_V = \min_{x \in V} \omega_x = 0$. A node v with $\omega_v = 0$ is referred to as a *start node*. Note that there may be several start nodes present. A node x is called *active* at time t if it either woke-up spontaneously or if it was activated by another active node u by receiving its wake-up signal in some time step $t' \leq t-1$.

A *wake-up protocol* $\mathcal{W}$ is a function that, for each label ℓ and for each step t, given all past messages received by the node v with label $\ell_v = \ell$, specifies whether v will transmit the wake-up signal in step t (since its activation). A message M transmitted in step t from a node v is sent instantly to all its out-neighbors. However, an out-neighbor u of v receives M in time step t only if no collision occurred, i.e., if the other in-neighbors of u do not transmit in step t at all. Further, collisions cannot be distinguished from the background noise. If u does not receive any message in step t, it knows that either none of its in-neighbors transmitted in step t, or that at least two did, but it does not know which of these two events occurred.

Given a network G and a wake-up schedule ω, by a *wake-up network* we will mean the pair $\langle G, \omega\rangle$. The wake-up network, together with a (deterministic) wake-up protocol $\mathcal{W}$ fully determine the state of the network at each time. Let $\mathit{acttime}_v(\mathcal{W}, G, \omega)$ denote the activation time step of node v, that is, either ω_v or the time step when v received a wake-up message, whichever comes first. Note that, by the definition of active nodes, v is active for the first time at time $\mathit{acttime}_v(\mathcal{W}, G, \omega) + 1$. By $\mathit{ActSet}_t(\mathcal{W}, G, \omega)$ we denote the set of all active nodes at time t. We will often simplify this notation and omit the arguments that are well understood from context, writing $\mathit{acttime}_v(\omega)$, ActSet_t, etc.

The running time of a wake-up protocol $\mathcal{W}$ is the smallest T such that, for any wake-up network $\langle G, \omega\rangle$ on V, all nodes are activated by time T, that is $\mathit{ActSet}_T(\mathcal{W}, G, \omega) = V$. Note that according to this definition some nodes may continue transmission after time T. However, without loss of generality we can assume that each node stops transmitting after T steps from its activation (any further transmissions are redundant.) Then the total termination time – from the first wake-up until all nodes complete their transmissions – is at most $2T$.

Protocols for leader election and synchronization, as well as their running times, are defined in an analogous manner. We remark here, however, that in our wake-up and leader election protocols, nodes only transmit wake-up signals to its neighbors; no other messages are used. For clock synchronization, messages include numerical values representing the global time.

3 Synchronizers

In this section we introduce a new type of *selector-like* structures (for the definition of selectors see, e.g., [9]) which "tolerate" arbitrary transpositions of transmission protocols used by the nodes of the network. We refer to the new type of selectors as *radio synchronizers* (or *synchronizers* for short), since they are used to coordinate the transmissions of nodes whose local clocks lack synchronization.

Let $\mathcal{S} = \{\mathcal{S}^x\}_{x\in V}$, where each $\mathcal{S}^x = S^x_1 S^x_2 \ldots S^x_m$ is a binary 0-1 sequence of length m. We say that $\mathcal{S}$ is a (n, k, m)-*synchronizer* if it satisfies the following property:

($*$) For any set $X \subseteq V$ of cardinality at most k, and for any wake-up schedule ω, there exists t, where $\omega_X < t \le \omega_X + m$, such that,

$$\sum_{x\in X} S^x_{t-\omega_x} \;=\; 1.$$

We assume here that $S^x_i = 0$, for $i < 0$.

One can interpret $\mathcal{S}$ as a transmission protocol, where $S^x_i = 1$ indicates that node x transmits in step $\omega_x + i$. Thus the condition ($*$) states that, in at most m steps after the first node in X wakes-up, there will be a time step when exactly one node in X transmits.

Lemma 1 *Let $C \ge 31$ be an integer constant. For each n and $k \le n$, there exists an (n, k, m)-synchronizer with $m = Ck^2 \log n$.*

Proof: We use here a probabilistic argument. Without loss of generality, we can assume that $n \ge k \ge 2$. For all $x \in V$ and $i = 1, \ldots m$, independently, assign $S^x_i = 1$ with probability $1/k$, and 0 otherwise. We show that, with very high probability, $\mathcal{S} = \{\mathcal{S}^x\}_{x\in V}$ is an (n, k, m)-synchronizer.

Initially, we fix a set of nodes X with $|X| \le k$ and a wake-up schedule ω. Pick $z \in X$ with $\omega_z = \omega_X$. Fix a time $t \in (\omega_X, \omega_X + m]$, and let $X' = \{x \in X : \omega_x < t\}$ be the set of nodes in X that are "awaken" at time t. By the independence of the random variables S^x_t, for $x \in X$, we have

$$
\begin{aligned}
\Pr\big[\textstyle\sum_{x\in X} S^x_{t-\omega_x} = 1\big] &\geq \Pr\Big[S^z_{t-\omega_z} = 1 \ \&\ \textstyle\sum_{x\in X'-\{z\}} S^x_{t-\omega_x} = 0\Big] \\
&= \frac{1}{k}\left(1-\frac{1}{k}\right)^{|X'|-1} \\
&\geq \frac{1}{k-1}\left(1-\frac{1}{k}\right)^{k} \\
&\geq \frac{1}{4k},
\end{aligned}
$$

where the last inequality follows from $k \geq 2$.

For different times t, the events "$\sum_{x\in X} S^x_{t-\omega_x} \neq 1$" are independent. Therefore, using the bound above, we get

$$
\begin{aligned}
\Pr\big[\textstyle\sum_{x\in X} S^x_{t-\omega_x} \neq 1 \ \ \forall t \in (\omega_X, \omega_X + m]\big] &\leq \left(1-\frac{1}{4k}\right)^{m} \\
&\leq \beta^{m/k},
\end{aligned}
$$

for $\beta = e^{-1/4} < 1$.

The probability that $\mathcal{S}$ violates the definition of the (n,k,m)-synchronizer for a fixed X, can be estimated by multiplying the above probability by the number of choices of ω. It is sufficient to count only those wake-up schedules ω that satisfy $0 \leq \omega_x \leq m$ for $x \in X$, and $\omega_x = 0$ for $x \notin X$. This is because, given an arbitrary ω, we can define another wake-up schedule ω' as $\omega'_x = \min\{\omega_x - \omega_X, m\}$, for $x \in X$, and $\omega'_x = 0$ for $x \notin X$. Then $\mathcal{S}$ satisfies $(*)$ for X and ω iff $\mathcal{S}$ satisfies $(*)$ for X and ω'. The number of such relevant wake-up schedules is not greater than $(m+1)^{|X|} \leq (2m)^k$, so the probability that $\mathcal{S}$ violates $(*)$ for X is at most $(2m)^k\beta^{m/k}$.

Now, to estimate the probability that $\mathcal{S}$ is not a (n,k,m)-synchronizer, we can multiply the above probability by the number of ways in which X can be chosen, which is at most $\sum_{j=1}^{k}\binom{n}{j} \leq \sum_{j=1}^{k} n^j \leq n^{2k}$. Thus

$$
\begin{aligned}
\Pr[\,\mathcal{S} \text{ is not a } (n,k,m)\text{-synchronizer}\,] &\leq n^{2k}(2m)^k\beta^{m/k} \\
&\leq n^{2k}(2Ck^2\log n)^k\beta^{Ck\log n} \\
&\leq \left[n^{6+\log C + C\log\beta}\right]^k \\
&< 1,
\end{aligned}
$$

as long as $C \geq 31$. Since this probability is less than 1, there exists a (n,k,m)-synchronizer $\mathcal{S}$ with $m = Ck^2\log n$. $\Box$

4 A Deterministic Wake-Up Protocol

In this section we use the probabilistic method to show the existence of a deterministic protocol that completes the wake-up task in n-vertex directed graphs in time $O(n^{5/3}\log n)$. We proceed in four steps:

(1) We define a random protocol $\mathcal{P}$ and a class of graphs that we call *path graphs*. We also define a slightly different communication model for those graphs that we call the *restricted model*.

(2) We show that, with probability exponentially close to 1, $\mathcal{P}$ will complete the wake-up task in time $O(n^{5/3} \log n)$, according to the restricted model, for an arbitrary n-vertex path graph and a wake-up schedule.

(3) Using the bound on the failure probability in (2), we show that there exists a *deterministic* protocol $\mathcal{D}$ for path graphs with running time $O(n^{5/3} \log n)$, in the restricted model.

(4) Finally, we prove that (3) implies that our protocol $\mathcal{D}$ works in fact also in arbitrary directed graphs, with running time $O(n^{5/3} \log n)$.

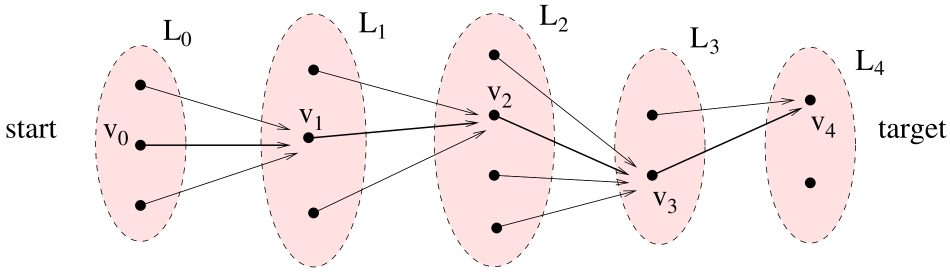
We point out here that applying the probabilistic method to path graphs, rather than directly to arbitrary graphs, is crucial, since the number of path graphs is substantially smaller than the number of arbitrary directed graphs – roughly n^n instead of $2^{n(n-1)}$.

Random protocol. We define a *random protocol* $\mathcal{P}$ as follows. Let $\mathcal{S} = \{\mathcal{S}^x\}_{x\in V}$ be a (n,k,m)-synchronizer with $k = n^{1/3}$ and $m = Cn^{2/3} \log n$, where $C = 31$. The existence of $\mathcal{S}$ has been established in Lemma 1. For each $x \in V$, we define a random 0-1 sequence $\mathcal{P}^x$ as

$$\mathcal{P}^x \quad = \quad \mathcal{S}^x\mathcal{U}^x,$$

where $\mathcal{U}^x$ is a random 0-1 sequence of length $O(n^{5/3} \log n)$, in which each bit is chosen, independently, to be 1 with probability $1/n$ and 0 with probability $1 - 1/n$. (The exact length of $\mathcal{U}^x$ will be determined later in this section.) Then $\mathcal{P} = \{\mathcal{P}^x\}_{x\in V}$ is a random protocol, where the 1's in $\mathcal{P}^x$ indicate the times when node x transmits the wake-up signal. We refer to the sequence $\mathcal{S}^x$ as the *S-stage* and $\mathcal{U}^x$ as the *U-stage* of $\mathcal{P}^x$.

Path graphs and restricted executions. A directed graph H is called a *path graph* if it has the following structure: The nodes of H can be partitioned into sets L_i, $i = 0, \dots, D$, called *layers*, each with a distinguished node $v_i \in L_i$. The edges of H are of the form (v, v_{i+1}), where $0 \le i < D$ and $v \in L_i$. We refer to D as the *depth* of H. The path $\pi = v_0, v_1, \dots, v_D$ is called the *main path* of H. Node v_0 is the *start node* and v_D is the *target node* (see the figure below). If H is a path graph and μ is a wake-up schedule, we refer to the pair $\langle H, \mu\rangle$ as a *path wake-up network*.



In path wake-up networks, we only consider wake-up schedules μ with v_0 as the start node, that is $\mu_{v_0} = 0$ and $\mu_v \ge 0$ for $v \ne v_0$. The wake-up task is also different: our objective is to wake-up only the target node v_D, not necessarily all nodes.

We will consider a *restricted communication model* for path graphs that differs slightly from the standard model described in Section 2. Let H be a path graph and μ a wake-up schedule. Assume that the sets of active nodes are already defined for times $1, \dots, t$ (as in a standard radio wake-up execution, if a node becomes active, it remains active through the end of the execution). We denote by h_t the greatest index i of an active layer L_i (that is, a layer with at least one active node) at time t. We refer to L_{h_t} as the *head layer* at time t. A non-active node v becomes activated in step t if one of the following cases apply:

(Act1) $\mu_v = t$, or

(Act2) $v = v_{h_t+1}$ and exactly one node in L_{h_t} transmits in step t.

Thus the difference between the standard and restricted models in path graphs is that the nodes v_i, for $i \leq h_t$, cannot be activated by their in-neighbors in H even if exactly one of them transmits. (Nodes v_i, for $i > h_t + 1$ cannot be activated by their in-neighbors in either model, by the choice of h_t.) Given a protocol $\mathcal{P}$, we refer to the execution of $\mathcal{P}$ on $\langle H, \mu\rangle$ under the restricted model as a *restricted execution*. By $\mathit{acttime}^*_u(\mathcal{D}, H, \mu)$ and $\mathit{ActSet}^*_t(\mathcal{D}, H, \mu)$, respectively, we denote the activation time step of u and the set of active notes at time t in the restricted execution of $\mathcal{D}$ on $\langle H, \mu\rangle$ (with the asterisk indicating a restricted execution.)

Recall that $m = Cn^{2/3}\log n$ is the length of our synchronizer, where $C = 31$ is the constant from Lemma 1.

Lemma 2 *Let $B \geq 15$ be an integer constant. Consider the restricted execution of protocol $\mathcal{P}$ on a path wake-up network $\langle H, \mu\rangle$ with vertices from V. The probability that $\mathcal{P}$ does not activate the target node of H in time $T = (B + C)n^{5/3}\log n$ is at most $n^{-\frac{1}{16}Bn}$.*

Proof: Let $L_0, \ldots, L_D$ be the layers of H and $\pi = v_0, \ldots, v_D$ be the main path. The activation time step of a layer L_i is a random variable that is equal t if either $t = \mu_{L_i}$, or if $h_t = i - 1$ and v_i receives the wake-up signal from L_{i-1} in time step t, whichever comes first.

Whenever the head index h_t increases, we call it an *advance*. Recall that v_0 is assumed to wake up at time step 0. We want to bound the probability of failure, i.e., that v_D is still not woken-up at time $T = (B + C)n^{5/3}\log n$.

Given some fixed execution of $\mathcal{P}$ (that is, with fixed random choices), we group all time steps t in this execution into two categories:

S-steps: time steps t when at least one node in a head layer L_{h_t} executes its S-stage, and

U-steps: time steps t when all active nodes in L_{h_t} execute their U-stage.

Suppose that layer L_i was activated in this execution at time step t'. If the number of active nodes in L_i at time $t' + m$ is at most $n^{1/3}$ then we say that L_i is *fast*. Otherwise, we call L_i *slow*. (We emphasize that the notions of fast and slow layers are relative to a particular execution of $\mathcal{P}$; the same layer can be slow or fast, depending on the outcomes of random choices before t'.)

We now make two important observations. First, if L_i is fast then, since $\mathcal{S}$ is a (n, k, m)-synchronizer with $k = n^{1/3}$, an advance to L_{i+1} will occur during the time period $[t' + 1, t' + m]$ with probability 1. The second observation, following from the previous one, is that U-steps occur only when the head layer is slow, and thus has size at least $n^{1/3}$.

Now we look at the execution of $\mathcal{P}$ on $\langle H, \mu\rangle$ globally. The total number of S-steps is at most $nm = Cn^{5/3}\log n$. During these S-steps, $\mathcal{P}$ will complete advances from all fast layers in its execution with probability 1.

Since each slow layer has size at least $n^{1/3}$, the number of slow layers in any execution cannot exceed $n^{2/3}$. The total number of U-steps is at least $M \stackrel{\text{def}}{=} T - Cn^{5/3}\log n = Bn^{5/3}\log n$. Let X_i be a random variable such that $X_i = 1$ if there is an advance in the ith U-step, and 0 otherwise. Thus, it is sufficient to prove that

$$\Pr\Big[\sum_{i=1}^{M} X_i < n^{2/3}\Big] \quad\le\quad n^{-\frac{1}{16}Bn}. \tag{1}$$

Consider one U-step, say the ith one, for some fixed execution history Ψ (that is, all random choices up to this point). At this step, the head layer has $x \ge n^{1/3}$ active nodes and all these nodes transmit with probability $1/n$. The probability of an advance is the probability that exactly one of them transmits, so

$$\Pr[\, X_i = 1 \mid \Psi \,] \;=\; x \cdot \frac{1}{n} \cdot \left(1 - \frac{1}{n}\right)^{x-1} \;\ge\; \tfrac{1}{4} n^{-2/3} \;\stackrel{\text{def}}{=}\; p. \tag{2}$$

The last inequality holds because it is true for $x = n^{1/3}$ and $x(1-1/n)^{x-1}$ is a non-decreasing function for $x < n$. (Recall that $n \ge 2$.) Therefore $\Pr[\, X_i = 1\,] \ge p$.

The idea behind the proof of (1) is this: we can think of the X_i's as a sequence of Bernoulli trials in which the probability of a success (an advance) is p. Then the probability that in this process the number of advances in M U-steps is less than $n^{2/3}$ can be bounded by $n^{-\Omega(n)}$ using an appropriate tail inequality.

The flaw in the above argument is that the advance events (variables X_i) are not fully independent, and thus this is not truly a Bernoulli process. To make the argument more rigorous, we introduce independent random variables $Y_i \in \{0,1\}$, where $\Pr[\, Y_i = 1\,] = p$. For these variables, the Chernoff bound (see [21], for example), states that $\Pr\Big[\sum_{i=1}^{M} Y_i < K\Big] \le \exp\left[-\frac{1}{2}Mp(1 - K/(Mp))^2\right]$, for any $0 \le K < Mp$. For $K = n^{2/3}$ and $B \ge 15$ we have

$$\left(1 - \frac{n^{2/3}}{Mp}\right)^2 \;=\; \left(1 - \frac{4}{Bn^{1/3}\log n}\right)^2 \;\ge\; \left(1 - \frac{4}{B}\right)^2 \ge \tfrac{1}{2}\,.$$

Therefore

$$\begin{aligned}
\Pr\Big[\sum_{i=1}^{M} Y_i < n^{2/3}\Big] &\le e^{-\frac{1}{2}Mp(1-n^{2/3}/(Mp))^2} \\
&\le e^{-\frac{1}{16}Bn\log n} \\
&\le n^{-\frac{1}{16}Bn}.
\end{aligned}$$

To complete the proof it is now sufficient to show that

$$\Pr\Big[\sum_{i=1}^{M} X_i < K\Big] \quad\le\quad \Pr\Big[\sum_{i=1}^{M} Y_i < K\Big]. \tag{3}$$

for each K. To show (3), we introduce new random variables $Z_i \in \{0,1\}$, $i = 1,\ldots,M$, where, for each history Ψ (the random choices up to before the ith U-step), we define

$$\begin{aligned}
\Pr[\, Z_i = 1 \mid \Psi \;\&\; Y_i = 1\,] &= 0, \\
\Pr[\, Z_i = 1 \mid \Psi \;\&\; Y_i = 0\,] &= \frac{\Pr[\, X_i = 1 | \Psi\,] - p}{1-p}.
\end{aligned}$$

Then processes $\{X_i\}$ and $\{Y_i + Z_i\}$ are stochastically equivalent. More precisely, for $i = 1,\ldots,M$, and any history Ψ,

$$\begin{aligned}
\Pr[\, Y_i + Z_i = 1 \mid \Psi\,] &= \Pr[\, Y_i = 1 \,\&\, Z_i = 0 \mid \Psi\,] + \Pr[\, Y_i = 0 \,\&\, Z_i = 1 \mid \Psi\,] \\
&= p + (1-p) \cdot \frac{\Pr[\, X_i = 1 \mid \Psi\,] - p}{1-p} \\
&= \Pr[\, X_i = 1 \mid \Psi\,]
\end{aligned}$$

and thus $\Pr[\, Y_i + Z_i = 1\,] = \Pr[\, X_i = 1\,]$ as well. Therefore

$$\Pr\Big[\, \textstyle\sum_{i=1}^{M} X_i < K \,\Big] \;=\; \Pr\Big[\, \textstyle\sum_{i=1}^{M} (Y_i + Z_i) < K \,\Big] \le \Pr\Big[\, \textstyle\sum_{i=1}^{M} Y_i < K \,\Big] \;,$$

completing the proof of (3). This also completes the proof of (1) and the whole lemma. □

Lemma 3 *There exists a deterministic protocol $\mathcal{D}$ that, under the restricted model, for any path wake-up network $\langle H, \mu\rangle$ with at most n nodes, activates the target node of H in time $O(n^{5/3}\log n)$.*

Proof: Let $T = (B + C)n^{5/3} \log n$ be the running time from Lemma 2, where $C = 31$ and $B \geq 15$. Each path graph has out-degree 1, so the number of (labelled) n-vertex path graphs is at most n^n. We only need to be concerned with wake-up schedules μ that satisfy $0 \leq \mu_v \leq T + 1 \leq 2T$, for all v, and $\mu_{v_0} = 0$ (where v_0 is the start node.) There are at most $(2T)^n$ such wake-up schedules. Thus, by Lemma 2, the probability that $\mathcal{P}$ does not complete the wake-up during a restricted execution on some n-vertex path graph and some wake-up schedule is at most

$$\begin{aligned}
n^{-\frac{1}{16}Bn} \cdot n^n (2T)^n \;&\leq\; \Big[n^{-\frac{1}{16}B} \cdot n \cdot 2(B+C)n^{5/3} \log n \Big]^n \\
&\leq\; \Big[n^{-\frac{1}{16}B + \log(B+C) + 5} \Big]^n \\
&<\; 1,
\end{aligned}$$

as long as $B \geq 15$ and $-\frac{1}{16}B + \log(B + C) + 5 < 0$. Any $B \geq 207$ satisfies these conditions. This implies that there exists a deterministic protocol $\mathcal{D}$ that, for any n-vertex path graph and any wake-up schedule, completes the wake-up task (under the restricted model) in time $O(n^{5/3} \log n)$. □

Theorem 1 *There exists a deterministic protocol that completes the wake-up process in each n-vertex strongly-connected directed graph in time $O(n^{5/3}\log n)$.*

Proof: We prove that protocol $\mathcal{D}$ from Lemma 3 satisfies the theorem. Let $T = O(n^{5/3} \log n)$ be the running time of protocol $\mathcal{D}$ from Lemma 3, in the restricted model, on any path wake-up network $\langle H, \mu\rangle$, where the vertex set of H is a subset of V. We show that $\mathcal{D}$ completes the wake-up in *any* graph with vertex set V under any wake-up schedule in time $T = O(n^{5/3} \log n)$.

Fix a wake-up network $\langle G, \omega\rangle$, and denote by v_0 the start node, where $\omega_{v_0} = 0$ and $\omega_u \geq 0$ for all $u \in G$. Let v be some arbitrary but fixed target node $v \in G$, and let $\pi = v_0, v_1, \ldots, v_d = v$ be the shortest path from v_0 to v. We define a subgraph H_v of G as follows: The vertices of H_v are all nodes $v_0, \ldots, v_d$, and all in-neighbors of nodes $v_1, \ldots, v_d$. The edges of H_v are all edges (u, v_i) in G such that $u \in H_v$ and $(u, v_j) \notin G$ for $j > i$. Clearly, H_v is a path graph with main path $\pi = v_0, v_1, \ldots, v_d = v$. As usual, by $L_0, \ldots, L_d$ we denote the layers of H_v.

Define a wake-up schedule μ for nodes $u \in H_v$ by $\mu_u = \mathit{acttime}_u(\mathcal{D}, G, \omega)$, the activation time of u in G under protocol $\mathcal{D}$ and wake-up schedule ω. We compare two executions of $\mathcal{D}$:

(i) the execution of $\mathcal{D}$ on G with wake-up schedule ω, and

(ii) the restricted execution of $\mathcal{D}$ on H_v with wake-up schedule μ.

The difficulty we need to address is that the nodes v_i may have different neighborhoods in G and in H_v. All neighbors of v_i in H_v are in L_{i-1}, while in G, v_i may also have neighbors in layers $L_i, \ldots, L_d$. Nevertheless, we claim that the following invariant holds at each time $t = 0, 1, \ldots, T$:

$$\text{for all } u \in H_v : \quad u \in \mathit{ActSet}_t(\mathcal{D}, G, \omega) \text{ iff } u \in \mathit{ActSet}^*_t(\mathcal{D}, H_v, \mu). \qquad (4)$$

If (4) holds, then the activation times of each node (in particular, the target node) in both executions are the same. Since v was chosen arbitrarily, the theorem follows from Lemma 3. Thus, to complete the proof of the theorem, it is sufficient to prove (4).

The proof of invariant (4) is by induction on times $t = 0, \ldots, T$. For $t = 0$, (4) holds by the very definition of μ. Suppose that (4) is satisfied up to time t. We show that it is also satisfied at time $t + 1$.

Note first, that for each node $u \in H_v - \pi$, its activation times in both executions are the same, $\mathit{acttime}^*_u(\mathcal{D}, H_v, \mu) = \mathit{acttime}_u(\mathcal{D}, G, \omega) = \mu_u$. Thus (4) holds for nodes $u \in H - \pi$. The implication ($\Rightarrow$) is also immediate from the definition of μ. It is thus only sufficient to prove the implication ($\Leftarrow$) for $u \in \pi$.

It follows directly from invariant (4) applied to times $1, \ldots, t$ that the activation time steps (up to step $t - 1$) of active nodes in H_v are the same in both executions of $\mathcal{D}$, the execution on $\langle G, \omega\rangle$ and the restricted execution on $\langle H_v, \mu\rangle$. Consequently, any $u \in \mathit{ActSet}^*_t(\mathcal{D}, H_v, \mu)$ performs the same action in step t in both executions. (This is because $\mathcal{D}^u$ does not depend on whether u woke up spontaneously or was activated by another node.)

As before, let h_t be the head layer of $\langle H_v, \mu\rangle$ at time t. For any node v_i, where $i \neq h_t + 1$, it follows directly from the definition of H_v and the rules of the restricted model that v_i is activated in step t of the execution of $\mathcal{D}$ on $\langle G, \omega\rangle$ iff it is activated in step t of the restricted execution of $\mathcal{D}$ on $\langle H_v, \mu\rangle$.

Consider v_i, for $i = h_t + 1$, and suppose that v_i is activated in H_v at time step t. If $\mu_{v_i} = t$, then v_i is activated in G as well, by the definition of μ. It is sufficient to prove that $\mu_{v_i} > t$ is not possible. Toward contradiction, suppose $\mu_{v_i} > t$, that is, v_i was activated at step t using rule (Act2) by a wake-up signal from some node in L_{i-1}. Then, by the definitions of H_v and h_t, and also by invariant (4), we have $N_G(v_i) \cap \mathit{ActSet}_t(\mathcal{D}, G, \omega) = N_{H_v}(v_i) \cap \mathit{ActSet}^*_t(\mathcal{D}, H_v, \mu)$, and all nodes in this set are in the same state in both executions. It follows that v_i would have received the same message in G in step t as well – contradiction with the assumption $\mu_i > t$. This implies that (4) holds at time $t + 1$, completing the proof. $\square$

We emphasize that the above proof is not valid for arbitrary protocols, as we strongly use the fact that in our protocol $\mathcal{D}$ the active nodes do not use any information about how they have been activated (spontaneously or by a wake-up signal).

Recall that in this section we made a simplifying assumption that the node labels are consecutive integers $1, 2, \ldots, n$. Theorem 1 remains valid, however, if the node labels are chosen from a range $\ell_{\text{min}}, \ldots, \ell_{\text{max}}$, where $\ell_{\text{max}} - \ell_{\text{min}} \le An$, for some constant A, for in this case we can simply use a protocol for auxiliary labels $0, 1, \ldots, An$ (that is, a node with label ℓ uses label $\ell - \ell_{\text{min}}$ to identify itself). This will not affect the asymptotic performance of the protocol.

5 A Randomized Wake-Up Protocol

In [17], the authors presented a randomized wake-up protocol for complete graphs. We now give a randomized Monte-Carlo wake-up protocol $\mathcal{R}$ for general multi-hop radio networks. $\mathcal{R}$ receives on input a parameter $\epsilon > 0$, which is the desired bound on the probability of failure.

Protocol $\mathcal{R}$. The pseudo-code of the protocol for an arbitrary node v is:

```
for s ← 0, 1, ..., log n − 1 do
    repeat b = 22 log(n/ε) times          // stage s starts
        TRANSMIT with probability 2^s/n
```

We start with the following technical lemma from [17]:

Lemma 4 [17] *Suppose we have k numbers, $p_1, p_2, \ldots, p_k \in (0, \frac{1}{2}]$, such that $\frac{1}{2} \leq \sum_{i=1}^{k} p_i \leq 2$. Then*

$$\sum_{i=1}^{k} p_i \prod_{\substack{j=1 \\ j \neq i}}^{k} (1 - p_j) \quad \geq \quad \tfrac{1}{32}.$$

Theorem 2 *Let $\epsilon > 0$ and let $\langle G, \omega \rangle$ be any wake-up network with n nodes and diameter D. With probability at least $1 - \epsilon$, $\mathcal{R}$ completes wake-up in $\langle G, \omega \rangle$ in time $O(D \log n \log(n/\epsilon))$.*

Proof: We divide the execution of $\mathcal{R}^v$ into $\log n$ stages, where stage s is simply the s-th iteration of the **for** loop. Recall that $b = 22 \log(n/\epsilon)$ is the number of iterations of the inner **repeat** loop. By $B = b \log n$ we denote the total execution time of $\mathcal{R}$ (for a single node).

Fix a wake-up network $\langle G, \omega \rangle$, and consider an execution of $\mathcal{R}$ on $\langle G, \omega \rangle$. Let v_0 be the start node (the one that wakes up first). For any node v, let $t_v = \mathit{acttime}_v$ and $t_v^* = \min_{u \in N_v} \mathit{acttime}_u$. Thus $t_v - t_v^*$ is the time it takes for v to become activated since the time when the first of its in-neighbors gets activated.

Call a node v an *obstruction* during an execution of $\mathcal{R}$ if $t_v^* < \infty$ but $t_v > t_v^* + B$. If there is no obstruction, then, clearly, the execution of $\mathcal{R}$ completes after at most $DB = O(D \log n \log(n/\epsilon))$ steps. Thus it remains to prove that the probability that there exists an obstruction during an execution of $\mathcal{R}$ is at most ϵ.

Consider some fixed node v, and fix some value of t_v^*. To simplify notation, we can in fact assume that $t_v^* = 0$. If $\mu_v < B$, then v is not an obstruction with probability 1, and we are done, so we can assume that $\omega_v \geq B$. We now need to estimate the probability that v will receive a wake-up signal from N_v in at most B steps.

For $u \in N_v$, let $p_{u,t}$ be the probability that u transmits at time step t (that is, $p_{u,t} = 2^s/n$ if u is in stage s at time t), and let $\pi_t = \sum_{u \in N_v} p_{u,t}$. We have $\pi_0 = 0$, $\pi_{B-b+1} \geq \frac{1}{2}$, and π_t is non-decreasing. By the definition of $\mathcal{R}$, for any $t = 1, ..., B - b$, we have $\pi_{t+b} \leq 2\pi_t + 1$, since in b steps the active nodes will double their probability of transmission, and each other node contributes at most $1/n$ to π_{t+b}. Choose the largest t' for which $\pi_{t'-1} < \frac{1}{2}$. Then $\pi_{t'} \geq \frac{1}{2}$ and $\pi_{t'+b-1} \leq 2\pi_{t'-1} + 1 \leq 2$. We conclude that

$$\tfrac{1}{2} \;\leq\; \pi_t \;\leq\; 2 \quad \text{for } t = t', t'+1, \ldots, t'+b-1. \tag{5}$$

Thus, by Lemma 4, for each time $t = t', t'+1, ..., t'+b-1$, the probability that v receives a wake-up signal from N_v at time t is

$$\sum_{u \in N_v} p_{u,t} \prod_{z \in N_v \setminus \{u\}} (1 - p_{z,t}) \quad \geq \quad \tfrac{1}{32}.$$

Then the probability that v will not receive a wake-up signal in steps $1, 2, \ldots, B$ is at most $(\frac{31}{32})^b$. We conclude that the probability that v is an obstruction is bounded by $(\frac{31}{32})^b$ as well. Therefore the probability that there exists an obstruction node during the execution of $\mathcal{R}$ is at most

$$n(\tfrac{31}{32})^b \;=\; n(\tfrac{31}{32})^{22 \log(n/\epsilon)} \;\leq\; n(\tfrac{1}{2})^{\log(n/\epsilon)} \;\leq\; \epsilon,$$

completing the proof. □

We can modify $\mathcal{R}$ to obtain a Las Vegas protocol with low expected running time. Let $\mathcal{R}_0$ be $\mathcal{R}$ with $\epsilon = n^{-3}$, so that the running time of $\mathcal{R}_0$ is $T_0 = O(D \log^2 n)$. Let $T = O(n^{5/3} \log n)$ be the running time of the deterministic protocol $\mathcal{D}$ from the previous section. In this Las Vegas protocol $\mathcal{R}'$, each node executes the following steps: (1) First, run $\mathcal{R}_0$, and then suspend itself for T_0 steps, (2) repeat n times: run $\mathcal{D}$ and then suspend itself for time T. At least one run of $\mathcal{D}$ will complete without interference from nodes that wake up spontaneously and execute $\mathcal{R}$. Thus $\mathcal{R}'$ will wake-up all nodes in time T_0 with probability at least $1 - \epsilon$, and in time $O(n^{8/3} \log n)$ with probability 1. By the choice of ϵ, its expected running time will be $O(D \log^2 n)$.

Theorem 3 *Let $\langle G, \omega \rangle$ be any wake-up network with n nodes. Protocol $\mathcal{R}'$ described above completes wake-up in $\langle G, \omega \rangle$ in expected time $O(D \log^2 n)$.*

6 Leader Election and Clock Synchronization

In the leader election problem, we want to designate one node as *the leader*, and to announce its identity to all nodes in the network. In the clock synchronization problem, upon the completion of the protocol, all nodes must agree on a common global time. In this section we show that any wake-up protocol $\mathcal{W}$ (deterministic or randomized) can be transformed into a leader election protocol or a clock synchronization protocol with only a logarithmic overhead.

Let $\mathcal{W}$ be a wake-up protocol. For any node v, let $\mathcal{W}^v$ be the instance of $\mathcal{W}$ executed by node v. (Formally, the execution of $\mathcal{W}$ depends on the label, but, for simplicity of notation, we write $\mathcal{W}^v$ instead of $\mathcal{W}^{\ell_v}$.) We can assume, without loss of generality, that in protocol $\mathcal{W}$ each node transmits only for at most w time steps since activation. Fix n, and by $w = w(n)$ denote the running time of $\mathcal{W}$, if all node labels are drawn from the range $\ell_{\text{min}}, \ldots, \ell_{\text{max}}$, where $\ell_{\text{max}} - \ell_{\text{min}} \leq An$, for some constant A. Note that our wake-up protocols described earlier in the paper satisfy this requirement.

Recall that, without loss of generality, we assume that $\ell_{\text{min}} = 0$. We can then think of the labels as binary strings of length $\log(An)$, and by $\ell_v[i, .., j]$ we denote the string consisting of the bits of ℓ_v on positions $i, i+1, ..., j$, counting from left to right (that is, highest bits first).

6.1 Leader Election

The leader election protocol consists of two stages: *wake-up stage* and *election stage*. We first describe both stages informally, and then give a pseudocode for the complete protocol.

Wake-up stage. In this stage all nodes become activated. Each node v, once it wakes-up, starts executing its wake-up protocol $\mathcal{W}^v$. After completing this protocol, v suspends itself for w time steps. The purpose of this suspension period is to ensure that the wake-up stage does not overlap with the election stage that follows.

Election stage. In this stage, all nodes in the network select the leader to be the node v^* that possesses the smallest label. The selection process is performed in $\log(An)$ rounds, where in round r, for $r = 0, 1, \ldots, \log(An) - 1$, the ith highest bit of ℓ_{v^*} is announced. At the beginning of round r all nodes know the top r bits of ℓ_{v^*}. Let *min_label* $= \ell_{v^*}[0, ..., r-1]$ be the string consisting of these bits. Nodes v for which $\ell_v[0, ..., r-1] =$ *min_label* are potential candidates for the leader. If such a candidate node v also has a 0 on the rth bit, it informs other nodes about its existence by initiating a wake-up process. If no wake-up process is initiated in round r, by the end of the round all nodes conclude that the ith most significant bit in ℓ_{v^*} is 1.

The difficulty that we need to overcome is that, due to lack of synchronization, the division into consecutive rounds must be based on the local clocks. It means that very likely there will be some time overlaps (of size at most w - the maximum offset between the activation times of any two nodes) between neighboring rounds of different nodes. In order to avoid simultaneous transmissions of signals belonging to two different rounds, we will pad each round with two wait periods of length w, one at the beginning and one at the end. More specifically, every round is split into four sub-rounds, each of length w. If a node v is a candidate for the leader and initiates the transmission process by itself, this is always done in the beginning of the second sub-round. On the other hand, v can be activated to start its wake-up process by another node u at any time during the first, second or third sub-round.

Leader election protocol $\mathcal{E}$. Below, we give a pseudocode for the protocol $\mathcal{E}^v$ executed by a node v. We use small capitals font, START, EXECUTE, etc, for commands related to timing. Variable *clock* measures the local time of a node.

```
SLEEPUNTIL wake_up_signal (spontaneous or external)
START clock ← 0
EXECUTE W^v (w steps)
WAITUNTIL clock = 2w
min_label ← [ ]                       // empty string
for r ← 0, 1, ..., log(An) − 1 do
    bit ← 1                           // round r starts, local time is clock = (4r + 2)w
    if ℓ_v[0, ..., r] = min_label ∘ 0 then
        bit ← 0
        WAITUNTIL wake_up_signal or clock = (4r + 3)w
    else
        WAITUNTIL wake_up_signal or clock = (4r + 5)w
        if wake_up_signal then bit ← 0
    if bit = 0 then EXECUTE W^v (w steps)
    min_label ← min_label ∘ bit
    WAITUNTIL clock = (4r + 6)w
```

One detail that we should clarify is that, when we simulate $\mathcal{W}$ in protocol $\mathcal{E}$ during the election stage, all nodes “pretend” to be asleep during the wait periods, and nodes that initiate wake-up “pretend” to be woken-up spontaneously. During the simulation of a wake-up process, each node uses an auxiliary clock to simulate a local clock used by $\mathcal{W}$.

To justify correctness, it is sufficient to show that there will be no interference between wake-up processes from different rounds. The crucial property used in the argument is that the (global) activation times (in the wake-up phase) of any two nodes differ by at most w. This implies, in particular, that two rounds of different nodes can overlap only if they are consecutive and if so, they can overlap on at most w time steps. More specifically, the only possible overlap is when the fourth sub-round of a node in round r overlaps the first sub-round of another node that is in round $r + 1$. (In our argument, to avoid dealing separately with the wake-up phase, we can treat the wake-up phase as round -1 of the election phase.)

If a wake-up process in round r is initiated at a (global) time t, then all nodes will be activate by time $t + w$, and they will have completed their transmissions at time $t + 2w$. Since wake-ups are initiated at the beginning of the second sub-round, and the local times are shifted by at most w, it follows that, in each round r, each node will be activated no later than at the end of its third sub-round, and thus complete its transmissions by the end of its round r.

Fix a node v that is first to execute its transmission from round $r + 1$. This means that v initiates a wake-up process, and it starts transmitting in the first step of its second sub-round. But, again, using the fact that the offset between different rounds is at most w, by that time, all other nodes completed their transmissions from round r.

Thus we obtain:

Theorem 4 (a) *Suppose that $\mathcal{W}$ is a deterministic wake-up protocol with running time $w(n)$. Then $\mathcal{W}$ can be converted into a leader election protocol $\mathcal{E}$ with running time $O(w(n) \log n)$.*

(b) *Suppose that $\mathcal{W}$ is a randomized wake-up protocol that completes the wake-up task in time $w_\epsilon(n)$ with probability at least $1 - \epsilon$. Then $\mathcal{W}$ can be converted into a randomized leader election protocol $\mathcal{E}$ that completes the election task in time $O(w_\epsilon(n) \log n)$ with probability at least $1 - O(\epsilon \log n)$.*

Note that, in the theorem above, we do not make any assumptions on the wake-up protocol $\mathcal{W}$.

6.2 Clock Synchronization

It is not difficult to extend the leader election protocol $\mathcal{E}$ from the previous sub-section to perform clock synchronization. We first run $\mathcal{E}$. Once a leader v^* has been elected, v^* waits for w steps, and then broadcasts its local time to all other nodes using some broadcasting protocol $\mathcal{B}$ (see, for example [9]). When the nodes perform $\mathcal{B}$, they increment their global time counter at each step, to make sure that the correct value is transmitted. (A similar idea appeared in [23].) As explained in the introduction, broadcasting can be reduced to wake-up, so we can assume that the running time of $\mathcal{B}$ is at most $w(n)$.

Thus we obtain:

Theorem 5 (a) *Suppose that $\mathcal{W}$ is a deterministic wake-up protocol with running time $w(n)$. Then $\mathcal{W}$ can be converted into a deterministic clock synchronization protocol $\mathcal{C}$ with running time $O(w(n) \log n)$.*

(b) *Suppose that $\mathcal{W}$ is a randomized wake-up protocol that completes the wake-up task in time $w_\epsilon(n)$ with probability at least $1-\epsilon$. Then $\mathcal{W}$ can be converted into a randomized clock synchronization protocol $\mathcal{C}$ that completes the election task in time $O(w_\epsilon(n) \log n)$ with probability at least $1 - O(\epsilon \log n)$.*

6.3 New Protocols

Recall that our deterministic wake-up protocol $\mathcal{D}$ runs in time $O(n^{5/3} \log n)$, our Monte Carlo protocol $\mathcal{R}$ in time $O(D \log n \log(n/\epsilon))$ and with failure probability ϵ, and our Las Vegas protocol $\mathcal{R}'$ runs in expected time $O(D \log^2 n)$. For broadcasting, we can use protocols from [9] or [18, 12], and any randomized broadcasting protocol in [3, 19, 12], or, alternatively, we can adapt wake-up protocols from our paper. Summarizing the results in this section, we obtain the following corollary.

Corollary 1 *There exist protocols for leader election and clock synchronization with the following performance in n-node networks:*

(a) *A deterministic protocol with running time $O(n^{5/3} \log^2 n)$.*

(b) *A Monte Carlo protocol with running time $O(D \log n \log(n/\epsilon))$ and failure probability at most $O(\epsilon \log n)$.*

(c) *A Las Vegas protocol with expected running time $O(D \log^3 n)$.*

Results in Corollary 1(a) and (b) are straightforward from Theorems 1, 2, 4, 5. The randomized Las Vegas protocols in Corollary 1(c) are obtained in the similar way as algorithm Las Vegas for wake-up problem: by taking the corresponding Monte Carlo protocol with sufficiently small failure probability, say $\epsilon = n^{-3}$, followed by n repetitions of a deterministic protocol. These algorithms succeed in time $O(D \log^3 n)$ with probability at least $1 - \epsilon \log n$, and in time $O(n^{8/3} \log n)$ with probability 1, so theirs expected running time will be $O(D \log^3 n)$.

7 Final Comments

In this paper, we initiated the study of wake-up protocols in a general model of multi-hop radio networks where the topology of the network is unknown and there is no collision detection.

Since the lower bounds for the broadcasting (see [5, 11, 20]) apply to the wake-up problem, no randomized wake-up protocol can be faster than $\Omega(D \log(n/D))$. Thus our randomized protocol is within a poly-logarithmic factor of the optimum. However, in the deterministic case, there still remains a substantial gap between the lower bound of $\Omega(n \log n)$ and the upper bound $O(n^{5/3} \log n)$ achieved by our protocol $\mathcal{D}$.

Another direction for study would be to develop explicit and efficient constructions of small synchronizers.

References

[1] B. Awerbuch, Complexity of network synchronization, J. ACM 32(4) (1985), 804-823.

[2] B. Awerbuch, S. Kutten, Y. Mansour, B. Patt-Shamir and G. Varghese, Time optimal self-stabilizing synchronization, Proc. 25th ACM Symposium on Theory of Computing (STOC'1993), 652-661.

[3] R. Bar-Yehuda, O. Goldreich and A. Itai, On the time complexity of broadcast in radio networks: an exponential gap between determinism and randomization, Journal of Computer and System Sciences 45 (1992), 104-126.

[4] R. Bar-Yehuda, A. Israeli and A. Itai, Multiple communication in multihop radio networks, SIAM J. on Computing 22 (1993), 875-887.

[5] D. Bruschi and M. Del Pinto, Lower bounds for the broadcast problem in mobile radio networks, Distributed Computing 10 (1997), 129-135.

[6] I. Chlamtac and O. Weinstein, The wave expansion approach to broadcasting in multihop radio networks, IEEE Trans. on Communications 39 (1991), 426-433.

[7] B.S. Chlebus, L. Gasieniec, A. Gibbons, A. Pelc and W. Rytter, Deterministic broadcasting in unknown radio networks, Distributed Computing 15 (2002), 27-38.

[8] B.S. Chlebus, L. Gąsieniec, A. Östlin and J.M. Robson, Deterministic radio broadcasting, Proc. 27th Int. Coll. on Automata, Languages and Programming (ICALP'2000), LNCS 1853, 717-728.

[9] M. Chrobak, L. Gąsieniec and W. Rytter, Fast broadcasting and gossiping in radio networks, Journal of Algorithms 46 (2003), 1-20.

[10] A. Clementi, P. Crescenzi, A. Monti, P. Penna and R. Silvestri, On computing ad-hoc selective families, Proc. 5th International Workshop on Randomization and Approximation Techniques in Computer Science (RANDOM-APPROX'2001), 211-222.

[11] A. Clementi, A. Monti and R. Silvestri, Selective families, superimposed codes, and broadcasting on unknown radio networks, Proc. 12th Ann. ACM-SIAM Symposium on Discrete Algorithms (SODA'2001), 709-718.

[12] A. Czumaj and W. Rytter, Broadcasting algorithms in radio networks with unknown topology, Proc. 44th Symposium on Foundations of Computer Science (FOCS'2003), 492-501.

[13] M.J. Fischer, S. Moran, S. Rudich and G. Taubenfeld, The wakeup problem, SIAM J. Comput. 25 (1996), 1331-1332.

[14] L. Gąsieniec, A. Pelc and D. Peleg, The wakeup problem in synchronous broadcast systems, SIAM Journal on Discrete Mathematics 14 (2001), 207-222.

[15] P. Indyk, Explicit constructions of selectors and related combinatorial structures, with applications, Proc. 13th Ann. ACM-SIAM Symposium on Discrete Algorithms (SODA'2002), 697-704.

[16] T. Jurdzinski, M. Kutylowski and J. Zatopianski, Efficient algorithms for leader election in radio networks, Proc. 21st ACM Symposium on Principles of Distributed Computing (PODC'2002), 51-57.

[17] T. Jurdziński, G. Stachowiak, Probabilistic algorithms for the wakeup problem in single-hop radio networks, Proc. 13th Annual International Symposium on Algorithms and Computation (ISAAC'2002), 535-549.

[18] D. Kowalski and A. Pelc, Faster deterministic broadcasting in ad hoc radio networks, Proc. 20th Annual Symposium on Theoretical Aspects of Computer Science (STACS'2003), 109-120.

[19] D. Kowalski and A. Pelc, Broadcasting in undirected ad hoc radio networks, Proc. 22nd ACM Symposium on Principles of Distributed Computing (PODC'2003), 73-82.

[20] E. Kushilevitz and Y. Mansour, An $\Omega(D \log(N/D))$ lower bound for broadcast in radio networks, SIAM J. on Computing 27 (1998), 702-712.

[21] R. Motwani and P. Raghavan, Randomized algorithms, Cambridge University Press, 1995.

[22] N. Lynch, Distributed Algorithms, Morgan-Kauffman, 1997.

[23] D. Peleg, Deterministic radio broadcast with no topological knowledge, manuscript (2000).

[24] R. Zheng, J.C. Hou and L. Sha, Asynchronous wakeup for power management in ad hoc networks, Proc. 4th ACM Symposium on Mobile Ad Hoc Networking and Computing (MOBIHOC'2003), 35-45.