Faster Deterministic Communication in Radio Networks[*][†]

Artur Czumaj[1] and Peter Davies[2]

1 Department of Computer Science and Centre for Discrete Mathematics and its Applications, University of Warwick, Warwick, United Kingdom
A.Czumaj@warwick.ac.uk

2 Department of Computer Science and Centre for Discrete Mathematics and its Applications, University of Warwick, Warwick, United Kingdom
P.W.Davies@warwick.ac.uk

Abstract

In this paper we improve the deterministic complexity of two fundamental communication primitives in the classical model of ad-hoc radio networks with unknown topology: broadcasting and wake-up. We consider an unknown radio network, in which all nodes have no prior knowledge about network topology, and know only the size of the network n, the maximum in-degree of any node Δ, and the eccentricity of the network D.

For such networks, we first give an algorithm for wake-up, in both directed and undirected networks, based on the existence of small universal synchronizers. This algorithm runs in $O(\frac{\min\{n, D\Delta\}\log n \log \Delta}{\log\log \Delta})$ time, improving over the previous best $O(n \log^2 n)$-time result across all ranges of parameters, but particularly when maximum in-degree is small.

Next, we introduce a new combinatorial framework of block synchronizers and prove the existence of such objects of low size. Using this framework, we design a new deterministic algorithm for the fundamental problem of *broadcasting*, running in $O(n \log D \log\log \frac{D\Delta}{n})$ time. This is the fastest known algorithm for this problems, improving upon the $O(n \log n \log\log n)$-time algorithm of De Marco (2010) and the $O(n \log^2 D)$-time algorithm due to Czumaj and Rytter (2003), the previous fastest results for directed networks, and is the first to come within a log-logarithmic factor of the $\Omega(n \log D)$ lower bound due to Clementi et al. (2003).

Our results have also direct implications on the fastest *deterministic leader election* and *clock synchronization* algorithms in both directed and undirected radio networks, tasks which are commonly used as building blocks for more complex procedures.

1998 ACM Subject Classification C.2.1 Distributed Networks

Keywords and phrases Radio networks, Communication networks, Broadcasting, Wake-Up, Deterministic algorithms

Digital Object Identifier 10.4230/LIPIcs.ICALP.2016.139

1 Introduction

1.1 Model of communication networks

We consider the classical model of *directed ad-hoc radio networks* with *unknown structure*. A *radio network* is modeled by a *directed* network $\mathfrak{N} = (V, E)$, where the set of nodes

[*] Due to space limitations, some details are deferred to the full version, available at https://arxiv.org/abs/1506.00853.
[†] Research partially supported by the Centre for Discrete Mathematics and its Applications (DIMAP).

corresponds to the set of transmitter-receiver stations. The nodes of the network are assigned different identifiers (IDs), and throughout this paper we assume that all IDs are distinct numbers in $\{1, \dots, |V|\}$. A directed edge $(v, u) \in E$ means that node v can send a message directly to node u. To make propagation of information feasible, we assume that every node in V is reachable in $\mathfrak{N}$ from any other.

In accordance with the standard model of unknown (ad-hoc) radio networks (for more elaborate discussion about the model, see, e.g., [1, 2, 5, 9, 10, 12, 17, 19, 22]), we make the assumption that a node does not have any prior knowledge about the topology of the network, its in-degree and out-degree, or the set of its neighbors. We assume that the only knowledge of each node is the *size* of the network n, the *maximum in-degree* of any node Δ, and the *eccentricity* of the network D, which is the maximum distance from the source node to any node in $\mathfrak{N}$. For a discussion of these assumptions, see the full version of this paper.

Nodes operate in discrete, synchronous time steps, but we do not need to assume knowledge of a global clock. When we refer to the "running time" of an algorithm, we mean the number of time steps which elapse before completion (i.e., we are not concerned with the number of calculations nodes perform within time steps). In each time step a node can either *transmit* a message to all of its out-neighbors at once or can remain silent and *listen* to the messages from its in-neighbors. We do not make any restriction on the size of messages.

The distinguishing feature of radio networks is the interfering behavior of transmissions. In the most standard radio networks model, the *model without collision detection* (see, e.g., [1, 2, 10, 22]), which is studied in this paper, if a node v listens in a given round and precisely one of its in-neighbors transmits, then v receives the message. In all other cases v receives nothing; in particular, the lack of collision detection means that v is unable to distinguish between zero of its in-neighbors transmitting and more than one.

The model without collision detection describes the most restrictive interfering behavior of transmissions; also considered in the literature is a less restrictive variant, the model with collision detection, where a node listening in a given round can distinguish between zero of its in-neighbors transmitting and more than one (see, e.g., [12, 22]).

1.2 Communications primitives: broadcasting and wake-up

In this paper we consider two fundamental communications primitives, namely *broadcasting* and *wake-up*, and consider *deterministic protocols* for each of these tasks.

1.2.1 Broadcasting

Broadcasting is one of the most fundamental problems in communication networks and has been extensively studied for many decades (see, e.g., [22] and the references therein).

The premise of the broadcasting task is that one particular node, called the *source*, has a message which must become known to all other nodes. We assume that all other nodes start in a dormant state and do not participate until they are "woken up" by receiving the source message (this is referred to in some works as the "no spontaneous transmissions" rule). As a result, while the model does not assume knowledge of a global clock, we can make this assumption in practice, since the current time can be appended to the source message as it propagates, and therefore will be known be all active nodes. This is important since it allows us to synchronize node behavior into fixed-length *blocks*.

1.2.2 Wake-up

The *wake-up problem* (see, e.g., [15]) is a related fundamental communication problem that arises in networks where there is no designated "source" node, and no synchronized time-step at which all nodes begin communicating. The goal is for all nodes to become "active" by receiving some transmission. Rather than a single source node which begins active, we instead assume that some subset of nodes spontaneously become active at arbitrary time-steps. The task can be seen as a variant of broadcast, with possibly multiple sources, and without the ability to assume a global clock. This last point is important, and results in wake-up protocols being slower than those for broadcast, since nodes cannot co-ordinate their behavior.

1.3 Related work

As a fundamental communications primitive, the task of *broadcasting* has been extensively studied for various network models for many decades.

For the model studied in this paper, directed radio networks with unknown structure and without collision detection, the first sub-quadratic deterministic broadcasting algorithm was proposed by Chlebus et al. [5], who gave an $O(n^{11/6})$-time broadcasting algorithm. After several small improvements (cf. [6, 21]), Chrobak et al. [9] designed an almost optimal algorithm that completes the task in $O(n \log^2 n)$ time, the first to be only a poly-logarithmic factor away from linear dependency. Kowalski and Pelc [17] improved this bound to obtain an algorithm of complexity $O(n \log n \log D)$ and Czumaj and Rytter [11] gave a broadcasting algorithm running in time $O(n \log^2 D)$. Finally, De Marco [20] designed an algorithm that completes broadcasting in $O(n \log n \log \log n)$ time steps. Thus, in summary, the state of the art result for deterministic broadcasting in directed radio networks with unknown structure (without collision detection) is the complexity of $O(n \min\{\log n \log \log n, \log^2 D\})$ [11, 20]. The best known lower bound is $\Omega(n \log D)$ due to Clementi et al. [10].

Broadcasting has been also studied in various related models, including undirected networks, randomized broadcasting protocols, models with collision detection, and models in which the entire network structure is known. For example, if the underlying network is undirected, then Kowalski [16] gave a deterministic broadcasting algorithm running in time $O(n \log D)$. If spontaneous transmissions are allowed and a global clock available, then deterministic broadcast can be performed in $O(n)$ time in undirected networks [5]. Randomized broadcasting has been also extensively studied, and in a seminal paper, Bar-Yehuda et al. [2] designed an almost optimal broadcasting algorithm achieving the running time of $O((D + \log n) \cdot \log n)$. This bound has been later improved by Czumaj and Rytter [11], and independently Kowalski and Pelc [18], who gave optimal randomized broadcasting algorithms that complete the task in $O(D \log \frac{n}{D} + \log^2 n)$ time with high probability, matching a known lower bound from [19]. In the model with collision detection, an $O(D + \log^6 n)$-time randomized algorithm due to Ghaffari et al. [12] is the first to exploit collisions and surpass the algorithms (and lower bound) for broadcasting without collision detection.

For more details, see e.g., [22] and the references therein.

The *wake-up problem* (see, e.g., [15]) is a related communication problem that arises in networks where there is no designated "source" node, and no synchronized time-step at which all nodes begin communicating. Before any more complex communication can take place, we must first require all nodes to be "active," i.e., aware that they should be communicating. This is the goal of wake-up, and it is a fundamental starting point for most other tasks in this setting, for example leader election and clock synchronization [8].

The first sub-quadratic deterministic wake-up protocol was given in by Chrobak et al. [8], who introduced the concept of *radio synchronizers* to abstract the essence of the problem.

They give an $O(n^{5/3} \log n)$-time protocol. Since then, there have been two improvements in running time, both making use of the radio synchronizer machinery: firstly to $O(n^{3/2} \log n)$ [4], and then to $O(n \log^2 n)$ [3]. Unlike for the problem of broadcast, the fastest known protocol for directed networks is also the fastest for undirected networks. A recent survey of the current state of research on the wake-up problem is given in [15].

1.4 Our results

In this paper we present a *new construction of universal radio synchronizers* and introduce and analyze a *new concept of block synchronizers* to improve the deterministic complexity of two fundamental communication primitives in the model of ad-hoc radio networks with unknown topology: broadcasting and wake-up.

By applying the analysis of block synchronizers, we present a new deterministic broadcasting algorithm (**Algorithm 1**) in directed ad-hoc radio networks with unknown structure, without collision detection, that for any directed network $\mathfrak{N}$ with n nodes, with eccentricity D, and maximum in-degree Δ, completes broadcasting in $O(n \log D \log\log \frac{D\Delta}{n})$ time-steps. This result almost matches a lower bound of $\Omega(n \log D)$ due to Clementi et al. [10], and improves upon the previous fastest algorithms due to De Marco [20] and due to Czumaj and Rytter [11], which require $O(n \log n \log\log n)$ and $O(n \log^2 D)$ time-steps, respectively.

Our result reveals that a non-trivial speed-up can be achieved for a broad spectrum of network parameters. Since $\Delta \leq n$, our algorithm has the complexity at most $O(n \log D \log\log D)$. Therefore, in particular, it significantly improves the complexity of broadcasting for shallow networks, where $D \ll n^{O(1)}$. Furthermore, the dependency on Δ reduces the complexity even further for networks where the product $D\Delta$ is near linear in n including sparse networks which can appear in many natural scenarios.

Our broadcasting result has also direct implications on the fastest *deterministic leader election algorithm* in directed and undirected radio networks. It is known that leader election can be completed in $O(\log n)$ times broadcasting time (see, e.g., [9, 13]) (assuming the broadcast algorithm extends to multiple sources, which is the case here as long as we have a global clock), and so our result improves the bound to achieve a deterministic leader election algorithm running in $O(n \log n \log D \log\log \frac{D\Delta}{n})$ time. For undirected networks the best result is $O(n \log^{3/2} n \sqrt{\log\log n})$ time [7] (we note that the $O(n \log D)$ broadcast protocol of [16] cannot be used at a $\log n$ slowdown for leader election, since it relies on token traversal and does not extend to multiple sources). Our result therefore favorably compares for shallow networks (for small D) even in undirected networks.

We also present a deterministic algorithm (**Algorithm 2**) for the related task of wake-up. We show the existence of universal radio synchronizers of delay $g(k) = O(\frac{n \log n \log k}{\log\log k})$, and demonstrate that this yields a wake-up protocol taking time $O(\frac{\min\{n, D\Delta\} \log n \log \Delta}{\log\log \Delta})$. This improves over the previous best result, the $O(n \log^2 n)$-time protocol of [3]; the improvement is largest when Δ is small, but even when it is polynomial in n, our algorithm is a $\log\log n$-factor faster.

Our improved result for wake-up has direct applications to communication algorithms in networks that do not have access to a global clock, where wake-up is an essential starting point for most more complex communication tasks. For example, wake-up is used as a subroutine in the fastest known protocols for fundamental tasks of *leader election* and *clock synchronization* (cf. [8]). These are two fundamental tasks in networks without global clocks, since they allow initially unsynchronized networks to be brought to a state in which synchronization can be assumed, and results from the better-understood setting with a global clock can then be applied. Our wake-up protocol yields $O(\frac{\min\{n,D\Delta\}\log^2 n\log\Delta}{\log\log\Delta})$-time leader election and clock synchronization algorithms, which are the fastest known in both directed and undirected networks.

1.5 Previous approaches

Almost all deterministic broadcasting protocols with sub-quadratic complexity (that is, since [5]) have made use of the concept of *selective families* (or some similar variant thereof, such as selectors). These are families of sets for which one can guarantee that any subset of $[n] := \{1, 2, \dots, n\}$ below a certain size has an intersection of size exactly 1 with some member of the family. They are useful in the context of radio networks because if the members of the family are interpreted to be the set of nodes which are allowed to transmit in a particular time-step, then after going through each member, any node with an active in-neighbor and an in-neighborhood smaller than the size threshold will be informed. Most of the recent improvements in broadcasting time have been due to a combination of proving smaller selective families exist, and finding more efficient ways to apply them (i.e., choosing which size of family to apply at which time).

One of the drawbacks of selective-family based algorithms is that applying them requires coordination between nodes. For the problem of broadcast, this means that some time may be wasted waiting for the current selective family to finish, and also that nodes cannot alter their behavior based on the time since they were informed, which might be desirable. For the problem of wake-up, this is even more of a difficulty; since we cannot assume a global clock, we cannot synchronize node behavior and hence cannot use selective families at all.

To tackle this issue, Chrobak et al. [8] introduced the concept of *radio synchronizers*. These are a development of selective families which allow nodes to begin their behavior at different times. A further extension to *universal synchronizers* in [4] allowed effectiveness across all in-neighborhood sizes. However, the adaptability to different node start times comes at a cost of increased size, meaning that synchronizer-based wake-up algorithms were slightly slower than selective family-based broadcasting algorithms .

The proofs of existence for selective families and synchronizers follow similar lines: a probabilistic candidate object is generated by deciding on each element independently at random with certain carefully chosen probabilities, and then it is proven that the candidate satisfies the desired properties with positive probability, and so such an object must exist. The proofs are all non-constructive (and therefore all resulting algorithms non-explicit; cf. Indyk [14] for an explicit construction of selective families).

Returning to the problem of broadcasting, a breakthrough came in 2010 with a paper by De Marco [20] which took a new approach. Rather than having all nodes synchronize their behavior, it instead had them begin their own unique pattern, starting immediately upon being informed. These behavior patterns were collated into a transmission matrix. The existence of a transition matrix with appropriate selective properties was then proven probabilistically. The ability for a node to transmit with a frequency which decayed over time allowed De Marco's method to inform nodes with a very large in-neighborhood faster, and this in turn reduced total broadcasting time from $O(n\log^2 D)$ [11] to $O(n\log n\log\log n)$.

A downside of this new approach is that having nodes begin immediately, rather than wait until the beginning of the next selector, gives rise to a far greater number of possible starting-time scenarios that have be accounted for during the probabilistic proof. This caused the logarithmic factor in running time to be $\log n$ rather than $\log D$. Further, the method was comparatively slow to inform nodes of low in-degree, compared to a selective family of appropriate size. These are the difficulties that our approach will have to overcome.

1.6 Overview of our approach

Our wake-up result follows a similar line to the previous works; we prove the existence of smaller universal synchronizers than previously known, using the probabilistic method. Our improvement stems from new techniques in analysis rather than method, which allow us to gain a log-logarithmic factor by choosing what we believe are the optimal probabilities by which to construct a randomized candidate.

Our broadcasting result takes a new direction, some elements of which are new and some of which can be seen as a compromise between selective family-type objects and the transmission schedules of De Marco [20]. We first note that nodes of small in-degree can be quickly dealt with by repeatedly applying $(n, \frac{n}{D})$-selective families "in the background" of the algorithm. This allows us to tailor the more novel part of the approach to nodes of large in-degree. We have nodes perform their own behavior patterns with decaying transmission frequency over time, but they are semi-synchronized to "blocks" of length roughly $\frac{n}{D}$, in order to cut down the number of circumstances we must consider. This idea is formalized by the concept of *block synchronizers,* combinatorial objects which can be seen as an extension of the radio synchronizers used for wake-up.

An important new concept used in our analysis of block synchronizers (and also in our proof of small universal synchronizers) is that of *cores*. Cores reduce a set of nodes and starting times to a (usually smaller) set of nodes which are active during a critical period. In this way we can combine many different circumstances into a single case, and demonstrate that for our purposes they all behave in the same way.

The most technically involved part of both of the proofs is the selection of the probabilities with which we generate a randomized candidate object (universal synchronizer or block synchronizer). Intuitively, when thinking about radio networks, a node in our network is aiming to inform its out-neighbors, and it should assume that as time goes on, only those with large in-neighborhoods will remain uninformed (because these nodes are harder to inform quickly). Therefore a node should transmit with ever-decreasing frequency, roughly inversely proportional to how large it estimates remaining uninformed neighbors' in-neighborhoods must be. However, these in-neighborhoods cannot be estimated exactly, and so we must tweak the probabilities slightly to cover the possible range. In block synchronizers we do this using phases of length $O(\log\log\frac{D\Delta}{n})$ during which nodes halve their transmission probability every step, but since behavior must be synchronized to achieve this we cannot do the same for radio synchronizers. Instead, we allow our estimate to be further from the true value, and require more time-steps around the same value to compensate.

As with previous results based on selective families, synchronizers, or similar combinatorial structures, the proofs of the structures we give are non-constructive, and therefore the algorithms are non-explicit.

2 Combinatorial tools

Our communications protocols rely upon the existence of objects with certain combinatorial properties, and we will separate these more abstract results from their applications to radio networks. In this section, we will define the combinatorial objects we will need to make use of, and prove their existence. Next, in Sections 3–4, we will demonstrate in details how these combinatorial objects can be used to obtain fast algorithms for broadcasting and wake-up.

2.1 Selective families

We begin with a brief discussion about *selective families*, whose importance in the context of broadcasting was first observed by Chlebus et al. [5]. A selective family is a family of subsets of $[n] := \{1, \dots, n\}$ such that every subset of $[n]$ below a certain size has intersection of size exactly 1 with a member of the family. For the sake of consistency with successive definitions, rather than defining the family of subsets S_i, we will instead use the equivalent definition of a set of binary sequences S^v (that is, $S^v_i = 1$ if and only if $v \in S_i$).

For some $m \in \mathbb{N}$, let each $v \in [n]$ have its own length-m binary sequence $S^v = S^v_0 S^v_1 S^v_2 \dots S^v_{m-1}$.

▶ **Definition 1.** $S = \{S^v\}_{v \in [n]}$ is an (n, k)**-selective family** if for any $X \subseteq [n]$ with $1 \le |X| \le k$, there exists $j \in [0, m)$ such that $\sum_{v \in X} S^v_j = 1$. (We say that such j *hits* X.)

2.1.1 Application to radio networks

During the course of radio network protocols we can "apply" a selective family S on an n-node network by having each node v transmit in time-step j if and only if v has a message it wishes to transmit and $S^v_j = 1$ (see, e.g., [5, 10]). Some previous protocols involved nodes starting to transmit immediately if they were informed of a message during the application of a selective family (or a variant called a selector designed for such a purpose), but here we will require nodes to wait until the current selective family is completed before they start participating. That is, nodes only attempt to transmit their message if they knew it at the beginning of the current application.

The result of applying an (n, k)-selective family is that any node u which has between 1 and k active neighbors before the application will be informed of a message upon its conclusion. This is because there must be some time-step j which hits the set of u's active neighbors, and therefore exactly one transmits in that time-step, so u receives a message. This method of selective family application in radio networks was first used in [5].

2.1.2 Existence of small selective families

The following standard lemma (see, e.g., [10]) posits the existence of (n, k)-selective families of size $O(k \log \frac{n}{k})$. This has been shown to be asymptotically optimal [10].

▶ **Lemma 2 (Small selective families).** *For some constant c and for any $1 \le k \le n$ there exists an (n, k)-selective family of size at most $m = ck \log \frac{n}{k}$.*

2.2 Radio synchronizers

Radio synchronizers are an extension of selective families designed to account for nodes in a radio network starting their behavior patterns at different times, and without access to a global clock. They were first introduced in [8] and used in an algorithm for performing wake-up, and this is also the purpose for which we will apply them.

To define radio synchronizers, we first define the concept of *activation schedule*.

▶ **Definition 3.** An n**-activation schedule** is a function $\omega : [n] \to \mathbb{N}$.

We will extend the definition to subsets $X \subseteq [n]$ by setting $\omega(X) = \min_{v \in X} \omega(v)$.

As for selective families, let each $v \in [n]$ have its own length-m binary sequence $S^v = S^v_0 S^v_1 S^v_2 \dots S^v_{m-1}$. We then define radio synchronizers as follows:

▶ **Definition 4.** $S = \{S^v\}_{v \in [n]}$ is an (n, k, m)**-radio synchronizer** if for any activation schedule ω and for any $X \subseteq [n]$ with $1 \leq |X| \leq k$, there exists $j \in [\omega(X), \omega(X) + m)$ such that $\sum_{v \in X} S^v_{j-\omega(v)} = 1$.

One can see that the definition is very similar to that of selective families (Definition 1), except that now each v's sequence is offset by the value $\omega(v)$. To keep track of this shift in expressions such as the sum in the definition, we will call such values j *columns*. As with selective families, we say that any column j satisfying the condition in Definition 4 *hits* X.

In [4], this concept was extended to universal radio synchronizers which cover the whole range of k from 1 to n. Let $g : [n] \to \mathbb{N}$ be a non-decreasing function, which we will call the *delay* function.

▶ **Definition 5.** $S = \{S^v\}_{v \in [n]}$ is an (n, g)-**universal radio synchronizer** if for any activation schedule ω, and for any $X \subseteq [n]$, there exists column $j \in [\omega(X), \omega(X) + g(|X|))$ such that $\sum_{v \in X} S^v_{j-\omega(v)} = 1$.

2.2.1 Application of universal radio synchronizers to radio networks

One can apply universal radio synchronizers to the problem of wake-up in radio networks by having $\omega(v)$ represent the time-step in which node v becomes active during the course of a protocol (either spontaneously or by receiving a transmission). Subsequently, v interprets S^v as the pattern in which it should transmit, starting immediately from time-step $\omega(v)$. That is, in each time-step j after activation, v checks the next value in S^v (i.e., $S^v_{j-\omega(v)}$), transmits if it is **1** and stays silent otherwise. Then, the selective property specified by the definition guarantees that any node u with an in-neighborhood of size q hears a transmission within at most $g(q)$ steps of its first in-neighbor becoming active.

2.2.2 Existence of small universal radio synchronizers

We will prove existence of small universal synchronizers.

▶ **Theorem 6.** *For some constant c and for any $n \in \mathbb{N}$ there exists an (n, g)-**universal radio synchronizer** with $g(q) = \frac{cq \log q \log n}{\log \log q}$.*

The proof of Theorem 6 is deferred to the full version, but here we present its main ideas.

Our universal synchronizer must hit any set X within $g(X)$ columns of the set's start column $\omega(X)$. Our first step is to narrow down the amount of sets and activation schedules we must consider by defining the *core* of a set. The core removes elements of X which only become active after we must already have hit X (i.e. after column $\omega(X) + g(X)$), and then shifts all values of ω so that $\omega(X) = 0$. Multiple different sets X under multiple different activation schedules can have the same core, and therefore behave the same way during the critical period. So, by considering cores instead of the original sets and activation schedules, we can reduce the amount of cases we must account for.

We then probabilistically generate a candidate universal synchronizer, and prove that it does indeed satisfy the required property with positive probability. The crucial part of this method is our choice of probabilities with which we include each element in the candidate.

We expect to be able to hit a core of size q within $\frac{cq \log q \log n}{\log \log q}$ columns. Therefore, for a particular element v and for columns greater than $\omega(v) + \frac{cq \log q \log n}{\log \log q}$, all remaining cores containing v that we have left to hit should be larger than q. If we want to hit a set of size roughly q by randomly selecting each element of $[n]$, then we optimize our probability of doing so by selecting each element with probability roughly $\frac{1}{q}$. This translates to setting

$S^v_{\frac{cq\log q\log n}{\log\log q}} = 1$ with probability roughly $\frac{1}{q}$. Substituting $j = \frac{cq\log q\log n}{\log\log q}$, this works out to give the probability of $S^v_j = 1$ to be roughly $\frac{\log j\log n}{j\log\log j}$. However, as it transpires, this is not quite the optimal choice.

To see what we should choose instead, we must first examine how we analyze our candidate. We define the *load* $f_C(j)$ of a column j with respect to a core C; this is the expected number of elements of the core selected in that column. We show that the probability of hitting the core on j is at least $f_C(j)2^{-f_C(j)}$. Ideally, we want $f_C(j)$ to be constant for cores of roughly the size we should currently be hitting, and then this hitting probability is also constant. However, if we use the probabilities $\mathbf{P}[S^v_j = 1] = \frac{\log j\log n}{j\log\log j}$, we in fact find that $f_C(j)$ can be between $\Omega(1)$ and $O(\log j)$, depending upon the times at which its elements became active. This $\log j$-sized gap cannot be closed, since we can easily find cores which do indeed have loads at either end of the range. What we can do, however, is tweak the probabilities to shift this range down and maximize hitting probability over it.

Specifically, if we shift probabilities down by a factor of $\frac{\log j}{\log\log j}$, i.e. to $\mathbf{P}[S^v_j = 1] \approx \frac{\log n}{j}$, we shift the range of possible loads to be between $\Omega(\frac{\log\log j}{\log j})$ and $O(\log\log j)$, ensuring that the hitting probability is $\Omega(\frac{\log\log j}{\log j})$. This is as close to constant as we can get; the gap is what necessitates the $\frac{\log q}{\log\log q}$ factor in our delay function g.

With this bound on the probability of hitting a particular core at a particular column, we can use independence to obtain an lower bound for hitting a core over the whole range of valid columns, and then use a union bound to show that we can hit all cores with positive probability.

2.3 Block synchronizers

Next, we introduce *block synchronizers*, which are a new type of combinatorial object designed for use in a fast broadcasting algorithm. They can be seen as an extension of both radio synchronizers and the transmission matrix formulation of De Marco [20].

Let each $v \in [n]$ have its own length-m binary sequence $S^v = S^v_0 S^v_1 S^v_2 \dots S^v_{m-1}$. Define a function $\mu_B : \mathbb{N} \to \mathbb{N}$, for some fixed B, which rounds its input up to the next multiple of B, that is, $\mu_B(x) = \min\{pB : p \geq \frac{x}{B}, p \in \mathbb{N}\}$; we will call $s(v) := \mu_B(\omega(v))$ the *start column* of v. We extend s to subsets of $[n]$ in the obvious way, $s(X) = \mu_B(\omega(X))$.

▶ **Definition 7.** $S = \{S^v\}_{v\in[n]}$ is an (n, Δ, r, B)-**block synchronizer** if for any activation schedule ω and any set $X \subseteq [n]$ with $|X| \leq \Delta$, there exists column $j \in [s(X), s(X) + B\lceil\frac{|X|}{r}\rceil)$ such that $\sum_{v\in X} S^v_{j-s(v)} = 1$.

Block synchronizers differ from radio synchronizers in two ways: Firstly, on top of the offsetting effect of the activation schedule, there is also the function μ_B that effectively "snaps" behavior patterns to blocks of size B, hence the name block synchronizer. Secondly, the size of the range in which we must hit X is linearly dependent on $|X|$ rather than being fixed. The parameter r is the increment by which each block increases the size of sets we can hit.

2.3.1 Application of block synchronizers to radio networks

The idea of our broadcasting algorithm will be that any node v waits until the start of the first block after its activation time $\omega(v)$, and then begins its transmission pattern S^v. The definition of block synchronizer aims to model this scenario. The hitting condition ensures that any node with an in-neighborhood of size $q \leq \Delta$ will be informed within $B\lceil\frac{q}{r}\rceil$ time-steps of the start of the block in which its first in-neighbor begins transmitting.

2.3.2 Existence of small block synchronizers

We prove the following theorem.

▶ **Theorem 8.** *For some constant c and for any $n, D, \Delta \in \mathbb{N}$ with D, $\Delta \leq n < D\Delta$, there exists an $(n, \Delta, \frac{n}{D}, c\frac{n}{D}\log D \log\log \frac{D\Delta}{n})$-**block synchronizer**.*

The proof of this theorem is deferred to the full version, but we outline its main ideas.

We begin in a similar way to that of Theorem 6, in that we define the concept of a core to narrow down the range of possibilities we must consider. The difference is that now elements v have their behavior patterns snapped to blocks, so the core need only store the number of the first block v becomes active, rather than the exact column. This significantly reduces the number of possible cores, and is essential for obtaining the $\log D$ factor in size, rather than $\log n$. As before, cores also shift the activation schedule to begin at 0.

We wish to proceed in a similar manner as in the proof of small universal synchronizers, generating a candidate probabilistically and then proving that it satisfies the desired property with positive probability. While we could do this directly for block synchronizers using the same method, we would obtain a size of $\Theta(n \log D \log\log \Delta)$. Our improved bound of $O(n \log D \log\log \frac{D\Delta}{n})$ results from noting that we can afford to hit cores smaller than $\frac{n}{D}$ using selective families, leaving a narrower range of cores to be hit by our randomized candidate object. Therefore, we define an *upper block synchronizer* which need only hit cores above a certain size threshold (in our case, $\frac{n}{D}$), and this is the object we prove the existence of with the probabilistic method.

Once again, the most technical aspect of the proof lies in the probabilities with which we choose elements in our candidate object. Again, we define the load of a column $f_{\mathcal{C}}(j)$ to be the expected number of elements in a core $\mathcal{C}$ which are selected in column j, and show that the probability of hitting the core at j is at least $f_{\mathcal{C}}(j)2^{-f_{\mathcal{C}}(j)}$. By choosing our probabilities to be roughly inversely proportional to the size of cores we expect to remain un-hit, we can guarantee that on a constant fraction of columns we have $f_{\mathcal{C}}(j) = \Omega(1)$ and $f_{\mathcal{C}}(j) = O(\log \frac{Dj}{n}) = O(\log \frac{D\Delta}{n})$. However, the way in which we deal with this range differs from the case of universal synchronizers.

Since our elements' behavior is snapped to blocks, we can synchronize changes in probability of selection between all elements. Specifically, we have $O(\log\log \frac{D\Delta}{n})$-length phases in which the selection probability of all elements halves every consecutive column (in addition to the slight decrease that occurs between columns naturally). This means that $f_{\mathcal{C}}(j)$ also roughly halves every column within phases, and so there is at least one column within each phase in which it is within some constant range (we use the interval $(\frac{1}{3}, 1)$). Then, the probability of hitting the core at that column is also at least some constant.

With this bound, we can again use independence of columns and a union bound to show that our candidate upper block synchronizer hits all cores larger than $\frac{n}{D}$ with positive probability. Then, it remains only to insert a $(n, \frac{n}{D})$-selective family at the start of every block in order to hit smaller cores, and we meet the conditions of a block synchronizer.

3 Algorithms for broadcasting and wake-up

In this section we present our algorithms for broadcasting and wake-up in radio networks.

3.1 Broadcasting

We will assume that $D\Delta > n$, otherwise an earlier $O(D\Delta \log \frac{n}{\Delta})$-time protocol from [10] can be used to achieve $O(D\Delta \log \frac{n}{\Delta}) = O(n \log D)$ time.

Let $\mathcal{S}$ be an $(n, \Delta, \frac{n}{D}, \mathcal{B})$-block synchronizer, with $\mathcal{B} = c\frac{n}{D}\log D \log\log\frac{D\Delta}{n}$, and recall that $\mu_{\mathcal{B}}(x) = \min\{p\mathcal{B} : p \geq \frac{x}{\mathcal{B}}, p \in \mathbb{N}\}$. We will say that the source node becomes active at time-step 0, and any other node v becomes active in a time-step i if it received its first transmission at time-step $i - 1$. Our broadcasting algorithm is the following (Algorithm 1):

Algorithm 1 Broadcast at a node v

```
Let $i$ be the time-step in which $v$ becomes active
**for** $j$ from 0 to $DB - 1$, in time-step $\mu_{\mathcal{B}}(i) + j$ **do** $v$ transmits source message iff $\mathcal{S}^v_j = 1$
**end for**
```

3.2 Wake-up

Let S be an (n, g)-universal radio synchronizer with $g(q) = \frac{cq\log q\log n}{\log\log q}$. We will say that a node v becomes active in a time-step i if it either spontaneous wakes up at i, or received its first transmission at time-step $i - 1$. Our wake-up algorithm is the following (Algorithm 2):

Algorithm 2 Wake-up at a node v

```
Let $i$ be the time-step in which $v$ becomes active
**for** $j$ from 0 to $g(n) - 1$, in time-step $i + j$ **do** $v$ transmits source message iff $S^v_j = 1$
**end for**
```

4 Analysis of broadcasting and wake-up algorithms

In this section we show that our algorithms for broadcasting and wake-up have the claimed running times. We begin with the analysis of the broadcasting algorithm.

▶ **Theorem 9.** *Algorithm 1 performs broadcast in $O(n\log D\log\log\frac{D\Delta}{n})$ time-steps.*

To begin the analysis, fix some arbitrary node v and let P be a shortest path from the source (or first informed node) x to v. Number the nodes in this path consecutively, e.g., $P_0 = x$ and $P_{dist(x,v)} = v$. Classify all other nodes into *layers* dependent upon the furthest node along the path P to which they are an in-neighbor (some nodes may not be an in-neighbor to any node in P; these can be discounted from the analysis). That is, layer $L_\ell = \{u \in V : \max_{u \text{ in-neighbour to } P_i} i = \ell\}$ for $\ell \leq dist(x, v)$. We separately define layer $L_{dist(x,v)+1}$ to be $\{v\}$.

At any time step, we call a layer *leading* if it is the foremost layer containing an active node, and our goal is to progress through the network until the final layer is leading, i.e., v is active. The use of layers allows us to restrict to the set of nodes of our main interest: if we focus on the path node whose in-neighborhood contains the leading layer, we cannot have interference from earlier layers since they contain no in-neighbors of this path node, and we cannot have interference from later layers since they are not yet active.

▶ **Lemma 10.** *Let $h : [\Delta] \to \mathbb{N}$ be a non-decreasing function, and define $T(n, D, \Delta, h)$ to be the supremum of the function $\sum_{i=1}^{D} h(q_i)$, where integers $1 \leq q_i \leq \Delta$ satisfy the additional constraint $\sum_{i=1}^{D} q_i \leq n$. If a broadcast or wake-up protocol ensures that any layer (under any choice of v) of size q remains leading for no more than $h(q)$ time-steps, then all nodes become active within $T(n, D, \Delta, h)$ time-steps.*

Proof. Let $q_i = |L_i|$. Layer $L_{dist(x,v)+1}$ must be leading (and thus node v active) once no other layers are leading, and so this occurs $\sum_{i=1}^{dist(x,v)} h(q_i)$ time-steps after layer L_1

becomes leading. Since $\sum_{i=1}^{dist(x,v)} h(q_i) \le \sum_{i=1}^{D} h(q_i)$ and $\sum_{i=1}^{D} q_i \le n$, this is no more than $T(n, D, \Delta, h)$ time-steps. Since v was chosen arbitrarily, all nodes must be active within $T(n, D, \Delta, h)$ time-steps of x becoming active. ◀

We make use of Lemma 10 to give bounds on the running times of our algorithms:

▶ **Lemma 11.** *Algorithm 1 ensures that any layer of size q remains leading for fewer than $\mathcal{B}\lceil \frac{q+r}{r} \rceil$ time-steps.*

Proof. For all nodes w, let $\omega(w)$ be the time-step that w becomes active during the course of the algorithm. By definition of a block selector, for any layer L_i of size q_i there is a time-step $j < s(L_i) + B\lceil \frac{q_i}{r} \rceil$ in which exactly one element of L_i transmits. Then, either path node P_i hears the transmission (and so layer L_i is no longer leading in time-step $j+1$), or P_i has active in-neighbors not in L_i, in which case these must be in a later layer so L_i is not leading. Thus, L_i can remain leading for no more than $s(L_i) + \mathcal{B}\lceil \frac{q_i}{r} \rceil - \omega(L_i) < \mathcal{B}\lceil \frac{q_i+r}{r} \rceil$ time-steps. ◀

With these tools, we are now ready to complete the proof of Theorem 9.

Proof. Proof of Theorem 9 By Lemma 10, Algorithm 1 ensures that all nodes are active (and have therefore heard the source message) within $T(n, D, \Delta, h)$ time-steps, where $h(q) = \mathcal{B}\lceil \frac{q+r}{r} \rceil$. We will use an upper bound $T(n, D, \Delta, h')$, where $h'(q) = \mathcal{B}\frac{q+2r}{r}$. Since h' is linear and increasing, $\sum_{i=1}^{D} h'(q_i)$ subject to $\sum_{i=1}^{D} q_i \le n$ is maximized whenever $\sum_{i=1}^{D} q_i = n$, for example at $q_i = \frac{n}{D}$ for all $i \in [D]$. So, the algorithm completes broadcast within

$$\sum_{i=1}^{D} h'(\frac{n}{D}) = \sum_{i=1}^{D} \mathcal{B}\frac{\frac{n}{D} + 2r}{r} = 3\mathcal{B}D = 3c'n \log D \log\log \frac{D\Delta}{n}$$

time-steps. ◀

In a similar way, we can analyze Algorithm 2.

▶ **Theorem 12.** *Algorithm 2 performs wake-up in $O(\frac{\min(n, D\Delta) \log n \log \Delta}{\log\log \Delta})$ time-steps.*

5 Conclusions

The task of broadcasting in radio networks is a longstanding, fundamental problem in communication networks. Our result for deterministic broadcasting in directed networks combines elements from several of the previous works with some new techniques, and, in doing so, makes a significant improvement to the fastest known running time. Our algorithm for wake-up also improves over the previous best running time, and relies on a proof of smaller universal synchronizers, a combinatorial object first defined in [4].

Neither of these algorithms are known to be optimal. The best known lower bound for both broadcasting and wake-up is $\Omega(\min(n \log D, D\Delta \log \frac{n}{\Delta}))$ [10]; our broadcasting algorithm therefore comes within a log-logarithmic factor, but our wake-up algorithm remains a logarithmic factor away.

As well as the obvious problems of closing these gaps, there are several other open questions regarding deterministic broadcasting in radio networks. Firstly, the lower bound for undirected networks is weaker than that for directed networks [18], and so one avenue of research would be to find an $O(n \log D)$ lower bound in undirected networks, matching the broadcasting time of [16]. Secondly, the algorithms given here, along with almost all previous work, are non-explicit, and therefore it remains an important challenge to develop explicit algorithms that can come close to the existential upper bound. The best constructive algorithm known to date is by Indyk [14], but it is a long way from optimality.

Some variants of the model also merit interest, in particular the model with collision detection. It is unknown whether the capacity for collision detection improves deterministic broadcast time, as it does for randomized algorithms [12]. Collision detection does remove the requirement of spontaneous transmissions for the use of the $O(n)$ algorithm of [5], but a synchronized global clock would still be required. It should be noted that collision detection renders the wake-up problem trivial, since if every active node transmits in every time-step, collisions will wake up the entire network within D time-steps.

References

1 N. Alon, A. Bar-Noy, N. Linial, and D. Peleg. A lower bound for radio broadcast. *Journal of Computer and System Sciences*, 43(2):290–298, 1991.

2 R. Bar-Yehuda, O. Goldreich, and A. Itai. On the time-complexity of broadcast in multi-hop radio networks: An exponential gap between determinism and randomization. *Journal of Computer and System Sciences*, 45(1):104–126, 1992.

3 B. Chlebus, L. Gąsieniec, D. R. Kowalski, and T. Radzik. On the wake-up problem in radio networks. In *Proceedings of the 32nd Annual International Colloquium on Automata, Languages and Programming (ICALP)*, pages 347–359, 2005.

4 B. Chlebus and D. R. Kowalski. A better wake-up in radio networks. In *Proceedings of the 23rd Annual ACM Symposium on Principles of Distributed Computing (PODC)*, pages 266–274, 2004.

5 B. S. Chlebus, L. Gąsieniec, A. Gibbons, A. Pelc, and W. Rytter. Deterministic broadcasting in unknown radio networks. *Distributed Computing*, 15(1):27–38, 2002.

6 B. S. Chlebus, L. Gąsieniec, A. Östlin, and J. M. Robson. Deterministic radio broadcasting. In *Proceedings of the 27th Annual International Colloquium on Automata, Languages and Programming (ICALP)*, pages 717–728, 2000.

7 B. S. Chlebus, D. R. Kowalski, and A. Pelc. Electing a leader in multi-hop radio networks. In *Proceedings of the 16th International Conference on Principles of Distributed Systems (OPODIS)*, pages 106–120, 2012.

8 M. Chrobak, L. Gąsieniec, and D. R. Kowalski. The wake-up problem in multihop radio networks. *SIAM Journal on Computing*, 36(5):1453–1471, 2007.

9 M. Chrobak, L. Gąsieniec, and W. Rytter. Fast broadcasting and gossiping in radio networks. *Journal of Algorithms*, 43(2):177–189, 2002.

10 A. E. F. Clementi, A. Monti, and R. Silvestri. Distributed broadcasting in radio networks of unknown topology. *Theoretical Computer Science*, 302(1-3):337–364, 2003.

11 A. Czumaj and W. Rytter. Broadcasting algorithms in radio networks with unknown topology. In *Proceedings of the 44th IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 492–501, 2003.

12 M. Ghaffari, B. Haeupler, , and M. Khabbazian. Randomized broadcast in radio networks with collision detection. In *Proceedings of the 32nd Annual ACM Symposium on Principles of Distributed Computing (PODC)*, pages 325–334, 2013.

13 M. Ghaffari and B. Haeupler. Near optimal leader election in multi-hop radio networks. In *Proceedings of the 24th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 748–766, 2013.

14 P. Indyk. Explicit constructions of selectors and related combinatorial structures, with applications. In *Proceedings of the 13th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 697–704, 2002.

15 T. Jurdziński and D. R. Kowalski. The wake-up problem in multi-hop radio networks. In M. Y. Kao, editor, *Encyclopedia of Algorithms*. Springer-Verlag, Berlin, 2015.

16 D. Kowalski. On selection problem in radio networks. In *Proceedings of the 24th Annual ACM Symposium on Principles of Distributed Computing (PODC)*, pages 158–166, 2005.

17 D. Kowalski and A. Pelc. Faster deterministic broadcasting in ad hoc radio networks. *SIAM Journal on Discrete Mathematics*, 18:332–346, 2004.

18 D. Kowalski and A. Pelc. Broadcasting in undirected ad hoc radio networks. *Distributed Computing*, 18(1):43–57, 2005.

19 E. Kushilevitz and Y. Mansour. An $\omega(d \log(n/d))$ lower bound for broadcast in radio networks. *SIAM Journal on Computing*, 27(3):702–712, 1998.

20 G. De Marco. Distributed broadcast in unknown radio networks. *SIAM Journal on Computing*, 39(6):2162–2175, 2010.

21 G. De Marco and A. Pelc. Faster broadcasting in unknown radio networks. *Information Processing Letters*, 79:53–56, 2001.

22 D. Peleg. Time-efficient broadcasting in radio networks: A review. In *Proceedings of the 4th International Conference on Distributed Computing and Internet Technology (ICDCIT)*, pages 1–18, 2007.