Radio Network Algorithms for Global Communication

by

Peter Davies

A thesis submitted in partial fulfilment
of the requirements for the degree of

Doctor of Philosophy in Computer Science

University of Warwick
Department of Computer Science

September 2018

Contents

1 Introduction **1**

1.1 Distributed Computing 1

1.2 Radio Networks 2

1.3 Types of Distributed Problem 3

2 Models, Problems, and Preliminaries **4**

2.1 Ad-Hoc Multi-Hop Radio Network Model 4

2.2 Tasks 8

2.3 Literature Review 10

2.4 Overview of Results 13

2.5 Notation and Conventions 14

3 Beep Model Communication **15**

3.1 Related Work 15

3.2 Our Results 16

3.3 Broadcasting 17

3.4 Multi-Broadcast 22

3.5 Lower Bounds 37

3.6 Discussion and Open Problems 43

4 Deterministic Radio Communication **45**

4.1 Related Work 46

4.2 Our Results 49

4.3 Combinatorial Tools 50

4.4 Algorithms for Multiple Access Channels 69

4.5 Analysis for Multi-hop Radio Networks 71

4.6 Discussion and Open Problems 73

5 Randomized Blind Broadcasting **75**
5.1 Related Work . . . 75
5.2 Our Results . . . 76
5.3 Overview of Approach . . . 76
5.4 Protocols . . . 83
5.5 Broadcast in Undirected Networks with Collision Detection . . . 91
5.6 Discussion and Open Problems . . . 91

6 Randomized Leader Election **93**
6.1 Related Work . . . 93
6.2 Our Results . . . 95
6.3 Leader Election Frameworks . . . 95
6.4 Implementation . . . 99
6.5 Running Times . . . 106
6.6 Discussion and Open Problems . . . 107

7 Spontaneous Transmissions **108**
7.1 Related Work . . . 108
7.2 Overview of Approach . . . 109
7.3 Algorithm for COMPETE . . . 114
7.4 Analysis of COMPETE Algorithm . . . 117
7.5 Applying COMPETE to Broadcasting and Leader Election . . . 124
7.6 Clustering property: Proof of Theorem 81 . . . 125
7.7 Discussion and Open Problems . . . 136

Acknowledgements

I would like to thank my supervisor Artur Czumaj, for his guidance, patience, and support throughout my research.

Declaration

This thesis is submitted to the University of Warwick in support of my application for the degree of Doctor of Philosophy. It has been composed by myself and has not been submitted in any previous application for any degree. The work presented was carried out by the author. Parts of this thesis have been published by the author:

Chapter 3:

[17]: Artur Czumaj, Peter Davies: **"Communicating with Beeps"**, proceedings of the 19th International Conference on Principles of Distributed Computing (OPODIS 2015), pages 1-16, 2015. Extended version appeared as **"Communicating with Beeps"**, Journal of Parallel and Distributed Computing (JPDC), 2019.

Chapter 4:

[22] Artur Czumaj, Peter Davies: **"Faster Deterministic Communication in Radio Networks"**, proceedings of the 43rd International Colloquium on Automata, Languages and Programming (ICALP), pages 139:1–139:14, 2016. Extended version appeared as **"Deterministic Communication in Radio Networks"**, SIAM Journal on Computing (SICOMP), 47(1), pages 218-240, 2018.

[21] Artur Czumaj, Peter Davies: **"Deterministic Blind Radio Networks"**, proceedings of the 32nd International Symposium on Distributed Computing (DISC), 2018.

Chapter 5:

[20] Artur Czumaj, Peter Davies: **"Brief Announcement: Randomized Blind Radio Networks"**, proceedings of the 32nd International Symposium on Distributed Computing (DISC), 2018.

Chapter 6:

[18] Artur Czumaj, Peter Davies: **"Brief announcement: Optimal Leader Election in Multi-hop Radio Networks"**, proceedings of the 2016 ACM Symposium on Principles of Distributed Computing (PODC), pages 47-49, 2016. Extended version appeared as **"Leader Election in Multi-hop Radio Networks"**, Theoretical Computer Science, 2019.

Chapter 7:

[19] Artur Czumaj, Peter Davies: **"Exploiting Spontaneous Transmissions for Broadcasting and Leader Election in Radio Networks"**, proceedings of the 2017 ACM Symposium on Principles of Distributed Computing (PODC), pages 3-12, 2017. **PODC 2017 Best Student Paper Award**.

Abstract

Radio networks are a distributed computing model capturing the behavior of devices that communicate via wireless transmissions. Applications of wireless networks have expanded hugely in recent decades due to their convenience and versatility. However, wireless communication presents practical difficulties, particularly in avoiding interference between transmissions. The radio network model provides a theoretical distillation of the behavior of such networks, in order to better understand and facilitate communication.

This thesis concerns fundamental global communication tasks in the radio network model: that is, tasks that require relaying messages throughout the entire network. Examples include broadcasting a message to all devices in a network, or reaching agreement on a single device to act as a coordinator.

We present algorithms to perform global tasks efficiently, and show improved asymptotic running times over a range of environments and model variants. Our results demonstrate an advance over the state of the art in radio network research, and in many cases reach or approach known lower bounds.

Chapter 1

Introduction

Radio networks are a long-standing distributed computing model, designed to capture the behavior of devices communicating wirelessly. Originating in the 1980s, during the rapid rise to prominence of distributed computing, they aim to provide a clean, general, yet powerful theoretical model to underpin the huge variety of devices, protocols, and physical behaviors involved in wireless networks.

1.1 Distributed Computing

Distributed computing is, at its heart, the study of decentralized computation: a collection of entities, each with some individual processing power and memory, are equipped with some means of communicating with each other. The entities and communication links together form a network, with potentially far more power that any of the constituent entities alone. The challenge is to effectively harness that power, by finding efficient ways for entities to share their resources and information, in order to complete some joint task.

Originally, the purpose of distributed computing was to reason about local area networks, small-scale groups of computers linked by technologies such as Ethernet. Quickly, though, applications expanded to encompass a wide array of computational architectures, including CPU cores within a chip, telephone networks, and the Internet. As ambitions of solving computational problems vastly outstripped the capabilities of any one device, virtually all computational systems became, in some way, distributed.

Due to the varied nature of applications for distributed computing, it is a very broad field, uniting aspects of mathematics, statistics, computational theory, software and hardware engineering, and physics. Our focus will be on the theory of communication in radio networks: networks of devices which communicate wirelessly (which generally means using the electromagnetic spectrum). Examples of such networks include mobile telephone networks, wireless Internet and local area networks, and broadcast radio [54].

1.2 Radio Networks

Wireless communication is ubiquitous in modern times, due to its convenience and adaptability compared to networks using physical connections. However, the behavior of wireless signals causes communication challenges: interference, path loss, and obstructions must all be worked around when trying to construct a reliable wireless network. As a result, the main focus of theoretical radio network models is to formulate concrete rules as to when transmissions can and can not be successfully received. Likewise, the problems studied in these models usually revolve around quickly disseminating information despite these communication difficulties (as opposed to other common goals in distributed computing such as efficient allocation of computation and memory resources).

There is a spectrum of approaches one could take to the challenge of formalizing wireless transmission rules. On one side, we could attempt to pin down exactly the physical behavior of the electromagnetic waves, and formulate some condition based on signal strength which tells us when devices can successfully receive transmissions. The advantage of doing so would be that hopefully the model would closely approximate what is possible in reality, allowing highly tailored, efficient algorithms. However, the model would also be highly sensitive to specific network characteristics and hardware capabilities, so developing versatile, transferable solutions to problems may not be possible. Furthermore, if the calculations governing successful transmissions are too complex, this complexity may hinder the development of sophisticated algorithms.

The alternative is to use an abstract model, with simplified criteria for successful transmissions. We would wish to simplify in ways that make the model weaker (i.e. disallow transmissions that could in practice be successful, rather than allowing transmissions that may fail in practice), since we prefer to give up some power than risk developing algorithms that fail on real networks. As a

result, we may lose some efficiency potential, but we gain a wider applicability for the model, and simplicity which makes deeper analysis easier.

The radio network model takes this latter approach, reducing the complex physical phenomena involved in wireless transmission with a graph indicating which pairs of devices are 'in range' of each other. Interference between devices is simplified in a very strict fashion, by assuming that any simultaneous transmissions a device receives from others within range will interfere and fail, providing no useful information. In most cases, this simplification errs on the side of caution, as desired (though it is possible to conceive of circumstances in which interference from devices outside transmission range hinder transmissions deemed successful by the model). The advantage of this abstraction is that we obtain a simple and clean theoretical model which, as we will see, admits some highly nontrivial theory.

1.3 Types of Distributed Problem

In general, almost any problem in theoretical computer science can be posed in a distributed setting, but the specifics and applications of the distributed model will affect which ones are useful and interesting. In the case of radio networks, the focus of the model is to capture the difficulty of ensuring successful transmissions. So, the problems that are studied are mostly based around ensuring messages from certain devices reach others; we are less concerned about the content of these messages, or about computations done by the devices.

One major distinction between problems in networks is that of **global** and **local** problems. Global problems are those in which we may require communication between *any* pair of devices in the network if we are to reach the correct solution. Hence, we cannot hope to solve the problem more quickly than we can relay a message between the furthest two devices, because we may be required to do just that. By contrast, **local** problems are those in which each device can reach the solution by conferring only with *nearby* devices, those in some *local neighborhood* which is strictly smaller than the overall network. In this thesis, we will be concerned with global problems.

Chapter 2

Models, Problems, and Preliminaries

In this chapter we describe in detail the specific models of radio networks that we are concerned with, and the problems we study therein.

2.1 Ad-Hoc Multi-Hop Radio Network Model

The main model we study is the **ad-hoc multi-hop radio network model**. Here the term 'ad-hoc' means that the network has arisen 'on the fly', i.e. there has been no central planning of the positions of the devices involved. 'Multi-hop' means that the devices are geographically far apart and cannot all reach each other directly, so solving global problems will require relaying messages through the network. We will see how these qualifiers affect the formal definition of our model shortly.

The network is modeled by a **directed** graph $\mathfrak{N} = (V, E)$, where the set of nodes V corresponds to the set of devices (and henceforth we will refer to the devices as nodes). A directed edge $(v, u) \in E$ means that node v can send a message directly to node u, i.e. u is within the transmission range of v. To admit global propagation of information (that is, to ensure that any node could send information to any other via some sequence of relayed transmission), we assume that $\mathfrak{N}$ is strongly connected. This is a necessary assumption if we wish to study general global problems.

We denote by n the size of $|V|$, and by D the diameter of $\mathfrak{N}$ (the distance

between the furthest pair of nodes in the graph). Algorithmic running times will be analyzed with respect to these two parameters.

The defining feature of radio networks is the set of rules governing communication. Nodes operate in discrete, synchronous 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. The most standard radio networks model is the **model without collision detection**, in which, 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. 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).

2.1.1 Variants

Real-world networks have a wide variety of specifications and device capabilities, and so many variants of the theoretical models have also arisen to match them. Here we discuss the most common differences between radio network models.

Undirectedness

We initially defined the network to be based on an underlying directed graph. However, for applications in which the devices all have similar transmission power, it may be the case that direct reachability is symmetric (i.e. if u can reach v, then v can reach u). So, it is also of interest to study **undirected** radio networks, where the underlying graph $\mathfrak{N}$ is undirected. This is a stronger model, and admits new techniques in algorithm design, since in undirected networks it is possible for nodes within some local area to communicate between themselves.

Collision detection

An important variant is the capability for *collision detection*. The standard rules for communication assume no collision detection, i.e. a listening node cannot distinguish between 0 transmitting in-neighbors, and multiple transmitting in-neighbors. However, if devices in a specific application are equipped with *carrier sensing* (a means of verifying whether any transmissions are in progress), then

it is more appropriate to model these two scenarios as being distinguishable. Hence, in the **model with collision detection**, listening nodes can hear one of three distinct outcomes: silence, a transmission, or a collision.

Another recently introduced model is closely related to radio networks with collision detection: the **beep model**, introduced by Cornejo and Kuhn [16], models weak devices whose *only* receiver capabilities are carrier sensing, i.e. they do not receive a message even when only one in-neighbor transmits. Listening nodes can only distinguish between 0 in-neighbors transmitting, and at least one. Since message content is now irrelevant, we refer to transmissions as *beeps*, and any information must be communicated by some pattern of beeps and silence.

Parameter knowledge

In the standard model we will assume that nodes have knowledge of the parameters n and D. Assumptions of exact knowledge can usually be relaxed to knowledge of some common upper bounds, say $n' \geq n$ and $D' \geq D$. The limit to which these upper bounds can exceed the true values without affecting asymptotic running time depends upon the algorithm, but is generally either linear (i.e. we would require $n' \leq cn$ for some constant c, such as in [23]) or polynomial (i.e. $n' \leq n^c$, such as in [14]).

However, due to the ad-hoc nature of the model, it may not always be reasonable to assume that nodes have such knowledge. So, we also study the model **without parameter knowledge**, in which nodes have no prior information about the network whatsoever. We will also call this variant the **blind** model.

Global clock

Another type of node knowledge which is often assumed is the presence of a **global clock**. Under this assumption, nodes all know the absolute time-step number (the number of time-steps that have elapsed since the algorithm began). In the weaker **model without a global clock**, nodes do not have this information; this makes it more difficult for them to co-ordinate behavior.

Single-hop

Much of the early research into radio networks focused on the **single-hop** model, where the underlying graph $\mathfrak{N}$ is a clique rather than an arbitrary strongly connected graph. That is, all nodes can communicate with each other directly, and the diameter of the network $D = 1$. These single-hop networks are also

called *multiple access channels*, and model situations when all of the devices in the network are geographically close, or share some direct (i.e. non-wireless) but exclusive communication channel. Single-hop networks are clearly a simpler model, but in some circumstances algorithms for single-hop networks can be simulated in multi-hop networks [3] or even transfer across directly (as we will see in Chapter 4).

Determinism

As in many algorithm design settings, randomization often allows more efficient algorithms. We will design both **randomized algorithms**, which assume nodes each have access to their own stream of random bits, and **deterministic algorithms**, where they do not. In the deterministic case, we require some means of breaking symmetry, since otherwise, for example, broadcasting is impossible on the 4-cycle. So, we assume that each node has a unique integer identifier (ID) between 0 and some parameter L (which is necessarily at least n). Nodes can then base their behavior upon their ID.

Other variants and related models

We will limit our scope in this work to the variants of radio networks described above; however, there is a vast amount of work on similar models. We briefly mention some of the most closely related:

Our work is on **ad-hoc** radio networks, designed to model networks of weak devices, in which the structure may change over time. To model this, we assume that nodes have no prior knowledge of the underlying communication graph. If the network infrastructure is more permanent, it may be more appropriate to assume that nodes do have this knowledge. Networks of this kind are called **known topology**.

Since our motivation is primarily derived from the interaction of devices positioned in physical space, we could consider a geometric representation rather than a (more general) graph-based model for the communication network. In particular, the **unit disk graph** (UDG) model restricts the underlying graph to the family of disk graphs, which are intersection graphs of identical circles on the Euclidean plane. This models a top-down view of devices on the ground, each with the same transmission range.

A further extension of this idea which attempts to capture more specifics of real-world wireless communication is the **SINR** (Signal to Interference plus

Noise Ratio) or **physical** model, which replaces the very strict communication rules of radio networks with a rule based on the physical behavior of electromagnetic waves. Rather that hearing a message iff only one of its neighbors transmits, a node instead hears a message if one neighbor's transmission is sufficiently stronger than all others (plus a noise parameter), determined by a formula based on geometric distances and power-law path loss.

Another popular paradigm among the practical community is **cognitive radio networks**: an attempt to better utilize the wireless spectrum, cognitive radio networks assume devices have access to multiple *communication channels*, and can choose on which one to operate each time-step. This would not help in our setting: since we consider global tasks on a single network, nodes would almost always be best served by choosing to operate on the same channel. The cognitive radio network model is instead aimed at real-world scenarios with multiple overlapping networks and less well-defined communication tasks.

On the more abstract side, much recent work has gone into distributed computing models which remove the communication restrictions of radio networks and instead focus on how *locality* and *congestion* affect graph problems. In the **LOCAL** model, nodes can send arbitrary messages to each of their neighbors (and successfully hear all messages they receive) each time-step. The **CONGEST** model is similar, but restricts message size to $O(\log n)$ bits, which raises the problem of edges becoming congested, i.e. allowing limited information throughput. The **CONGESTED-CLIQUE** model is a variant focused *only* on congestion: the arbitrary input *problem* graph (on which we aim to solve some standard graph problem) is separate from the *communication* graph, which is assumed to be a clique (so all nodes can communicate directly with all others).

2.2 Tasks

We will study several of the most fundamental global communication tasks in radio networks.

2.2.1 Broadcasting and Wake-up

Broadcasting is possibly the most studied problem in radio networks (and many other distributed computing models), due to its status as the simplest global communication task. The task is specified as follows:

A single, arbitrary node is designated the *source*; nodes are aware whether

they are the source or not, but non-source nodes do not initially know the identity of the source. The source node is provided with an arbitrary *message*, which we assume can be passed in a single transmission. Our aim is to provide a schedule, determining when nodes should transmit (and the content of their transmissions), which ensures that there is some time-step in which all nodes can correctly output the source message.

Wake-up is a generalization of broadcasting. Nodes begin in a dormant state in which they can only listen, and *wake up* either spontaneously, at arbitrary (adversarial) time-steps, or upon successfully hearing a transmission. Once nodes are awake, they can choose to transmit as normal. The goal is to ensure that all nodes are woken up, in as few as possible time-steps after the first spontaneous wake-up. This is a harder task than broadcasting since there are effectively multiple sources, which reduces the opportunities for co-ordination. Furthermore, a global clock is not usually assumed for wake-up, whereas for broadcasting one can be simulated by appending the current time-step to the source message.

2.2.2 Leader Election

Leader election is the problem of ensuring that all nodes agree on a single node to be designated leader. Specifically, at the conclusion of a leader election algorithm, all nodes should output the same node ID, and precisely one node should identify this ID as its own. Leader election is a fundamental primitive in distributed computations and, as the most fundamental means of breaking symmetry within radio networks, it is used as a preliminary step in many more complex communication tasks. For example, many fast multi-message communication protocols require construction of a breadth-first search tree (or some similar variant), which in turn requires a single node to act as root (see e.g. [12, 26, 27]).

2.2.3 Multi-broadcast and Gossiping

Multi-broadcast, as the name might suggest, is an extension of broadcasting to accommodate multiple sources. Some subset of source nodes begin with messages at time-step 0, and all source messages must become known by all nodes. We will denote the number of sources k. **Gossiping** is a commonly-studied special case of multi-broadcast in which $k = n$, i.e. all nodes are sources.

That is, in order for gossiping to be successfully completed, all nodes must have heard from all other nodes.

Gossiping, and the generalized problem of multi-broadcast, are interesting problems in ad-hoc radio networks because they incentivize the construction of some kind of structure in the network. Efficient multi-broadcast algorithms aim to build up enough network knowledge to perform the task more quickly than k separate broadcasts.

We will mainly be studying multi-broadcast in the beep model, where message sizes are very important. Because of this, we will define two variants of the multi-broadcast problem: **multi-broadcast with provenance**, where the network must become aware of all (source ID, source message) pairs, and **multi-broadcast without provenance**, where the IDs need not be known. Since we do not assume that messages are unique, we also allow in the case without provenance that only one copy of each distinct message must be output. That is, nodes need not be aware of how many sources held each message.

2.3 Literature Review

Here we give an overview of the state of research into global tasks in the most mainstream model variants, at the time our research began. Later, in each chapter, we will discuss in more detail the most relevant work and any recent advances.

Recall that algorithmic running time is stated in terms of the following parameters: n is the number of nodes in the network, D is the diameter, L is the range of node labels (assumed for deterministic algorithms), M is the message range (important in the beep model), and k is the number of sources for the multi-broadcast task.

2.3.1 Broadcasting

Broadcasting in radio networks has been studied since their inception, with a wealth of literature in many variants of the model.

Chlamtac and Kutten [5] first studied broadcasting in known topology radio networks without collision detection. Here the problem is closer to a classical graph problem than a distributed computing paradigm, since nodes already have all the information they need and must run a centralized algorithm to compute a schedule. Their work was improved upon by several authors over the next

two decades, culminating in an $O(D + \log^2 n)$-time randomized algorithm by Gąsieniec, Peleg and Xin [32], and then a deterministic algorithm with the same running time due to Kowalski and Pelc [41]. This is known to be asymptotically optimal [25].

Work on ad-hoc networks began with Bar-Yehuda et al. [4], who designed the now-ubiquitous randomized DECAY protocol and showed that it can be applied to global broadcast, running in $O((D + \log n) \cdot \log n)$ time and succeeding with high probability. A later $\Omega(D \log \frac{n}{D} + \log^2 n)$ lower bound [2, 46] demonstrated that DECAY was almost optimal, and randomized algorithms reaching this lower bound were subsequently found by Czumaj and Rytter [23], and independently Kowalski and Pelc [44].

Deterministic algorithms in the ad-hoc model have also been long-studied; these usually rely on schedules based on types of combinatorial structures, the existence of which is proven existentially by the probabilistic method. Refinements of these structures gradually reduced running time from quadratic to subquadratic [10] to almost linear [14]. When our research began, the fastest known deterministic broadcasting algorithms were the $O(n \log^2 D)$-time algorithm of Czumaj and Rytter [23] and the $O(n \log n \log \log n)$-time algorithm of De Marco [47] . A gap remained between these running times and the $\Omega(n \log D)$ lower bound of Clementi et al. [15]. In undirected networks Kowalski [42] achieved an $O(n \log D)$ time algorithm, but this is not known to be optimal since the lower bound of [15] is only for directed networks.

The algorithms of [23], [15] and [42] assume that nodes have linear-size labels (i.e. $L = O(n)$), and are non-explicit due to non-constructive proofs of the combinatorial structures involved. Significant advances in explicit algorithms have been made [38, 8], but these remain a poly-logarithmic factor slower.

Since the randomized complexity of broadcast in the most standard model was settled, more recent work on randomized algorithms has focused on variants in which the lower bound no longer holds. In particular, Ghaffari, Haeupler and Khabbazian [28] showed that when collision detection is available this bound can be surpassed, designing an $O(D + \log^6 n)$ time algorithm based on simulating the known-topology approach of [32].

2.3.2 Wake-Up

Wake-up has often been studied in parallel with broadcasting, and both deterministic and randomized wake-up algorithms are generally similar to their

broadcasting counterparts, though with added complications arising from the lack of a global clock.

Deterministic wake-up algorithms again followed a gradual path of improvement from quadratic to almost linear time, based on improvements to combinatorial structures used for schedules. The fastest known algorithm was $O(n \log^2 n)$-time [6]. Research on randomized algorithms tended to focus on single-hop networks, but an $O(n \log^2 n)$-expected time algorithm due to Chrobak, Gąsieniec and Kowalski [13] exists for the multi-hop model.

2.3.3 Leader Election

In one of the classic early works in multi-hop radio networks, Bar-Yehuda et al. [3] developed a general randomized framework of simulating single-hop networks with collision detection by multi-hop networks without collision detection. The framework yields leader election algorithms for multi-hop networks (in directed and undirected networks) running in $O(T_{BC} \cdot \log \log n)$ expected time and $O(T_{BC} \cdot \log n)$ time w.h.p., where T_{BC} is the time required to broadcast a message from a single source to the entire network. The same authors also gave a randomized broadcasting algorithm running in $O(D \log n + \log^2 n)$ time w.h.p., thereby yielding a leader election algorithm taking $O((D \log n + \log^2 n) \log \log n)$ expected time and $O(D \log^2 n + \log^3 n)$ time w.h.p. With the subsequent faster broadcast algorithms of [23] and [44], one obtains leader election algorithms (even in the model without collision detection, and in both directed and undirected graphs) running in $O((D \log \frac{n}{D} + \log^2 n) \log \log n)$ expected time and $O((D \log \frac{n}{D} + \log^2 n) \log n)$ time w.h.p.

In undirected networks, Ghaffari and Haeupler [27] showed a different method based on building local *clusters* and iteratively expanding them by having them compete with neighboring clusters (a process they call a *debate*). They show that in $O(\log \log n)$ rounds of debates, a single cluster dominates the whole network (and its *center* node can be designated leader, completing leader election) with high probability. Thereby they obtained leader election algorithms in undirected networks with and without collision detection with improved high-probability running time of $O(\log \log n)$ times broadcasting time.

A leader election result by Förster, Seidel and Wattenhofer was one of the very few existing algorithms for global problems in the beep model. They achieved an $O(D + \log L)$ deterministic running time.

2.3.4 Gossiping

Gossiping has been studied in both directed and undirected radio networks. In the former, the fastest randomized algorithm is an $O(n \log^2 n)$-time algorithm due to Czumaj and Rytter [23], and the fastest deterministic algorithm is the $O(n^{4/3} \log^4 n)$-time algorithm of [33]. In undirected networks, gossiping can be performed faster using a 'token-passing' graph traversal, after performing leader election to find a root for the spanning tree. Chlebus, Kowalski and Pelc [12] showed how to do this in $O(n \log^{\frac{3}{2}} \sqrt{\log \log n})$ time deterministically, and in $O(n)$ expected time using randomization.

The more general multi-broadcast problem is less well-studied; nevertheless, some results are known, notably the $O(k \log n + D \log^2 n + \log^3 n)$-time randomized algorithm of Khabbazian and Kowalski [40] for undirected networks.

2.4 Overview of Results

We begin, in Chapter 3, by giving algorithms and lower bounds for global tasks in the *beep model*, in which nodes can communicate only by patterns of beeps and silence. This is a comparatively new model for which global communication is not well understood, and our aim is to provide the first comprehensive study thereof. We show optimal algorithms for broadcasting, and near-optimal algorithms for multi-broadcasting (and hence also the special case of gossiping). As a weaker variant of the radio network model with collision detection, the beep model also provides a good introduction to some standard radio network techniques.

In Chapter 4 we present deterministic algorithms for broadcasting and wake-up in radio networks. In the most standard model, directed networks with parameter knowledge and without collision detection, we give the fastest known broadcasting algorithm improving over a long line of previous work and almost reaching the lower bound. We also show how efficient algorithms for broadcasting and wake-up can be achieved even in the more restrictive and less-studied model of blind networks, where nodes have no parameter knowledge. In the process, we show a wake-up algorithm for blind networks that is the fastest known even with parameter knowledge.

Next, in Chapter 5 we show randomized algorithms for the same model of blind networks, demonstrating how randomized broadcasting can be performed efficiently in the absence of parameter knowledge. We start the chapter with an

analysis of the classic DECAY protocol of Bar-Yehuda et al. [4], and the improved global broadcasting algorithms of [23, 43], and then discuss the difficulties in adapting these for the blind network model, and our methods to overcome them.

In Chapter 6, we examine the problem of leader election in radio networks, giving algorithmic frameworks for the task and showing how they can be implemented in networks with and without collision detection.

Finally, in Chapter 7, we show how, if nodes are permitted to spontaneously transmit (i.e. transmit before receiving a source message), broadcasting can be performed more quickly, surpassing the lower bound of [2, 46]. Our result develops and improves over recent work of Haeupler and Wajc [34]. Furthermore, we demonstrate that the method can be further extended to perform leader election at no extra asymptotic cost, significantly improving the running time for that task.

2.5 Notation and Conventions

Our algorithmic running times and lower bounds will be asymptotic, and as a result we will make certain simplifications of notation which do not have an impact on asymptotic results:

- We will, in most cases, omit floor and ceiling functions.
- When taking logarithms, we will use the convention $\log x := \max\{1, \log_2 x\}$.
- Our proofs will often assume that network parameters n and L are 'sufficiently large', i.e. larger than some implied constant.
- We will also use 'sufficiently large' constants in our proofs, with the notation $c, c_1, c_2, c_3, \ldots$. These are large constant values which are necessary for calculations, and for which we shall set an appropriate value during the proof.
- When discussing probabilistic algorithms, we will say that an event occurs 'with high probability' (w.h.p.) if it occurs with probability at least $1-n^{-c}$ for some constant $c \geq 1$.

Chapter 3

Beep Model Communication

As an introduction to some of the techniques we will encounter often when analyzing radio networks, we will first study the simpler *beep model*. Introduced recently by Cornejo and Kuhn [16], the beep model attempts to capture the behavior of very weak devices, making minimal assumptions about their knowledge and capabilities. In addition to the defining feature, that messages cannot be passed by transmission and information can only be conveyed via beeps and silence, most work on the beep model (including that presented here) also assumes nodes have no parameter knowledge. A global clock (or the generally equivalent assumption of *synchronous starts*, where nodes all wake up in the same time-step) is usually assumed.

The beep model removes the necessity of scheduling to avoid collisions, which is the cause of much of the complexity in radio network algorithms. Instead, the main challenge will be how to most efficiently use beeps to convey information.

3.1 Related Work

Cornejo and Kuhn's paper [16] introduced the beep model, and presented an algorithm for interval coloring. This task is a variant of vertex coloring used in resource allocation problems, and is somewhat specific to the model. Addressing a more standard problem in distributed computing, Afek et al. [1] presented an algorithm for finding a maximal independent set, and an algorithm for the

related problem of minimum connected dominating set is given in [56].

These results are all for local problems, which can be completed in $o(D)$ time, i.e. a node can correctly determine its output despite not having time to send or receive any information from most of the network. The first global problem to be studied in the beep model was leader election, with a $O(D \log L)$-time deterministic leader election algorithm given by Förster et al. [26], in the model without parameter knowledge. A randomized leader election algorithm by Ghaffari and Haeupler [27] also exists, running in almost optimal $O((D + \log n \log \log n) \cdot \min\{\log \log n, \log \frac{n}{D}\})$-time, and succeeding with high probability; however it does require parameter knowledge. This work also introduces the method of "beep waves" to transmit bit strings, a method which is also employed here for the purpose of broadcast.

Concurrently with this work, Hounkanli and Pelc [36] give a $O(D + \log M)$ time broadcasting algorithm and an $O(n^2 \log M + nD \log L)$-time gossiping algorithm in the model with parameter knowledge but without a global clock. Further recent works explore other tasks in various related beeping models [9, 35, 37].

3.2 Our Results

Our aim here is to provide the first comprehensive study of global communication algorithms in the beep model. We present the following results:

- An optimal $O(D + \log M)$-time algorithm for broadcasting a $\log M$ bit message, developing and formalizing the "beep waves" method of [27].
- A corresponding $\Omega(D + \log M)$ lower bound.
- An optimal $O(D + \frac{D \log M}{\log D})$-time algorithm for broadcasting a $\log M$ bit message in directed networks.
- A corresponding $\Omega(D + \frac{D \log M}{\log D})$ lower bound.
- An $O(k \log \frac{LM}{k} + D \log L)$-time explicit algorithm, and an optimal $O(k \log \frac{LM}{k} + D)$-time non-explicit algorithm for multi-broadcast with provenance (where every node must learn all (source ID, source message) pairs).
- A corresponding $\Omega(k \log \frac{LM}{k} + D)$ lower bound.

- An explicit algorithm for multi-broadcast without provenance (where every node must learn all unique source messages) taking $O(k \log \frac{M}{k} + D \log L)$ time when $M > k$ and $O(M + D \log L)$ time when $M \leq k$.
- A non-explicit algorithm for multi-broadcast without provenance taking $O(k \log \frac{M}{k} + D + \log L)$ time when $M > k$ and $O(M + D + \log L)$ time when $M \leq k$.
- A corresponding lower bound of $\Omega(k \log \frac{M}{k} + D)$ when $M > k$ and $\Omega(M + D)$ when $M \leq k$.

These multi-broadcasting algorithms imply $O(n \log \frac{LM}{n})$ and $(n \log \frac{M}{n} + \log L)$-time gossiping algorithms with and without provenance respectively.

3.3 Broadcasting

The first, and most basic, task we will consider in the beep model is that of broadcasting, where a source node begins with a message of which to inform all other nodes. Since messages must, in effect, be transmitted "bit by bit" in a pattern of beeps and silence, algorithmic running time is affected by the length of the message we must transmit (this is not generally the case in standard radio networks, where we assume the message can be passed in a single transmission). So, we introduce a new parameter M to specify message range, and assume that all messages to be broadcast are integers in $[M]$.

3.3.1 Broadcasting in Undirected Networks

Broadcasting in undirected networks will be performed using a method known as 'beep waves'. Beep waves were first introduced by Ghaffari and Haeupler [27] as a means of transmitting information in the beep model. Variations of the technique are useful for different circumstances, and here we give a simple formalization tailored to the task of broadcasting from a single source.

The idea is the following: every three time-steps, starting at zero, the source transmits a bit of its message, that is it beeps to represent a **1** or remains silent to represent a **0**. All other nodes aim to relay any beep coming from a neighbor one hop closer to the source, in the next time-step after they hear it. Of course, nodes do not know the provenance of beeps they hear, but we can ensure that nodes will not hear any beeps from their own layer since they will themselves be

beeping rather than listening. We can also stipulate that nodes become 'deaf' and ignore any beeps they hear in time-steps immediately after they transmitted themselves, and this rules out beeps from the next layer. Then, nodes will only relay beeps from the previous layer, so the waves of beeps will emanate out from the source, one distance hop per time-step, and inform all nodes of the source message.

Algorithm 1 BEEP-WAVE(s, m) at source s

```
s beeps at time-step 0
for t = 1 to |m| do
    if bit m_t is 1 then s beeps in time-step 3t
end for
```

Algorithm 1 BEEP-WAVE(s, m) at non-source u

```
j ← first time-step u hears a beep
while end of message not heard do
    if u hears a beep in time-step t ≡ j mod 3 then
        u beeps in time-step t + 1
        bit m(u)_{(t−j)/3} ← 1
    end if
end while
output m(u)
```

Theorem 1. BEEP-WAVE(s, m) *correctly performs broadcast in time* $O(D + |m|) = O(D + \log M)$.

Proof. Partition all nodes into layers depending on their distance from the source s, i.e., layer $L_i = \{v \in V : dist(v, s) = i\}$. We show that a node in layer L_i beeps in time-step t iff $3|t - i$ and either $m_{\frac{t-i}{3}} = 1$ or $t = i$, by induction on t.

For $t = 0$, the claim is trivially true, since the source $s \in L_0$ beeps, and all u in later layers do not.

For $t = t' > 0$, the claim is again clearly true for the source s. Consider a non-source node $u \in L_i$, with $i \leq t'$. Such a node hears its first beep, from a neighbor in layer L_{i-1}, at time-step $i - 1$ by the inductive assumption, and so sets $j = i - 1$. Node u can only beep in time-step t' if $t' \equiv i \bmod 3$, and in this case it beeps only upon hearing a beep in time-step $t' - 1$ (which, by the inductive assumption, can only come from a node in layer L_{i-1}). So, again by

the inductive assumption, $m_{\frac{(t'-1)-(i-1)}{3}} = m_{\frac{t'-i}{3}} = 1$, i.e. u beeps if and only if the correct conditions are satisfied.

When u beeps in time-step t, $m(u)_{\frac{t-1-j}{3}} = m(u)_{\frac{t-i}{3}}$ is set to 1. So, $m_{\frac{t-i}{3}} = 1 \iff m(u)_{\frac{t-i}{3}} = 1$, i.e. u's output message is correct.

By induction the claim is true for all t, and so $m_{\frac{t-i}{3}} = 1 \iff m(u)_{\frac{t-i}{3}} = 1$. Furthermore, after $D + 3\log M$ time-steps, all nodes cease transmission.

□

This is, to our knowledge, the first formalization of beep waves for the task of broadcasting, and the first efficient beeping algorithm for the task.

3.3.2 Broadcasting in Directed Networks

Allowing the underlying graph of the network to be directed greatly restricts what can be done efficiently in the beep model. Beep waves as described above, which are the basis of almost all efficient beeping algorithms, do not work on directed graphs since nodes cannot distinguish between new waves from the source and 'backtracking' from further out layers. In particular, a beep-wave moving through the network can flood all previously reached layers with beeps every time-step, preventing any further communication until it is completed.

Despite these difficulties, we present an algorithm which broadcasts a message in $[M]$ within an optimal $O(\frac{D\log M}{\log D})$ time-steps. We assume throughout that $M \geq D$ (and this is necessary for the running time, since $\Omega(D)$ is a lower bound for broadcasting).

We first give an algorithm which assumes knowledge of D (Algorithm 2), and then describe how it can be extended to remove this assumption. To allow this subsequent extension, we will design Algorithm 2 to broadcast from a set S of sources rather than a single source.

Algorithm 2 DIRECTEDBROADCAST(m, D) at source $s \in S$

```
beep in time-step 0
for j from 1 to log M / log D do
    interpret bits j log D to (j+1) log D − 1 of m as an integer x_j ∈ [0, D−1]
    beep x_j + D + 1 time-steps after previous beep sent
end for
beep 2D + 1 time-steps after previous beep sent
```

The idea of this algorithm is still similar to beep-waves, in that beeps propagate out from the source set one distance layer per time-step. However, these

Algorithm 2 DIRECTEDBROADCAST(m, D) at non-source u

```
when u first hears a beep in time-step i, it beeps in time-step i + 1
loop
    if u hears a beep in time-step t then
        u beeps in time-step t + 1
        x ← number of time-steps since last beep heard
        if x ≤ 2D then append x − D − 1 as a bit-string to m(u)
        else output m(u)
        end if
        u becomes deaf until time-step t + D + 1
    end if
end loop
output m(u)
```

waves could interfere with any layer they have already passed at any later time-step, so the waves cannot be pipelined as before, and we must instead wait D time-steps for the wave to complete before anything more can be done. This is the purpose of nodes becoming *deaf* for D time-steps after relaying a beep; by this we mean that even if nodes hear beeps, they act as if they did not.

Since we cannot pipeline the waves, we instead use their timing to convey additional information; the source set must wait at least $D + 1$ time-steps between waves, but if we allow it to choose any delay between $D + 1$ and $2D$ then it can use these D options to convey $\log D$ bits of the message. In this way we improve run-time by a factor of $\log D$ over the naive approach (of using beep-waves with D time-steps delay).

Lemma 2. *Algorithm 2 performs broadcast from a set of sources S in $O(\frac{D \log M}{\log D})$ time when D is known.*

Proof. Similarly to our analysis of Algorithm 1, we divide nodes into layers based on their distance from the source set, i.e. layer $L_i := \{v \in V : \min_{s \in S} dist(v, s) = i\}$. As before, nodes hear their first beep in time-step $i-1$, and first beep themselves in time-step i.

Let m' be the bit-string transmitted by the sources, i.e. with $m'_0 = 1$ and 1s placed at each interval x_j, where x_j is the integer value of the j^{th} block of $\log D$ message bits, as described. We prove that a node $v \in L_i$ beeps in time-step t iff $m'_{t-i} = 1$, by induction on t.

The base case $t = 0$ is obvious, since sources beep and non-sources do not, as required. Indeed, source nodes clearly have the correct behavior in all time-steps. For the inductive step $t = t'$, we examine a non-source node $v \in L_i$ and

divide into two cases:

Case 1: $m'_{t'-i} = 1$, i.e. v should beep. In this case, by the inductive assumption, in time-step $t'-1$ all nodes in L_{i-1} beep, including a neighbor of v, so we need only show that v is not *deaf* at this time. This is the case, since, again by the inductive assumption, v became deaf the last time it beeped (at time-step $\tilde{t} := t' - 1 - x_j$ for appropriate j), and no nodes in layers $L_{\geq i-1}$ have beeped between time-step $\tilde{t} + D$ and $t' - 2$.

Case 2: $m'_{t'-i} = 0$, i.e. v should not beep. If v has beeped since time-step $t' - (D+1)$ steps then it will be *deaf* and will not beep. Otherwise, the last time-step in which v beeped (again denoted $\tilde{t} := t' - 1 - x_j$ for appropriate j) satisfies $\tilde{t} < t' - (D+1)$, in which case by the inductive assumption no node in layers $L_{\geq i-1}$ beep in time-step $t'-1$, so v is silent in time-step t' as required.

Having proven that the beeping behavior of each node is as expected, it is easy to see that nodes can correctly reconstruct the intervals x_j and therefore the message m from their beeping pattern. Furthermore, all nodes cease beeping after at most $D + 2D\frac{\log M}{\log D} = O(D\frac{\log M}{\log D})$ time-steps. □

This algorithm requires knowledge of D. However, it is easy to see that this assumption can be removed by using a doubling technique. Since nodes know their distance from the source after receiving their first beep, we can have them partition themselves into groups based on an exponentially increasing distance range, i.e., group i consists of nodes of distance between 2^i and 2^{i+1} from the source. Then, we simply perform the algorithm in sequence for each group, with the closest distance layer in the group as the source set and the width of the group as the value for D.

Theorem 3. *There is an algorithm which performs broadcasting in a directed network in the beep model in $O(\frac{D \log M}{\log D})$ time, without knowledge of network parameters.*

Proof. Consider an application of Algorithm 2 to a group i (of width 2^i) as described above. Every beep propagated through the group informs the nodes of $\log 2^i = i$ bits of the message, so after $\frac{\log M}{i}$ rounds broadcast is completed within the group. Each round takes at most 2^{i+1} time-steps, so the total time to broadcast within the group is $O(\frac{2^i \log M}{i})$. Therefore broadcasting is completed in the whole network within $O\left(\sum_{i=1}^{\log D} \frac{2^i \log M}{i}\right)$ time. This can be bounded as follows:

$$\sum_{i=1}^{\log D} \frac{2^i \log M}{i} \leq \log M \left(\sum_{i=1}^{\frac{\log D}{2}} \frac{2^i}{i} + \sum_{i=\frac{\log D}{2}}^{\log D} \frac{2^i}{i} \right)$$
$$\leq \log M \left(\sum_{i=1}^{\frac{\log D}{2}} 2^i + 2 \sum_{i=\frac{\log D}{2}}^{\log D} \frac{2^i}{\log D} \right)$$
$$\leq \log M \left(2\sqrt{D} + \frac{4D}{\log D} \right) = O\left(\frac{D \log M}{\log D} \right) \quad . \qquad \square$$

3.4 Multi-Broadcast

In this section we present our algorithms for the more complex task of multi-broadcast, in undirected networks.

3.4.1 Auxiliary Tasks

Our multi-broadcast algorithms will have a modular structure, i.e. we will use several sub-procedures to solve simpler tasks. We detail these tasks, and the algorithms we will use to solve them:

Broadcasting

The multi-broadcasting algorithms we present will, as one might expect, use single-source broadcasting as a sub-routine, and for this we can make use of Beep-Wave (Algorithm 1). Since we must perform several broadcasts with several different messages, however, we must take care to ensure that these are distinguishable. This can be done by encoding the message so that it is obvious when the beginning and end are, for example by duplicating every bit of the message and then placing **10** at the beginning and end. Note that this coding method does not increase the asymptotic length, in bits, of the message, and that we can decode to find the original message(s), even if there are several, separated by any number of **0**s. We will henceforth assume that all source messages will be encoded in this way.

Algorithm 1 only functions correctly when called with a *single* source node, and so we must somehow have the network agree on which node this source should be. To achieve this agreement, we will use an existing algorithm for *leader election*.

Leader Election

Leader election enables all nodes to agree on the ID of one particular node to designate leader. In our applications, we will always choose the node with the highest ID in the entire network. More generally, though, leader election can be used on any subset of nodes, whenever each holds some integer value, to find the participating node with the highest (or lowest) such value. The values need not even be unique, since if multiple nodes hold the target value, we can pick out one by performing leader election again on their IDs.

We wish to be able to perform leader election in $O(D \log L)$ time. If we assume parameter knowledge, there is a straightforward way to do this: we can perform a binary search for the highest ID, iterating through the bits of the IDs and having all nodes who are still "in the running" for leader, and who have a 1 in the current position, broadcast. While we cannot use our previous broadcast procedure with multiple sources, since these nodes need only transmit a single bit we can still use beep-waves to ensure that the network hears *something*. This is sufficient for all nodes to determine whether any have a 1 in the current position. A similar method to this was used to perform leader election in radio networks in [14].

Without parameter knowledge, however, the task is much more difficult, since without estimates of how long broadcasting, for example, will take, we cannot globally co-ordinate node behavior. Fortunately, one of the few existing results in the beep model is an algorithm by Förster, Seidel, and Wattenhofer [26] that achieves this:

Theorem 4. *There is an algorithm* ElectLeader *which performs leader election in time $O(D \log L)$ without prior knowledge of D or L.* □

Furthermore, upon completion, all nodes have knowledge of the highest ID, and can therefore use this as L in future operations.

To perform further tasks after leader election, we require that nodes should know that leader election is complete and that the next stage should begin. While the leader election algorithm [26] does not immediately allow all nodes to agree on a time-step when this is the case, it does provide the property that the leader is aware of a time-step $t = O(D \log L)$ for which all nodes at distance i from the leader have finished leader election by time-step $t + i$. That is, a procedure commencing with a beep wave from the leader at time-step t will execute successfully. The procedure for diameter estimation we now describe has precisely this property.

Diameter Estimation

Our model assumes that nodes do not have access to any of the network parameters. In algorithms for complex tasks, we generally wish to start with a leader election phase, and this provides all nodes with knowledge of L. However, if we also wish to know the value of D, we must perform an extra task for this purpose.

Our diameter estimation procedure (Algorithm 3) works as follows: we take as input a leader node to co-ordinate the process. An initial beep from the leader propagates through the network. Having received this beep, nodes beep to acknowledge their existence back to the leader; a modularity restriction on when nodes can transmit ensures that these beeps only travel backwards through the layers. While the initial beep from the leader is still reaching further nodes, acknowledgment beeps will continue to return through the network every three time-steps. Once all nodes have been reached, this pattern will cease, and the leader will know the distance of the furthest node, and hence a 2-approximation of diameter. All of the other nodes have also ceased transmission, and so an application of Beep-Wave can safely be used to broadcast the diameter estimate.

We split the algorithm into two parts, one performed by the leader, and one performed by all non-leader nodes, since their behavior is quite different.

Algorithm 3 EstimateDiameter(v) at leader v

v beeps in time-step 2
let t be the first time-step (greater than 3) in which v has not received a beep for 3 previous time-steps
let $\tilde{D} = \frac{2t-8}{3}$
perform Beep-Wave($v, \tilde{D}$)
output $\tilde{D}$

Algorithm 3 EstimateDiameter(v) at non-leader u

let j be the first time-step in which u receives a beep
u beeps in time-step $j + 2$
while u has heard a beep in the last 3 time-steps **do**
    any beep u hears in a time-step equivalent to $j + 1 \bmod 3$, it relays in the next time-step
end while
$\tilde{D} \leftarrow$ Beep-Wave($v, \tilde{D}$)
output $\tilde{D}$

Lemma 5. EstimateDiameter *correctly broadcasts an estimate* $\tilde{D}$ *satisfying* $D \leq \tilde{D} \leq 2D$*, and terminates within* $O(D)$ *time-steps.*

Proof. The first part of the algorithm, in which the leader v beeps in time-step 2 and other nodes relay beeps after two steps, is effectively a beep wave propagating outwards from the leader one hop per two time-steps. It is easy to see that a node at distance i from the leader receives its first beep in time-step $2i$, and so sets $j = 2i$. Furthermore, any node in of distance $i + 1$ receives its first beep in time-step $2i + 2$, and subsequently beeps itself in time-step $2i + 4 \equiv 2i + 1 \bmod 3$. This meets the modularity requirement for a node at distance i to relay the beep in the next time-step. Indeed, in general, the modularity requirement ensures that nodes always relay beeps received from the nodes 1 hop further from the leader, and never relay beeps from nodes 1 hop nearer, or the same distance. So, the effect is that a beep wave is sent back to the leader, every three time-steps, by nodes as they are reached by the initial wave.

When the leader no longer receives these beep waves (i.e. as soon as 3 consecutive time-steps occur with no beep heard), it can conclude that all nodes have been reached by the initial beep-wave and have sent a beep-wave back.

Let D' be the distance from the leader to some the furthest node u. Then, $D \leq 2D' \leq 2D$. The leader emits a beep in time-step 2 which travels to this furthest node in time-step $2D'$. Node u then beeps in time-step $2D' + 2$, and this beep is relayed back to the leader in time-step $3D' + 1$. After another 3 time-steps, the leader knows that it has received the final acknowledgment beep, and sets $t = 3D' + 4$, making its diameter estimate $\tilde{D} = 2D'$. Hence, as required, $D \leq \tilde{D} \leq 2D$.

To analyze running time, notice that the leader v reaches its estimate $\tilde{D}$ in $3D' + 4 = O(D)$ time-steps, and the final beep-wave of this value takes also takes $O(D + \log D) = O(D)$ time. □

Since we are only interested in asymptotic behavior, we will assume, for ease of notation, that having performed EstimateDiameter as part of a more complex algorithm we can then make use of the exact value of D. Furthermore, once leader election and diameter estimation are performed, all nodes have common linear estimates of D and L and so can agree on a time-step in which both tasks are complete and further procedures can commence.

Message Collection

We next introduce a sub-procedure (Algorithm 4) which will allow the leader to collect messages $m(S)$ from a set of sources S, receiving an **OR**-superimposition of all the messages. This works similarly to the usual beep-waves procedure, except that nodes use their distance from the leader (inferred by the time taken to receive the initial BEEP-WAVE$(v, \mathbf{1})$) to ensure that the waves only travel towards the source, and all messages arrive at the same time. We must have an input parameter p giving an upper bound on the length of messages, so that nodes know when the procedure is finished, and we assume that we have already performed ESTIMATEDIAMETER and so can make use of D. We denote by $dist(u)$ the distance from u to the leader node v, which can be determined during an application BEEP-WAVE$(v, \mathbf{1})$.

Algorithm 4 COLLECTMESSAGES$(v, S, m(S), p)$ at node u

perform BEEP-WAVE$(v, \mathbf{1})$
for $j = 0$ to p **do**
  if $m(u)_j = 1$ or u hears a beep in time-step $D - dist(u) + 3j - 1$ **then**
    u beeps in time-step $D - dist(u) + 3j$
    if $u = v$ **then** bit $m(u)_{(j-D)/3} \leftarrow 1$
  end if
end for
output $m(v)$

Lemma 6. COLLECTMESSAGES$(v, S, m(S), p)$ *correctly informs* v *of the* ***OR****-superimposition of* $m(S)$ *within* $O(D + p)$ *time-steps*

Proof. It is clear that (excluding the initial beep wave) a node u at distance $dist(u)$ from the leader v only ever beeps in time steps equivalent to $D - dist(u) \bmod 3$. Furthermore, nodes only relay beeps they hear in time-steps equivalent to $D - dist(u) - 1 = D - (dist(u) + 1) \bmod 3$, i.e. they only relay beeps from nodes one hop further than them from the leader. So, if any source node s has $m(s)_j = 1$ for some j, it beeps in time-step $D - dist(u) + 3j$, and this is relayed back to the leader one distance-hop per time-step. The leader v beeps in time-step $D - dist(u) + 3j + dist(u) = D + 3j$, and hence correctly sets $m(v)_j = 1$.

The running time for the initial beep wave is D steps, and for the loop is $3p + D$. So, total running time is $O(D + p)$. □

Message Length Determination

One issue with using CollectMessages is the necessity of prior knowledge of a common upper bound on message size. We give a simple method of obtaining this bound (Algorithm 5).

We perform CollectMessages using strings which are as long as the messages we actually want to collect, but consist of entirely **1**s. The superimposition of these strings is a **1**-string of equal length to the longest message. Since the leader will be able to tell that this string has ended when it hears the substring **10**, the procedure can be terminated even without an upper bound for the CollectMessages call.

Algorithm 5 GetMessageLength$(v, S, m(S))$

perform $p \leftarrow$ CollectMessages$(v, S, \mathbf{1}^{m(S)}, \infty)$, terminating when v hears the substring **10**
perform Beep-Wave$(v, |p|)$
output $|p|$

Lemma 7. GetMessageLength$(v, S, m(S))$ *correctly informs all nodes of* $q = \max_{s \in S} |m(s)|$ *within* $O(D + q)$ *time-steps*

Proof. CollectMessages will terminate after $D + 3q$ steps, since v will hear the final **1** and then a **0**. All other nodes will be inactive and so Beep-Wave$(v, |p|)$ will successfully inform the network of q (nodes will be aware that the CollectMessages phase is over and so perform Beep-Wave correctly, since they either heard a string of contiguous **1**s and then a **0** during CollectMessages, or silence for more than D time-steps).

Running time is $O(D + q)$ for CollectMessages and $O(D + \log q)$ for Beep-Wave, giving $O(D + q)$ total. □

3.4.2 Explicit Multi-Broadcast Algorithms

We are now ready to combine these sub-procedure to perform multi-broadcast. Recall that we consider two variants of the problem: *multi-broadcast with provenance*, where the network must become aware of all (source ID, source message) pairs, and *multi-broadcast without provenance*, where the IDs need not be known.

Multi-Broadcast With Provenance

We first present an algorithm for multi-broadcast with provenance, where all nodes must be made aware of not only the source messages, but also the IDs of the sources they originated from.

The idea of the algorithm is essentially to conduct k simultaneous binary searches to allow a leader to ascertain the IDs of all sources. The process consists of $\log L$ rounds, one for each bit of the IDs. Each node will maintain a list of known prefixes of source IDs, and we aim to preserve the invariant that, after round i, all nodes know the first i bits of every source ID. We denote the number of distinct known prefixes at the start of round i by k_i.

At the start of round i, sources know k_i distinct $i-1$-bit ID prefixes (note k_i may be less than k, since some IDs may share prefixes), and they will each construct a $2k_i$-bit string in which each bit corresponds to a particular i-bit prefix. Specifically, if we denote the known prefixes in lexicographical order by $(p_1, p_2, \ldots, p_{k_i})$, then bit $2j$ in the new string will represent the prefix $p_j\mathbf{0}$, and bit $2j+1$ will represent $p_j\mathbf{1}$. Each source constructs its string by placing a $\mathbf{1}$ in the position corresponding to its own ID's i-bit prefix, and $\mathbf{0}$ in all others. We will denote the string constructed in this manner by source s in round i by $Z_{s,i}$.

Performing CollectMessages with these strings ensures that the leader receives the **OR**-superimposition, which informs it of all i-bit prefixes of source IDs (since it is aware of which prefix each position corresponds to). It then broadcasts this back out to the network via the standard beep wave procedure, and thus the invariant is fulfilled round i. After $\log L$ rounds, the IDs of all sources are known in entirety by all nodes. We then perform one final CollectMessages procedure, this time to collate all of the messages the sources wish to broadcast to the network. We construct a $k \log M$-bit string in which the j^{th} block of $\log M$ bits corresponds to the message of the j^{th} source (in lexicographical order of ID). Each source individually fills in its own message in the appropriate block, leaving all other bits as $\mathbf{0}$. We denote the string constructed in this manner by source s as $\tilde{m}_s$. Performing CollectMessages on these strings ensures that the full string of messages arrives at the leader, who then broadcasts it back out to the network.

Theorem 8. Multi-Broadcast With Provenance$(S, m(S))$ *correctly performs multi-broadcast with provenance within* $O(k \log \frac{LM}{k} + D \log L)$ *time-steps.*

Proof. The three sub-procedure calls in the initial 'set-up' phase take a total of

Algorithm 6 MULTI-BROADCAST WITH PROVENANCE$(S, m(S))$

$v \leftarrow$ ELECTLEADER
$D \leftarrow$ ESTIMATEDIAMETER(v)
$\log M \leftarrow$ GETMESSAGELENGTH$(v, S, m(S))$
for $i = 1$ to $\log L$ **do**
  $Z_i \leftarrow$ COLLECTMESSAGES$(v, S, Z_{S,i}, 2k_i)$
  perform BEEP-WAVE(v, Z_i)
end for
$\tilde{m} \leftarrow$ COLLECTMESSAGES$(v, S, \tilde{m}_S, k \cdot \log M)$
perform BEEP-WAVE$(v, \tilde{m})$

$O(D \log L + \log M)$ time-steps, and provide a leader node and knowledge of D and $\log M$.

Round i of the main loop of the algorithm takes $O(D + k_i)$ time, since it consists of performing COLLECTMESSAGES on strings of length $O(k_i)$, and then BEEP-WAVE on a string of the same length. Furthermore, since the number of known prefixes at most doubles each round, $k_i \leq 2^{i-1}$. Hence, there exists some constant c such that total time for the loop is bounded by:

$$\begin{aligned}\sum_{i=1}^{\log L} c(D + k_i) &= cD \log L + c\left(\sum_{i=1}^{\log k} k_i + \sum_{i=\log k+1}^{\log L} k_i\right) \\ &\leq cD \log L + c\left(\sum_{i=1}^{\log k} 2^{i-1} + \sum_{i=\log k+1}^{\log L} k\right) \\ &\leq cD \log L + c(k + k(\log L - \log k)) = O(D \log L + k \log \frac{L}{k}) \ .\end{aligned}$$

The final call to COLLECTMESSAGES then takes a further $O(D + k \log M)$ time, and so total running time is $O(D \log L + k \log \frac{L}{k} + k \log M) = O(k \log \frac{LM}{k} + D \log L)$

Correctness follows since each round of the loop informs the leader of the next bit in each ID prefix, and it then broadcasts this information to the network. After $\log L$ rounds, all nodes know all source IDs and each source s can correctly construct its string $\tilde{m}_s$. The **OR**-superimposition of these strings, broadcast to all nodes, is a list of messages in source ID order, which fulfills the goal of the algorithm. □

Multi-Broadcast Without Provenance

It may be the case that we do not need to know where messages originated from, or the number of duplicate messages; for example when using short control messages instructing all nodes to perform some action, for which provenance might be irrelevant. For this reason, we also study the variant of multi-broadcast where nodes need only know one copy of each unique source message, and no source IDs.

The main difference in concept for our multi-broadcast without provenance algorithm (Algorithm 7) is that the concurrent binary searches are performed on the bits of the source messages rather than the IDs. However, this requires $O(D \log M)$ time, which is too slow when $k < D$ and $L < M$, and so we first run Algorithm 6, curtailing it when our number k_i of known ID prefixes (which is a lower bound for k) exceeds D, in order to efficiently deal with these cases.

If $k \leq D$ then the call to algorithm 6 will complete multi-broadcast (meeting the requirements for the case without provenance, since they are strictly weaker than those with provenance). Otherwise, we move onto performing binary searches on the bits of the message. This functions in much the same way as in Algorithm 6, except that we do not need the final CollectMessages and Beep-Wave stage since the network is already aware of all source messages upon completion of the main loop. We will use $\tilde{k}_i$ to denote the number of $i-1$-bit message prefixes known to nodes at the start of round i of the for loop, and $\tilde{Z}_{s,i}$ to be the string constructed by source s in round i by placing a **1** in the position corresponding to the i-bit prefix of its message and **0** in all others.

Algorithm 7 Multi-Broadcast Without Provenance$(S, m(S))$

```
perform Multi-Broadcast With Provenance(S, m(S)) until k_i > D
if it did not complete then
    for i = 1 to log M do
        Z̃_i ← CollectMessages(v, S, Z̃_{S,i}, 2k̃_i)
        perform Beep-Wave(v, Z̃_i)
    end for
end if
```

Theorem 9. Multi-Broadcast Without Provenance$(S, m(S))$ *correctly performs multi-broadcast without provenance within* $O(k \log \frac{M}{k} + D \log L)$ *time-steps if* $k < M$*, and* $O(M + D \log L)$ *time-steps if* $k \geq M$*.*

Proof. By the same argument as for Theorem 6, each round of the main loop informs all nodes of the next bit in each message prefix. Therefore, after $\log M$ rounds we have performed multi-broadcast without provenance.

We separate the running-time proof into four cases:

(1) $k \leq D$ and $k < M$;

(2) $k \leq D$ and $k \geq M$;

(3) $k > D$ and $k < M$;

(4) $k > D$ and $k \geq M$.

Case 1: $k \leq D$ and $k < M$. For the $k \leq D$ case, the number of unique i-bit source ID prefixes k_i will never exceed D (since it is bounded above by k), and so the all to Multi-Broadcast With Provenance will successfully perform multi-broadcast (with provenance, and therefore also without) in $O(k \log \frac{LM}{k} + D \log L) = O(k \log L + k \log \frac{M}{k} + D \log L) = O(k \log \frac{M}{k} + D \log L)$ time-steps.

Case 2: $k \leq D$ and $k \geq M$. As above, the call to Multi-Broadcast With Provenance will successfully perform multi-broadcast in $O(k \log \frac{LM}{k} + D \log L) = O(k \log L + D \log L) = O(D \log L)$ time-steps.

Case 3: $k > D$ and $k < M$. Since $k > D$, the call will not complete multi-broadcast, but its "set-up" phase will provide a leader v and knowledge of D and $\log M$, so these steps are not duplicated in our description of Algorithm 7. Each round of the main loop then informs every node of the next bit in each unique message prefix, and so after $\log M$ rounds we are done.

Let t be the round of the loop at which the call to Multi-Broadcast With Provenance terminates. Running time for the call is then bounded above (for some constant c) by

$$\begin{aligned} cD \log L + \sum_{i=1}^{t} c(D + k_i) &\leq cD \log L + \sum_{i=1}^{t} 2cD \\ &\leq cD \log L + \sum_{i=1}^{\log L} 2cD \\ &= 3cD \log L = O(D \log L) \ , \end{aligned}$$

where the first inequality is due to the fact that $k_i \leq D$ until termination.

Running time for the main loop of Algorithm 7 is bounded above (again for some constant c) by:

$$\begin{aligned}\sum_{i=1}^{\log M} c(D+\tilde{k}_i) &= cD\log M + c\left(\sum_{i=1}^{\log k}\tilde{k}_i + \sum_{i=\log k+1}^{\log M}\tilde{k}_i\right)\\ &\leq cD\log M + c\left(\sum_{i=1}^{\log k} 2^{i-1} + \sum_{i=\log k+1}^{\log M} k\right)\\ &\leq cD\log M + c(k + k(\log M - \log k))\\ &= O(D\log M + k\log\frac{M}{k}) \ .\end{aligned}$$

Total time is therefore

$$\begin{aligned}O(D\log L + D\log M + k\log\frac{M}{k}) &= O(D\log L + D\log\frac{M}{k} + D\log k + k\log\frac{M}{k})\\ &= O(k\log\frac{M}{k} + D\log L) \ ,\end{aligned}$$

where the last expression holds since $D\log k \leq D\log L$ and $D\log\frac{M}{k} \leq k\log\frac{M}{k}$.

Case 4: $k > D$ and $k \geq M$. The call to Multi-Broadcast With Provenance will fail and take $O(D\log L)$ time as before. Running time for the main loop of Algorithm 7 is now bounded by:

$$\begin{aligned}\sum_{i=1}^{\log M} c(D+\tilde{k}_i) &= cD\log M + c\sum_{i=1}^{\log M}\tilde{k}_i \leq cD\log M + c\sum_{i=1}^{\log M} 2^{i-1}\\ &\leq cD\log M + cM = O(D\log M + M) \ .\end{aligned}$$

Since $M \leq k \leq L$, total running time is $O(M + D\log L)$.

Combining cases: When $M > k$ total running time is $O(k\log\frac{M}{k} + D\log L)$, and when $M \leq k$, total running time is $O(M + D\log L)$. □

It may seem nonintuitive that Algorithm 7 achieves multi-broadcast in fewer then the $k\log M$ time-steps required for a single node to directly transmit or

hear the messages, since this might seem to be a natural lower bound. The improvement stems from implicit compression of the messages within the algorithm's method.

3.4.3 Faster Non-Explicit Multi-Broadcast

A very recent result by Dufoulon, Burman, and Beauquier [24] improves the running time for leader election in the beep model to an optimal $O(D+\log L)$. This allows them to slightly improve the running time of our explicit multi-broadcast with provenance algorithm (Algorithm 6) to $O(k \log \frac{LM}{k} + D \min\{k, \log L\})$. However, it does not directly lead to significantly faster algorithms, because leader election was not a bottleneck in our analysis. In this section, we show how to exploit this improved leader election procedure to attain an optimal algorithm for multi-broadcast with provenance, and near-optimal for multi-broadcasting without provenance.

We take a different approach from Algorithms 6 and 7, using a new type of *superimposed code* to collect information. A superimposed code is a function which maps each element of its domain to a unique binary codeword, in such a way that information can be inferred from the binary **OR**-superimposition of a set of codewords. In our case, we define a (k, X)-*choice superimposed code* which guarantees that, given the superimposition of any k codewords, there are at most $O(k)$ codewords that could have been included in the superimposition (because all of the others have a **1** where the superimposition has a **0**).

We will say one binary string a is *dominated* by another string b (denoted $a \preceq b$) if $a_i \leq b_i \forall i$. Our goal now is to show that any superimposition of k codewords dominates at most $O(k)$ others.

Definition 10. *A (k, X)-**choice superimposed code of length** ℓ is an injective function $C : X \to \{0,1\}^\ell$ such that for every set $K \subseteq X$ with $|K| := k$, the size of the set $Y = \{u \in X : C(u) \preceq \bigvee_{v \in K} C(v)\}$ is at most $9k$.*

This set Y is the set of all codewords dominated by the superimposition. Note that while the definition specifies superimpositions of exactly k codewords, the set Y is also size $O(k)$ for any superimposition of fewer than k codewords, since these can be arbitrarily extended to k codewords without reducing the size of Y.

Lemma 11. *For any k, X with $k \leq |X|$, there exists a (k, X)-choice superimposed code of length $9k \ln \frac{|X|}{k}$.*

Proof. This is the first example we will see of a proof of existence by the probabilistic method, which will be very useful to us later in Chapter 4. The idea is that we prove the existence of some combinatorial object by randomly generating a candidate object, and then proving that it satisfies the required criteria with positive probability. Then, some such object must exist. The downside of this type of argument is that it is existential, i.e. does not tell us how to construct the object, and so algorithms making use of the object are non-explicit.

Let $x = |X|$. We randomly generate a candidate code $C : X \to \{0,1\}^\ell$, by choosing each bit of each code-word independently to be:

- $\mathbf{1}$ with probability $\frac{1}{2k}$ and $\mathbf{0}$ otherwise, for the first $6k \ln \frac{x}{k}$ bits.

- $\mathbf{1}$ with probability $\frac{1}{2}$ and $\mathbf{0}$ otherwise, for the last $3k \ln \frac{x}{k}$ bits.

The last $3k \ln \frac{x}{k} \geq 3 \ln x$ bits are solely to ensure that no two code-words are the same (i.e. C is injective as required), which is the case since the probability that two particular codewords agree on those bits is at most $(\frac{1}{2})^{3 \ln x} \leq x^{-2.07}$. Taking a union bound over all $\binom{x}{2} \leq \frac{1}{2}x^2$ pairs of codewords, the probability that any two are the same is at most $\frac{1}{2}x^2 \cdot x^{-2.07} \leq \frac{1}{2}$.

For the rest of our analysis we consider only the first $6k \ln \frac{x}{k}$ bits. Fix some subset $K \subseteq X$ of size k. Clearly for all $u \in K$, $C(u) \preceq \bigvee_{v \in K} C(v)$. The probability that any particular bit $\bigvee_{v \in K} C(v)_i$ is $\mathbf{0}$ is at least

$$\prod_{v \in K} \mathbf{Pr}\left[C(v)_i = \mathbf{0}\right] \geq \prod_{v \in K} (1 - \frac{1}{2k}) \geq \prod_{v \in K} 4^{-\frac{1}{2k}} = 4^{-\frac{k}{2k}} = \frac{1}{2} \ .$$

We can then show that any codeword not in K is unlikely to be dominated by K's superimposition. For any $u \notin K$, the probability that $C(u) \preceq \bigvee_{v \in K} C(v)$ is at most

$$\begin{aligned}\prod_{i \in [\ell]} \mathbf{Pr}\left[\neg\left(C(u)_i = \mathbf{1} \wedge \bigvee_{v \in K} C(v)_i = \mathbf{0}\right)\right] &\leq \prod_{i \in [\ell]} (1 - (\frac{1}{2k} \cdot \frac{1}{2})) \\ &\leq \prod_{i \in [\ell]} e^{\frac{-1}{4k}} = e^{\frac{-\ell}{4k}}\end{aligned}$$

So, the probability that $|Y \setminus K| \geq 8k$ (i.e. $|Y| \geq 9k$) is at most:

$$\binom{X}{8k} (e^{\frac{-\ell}{4k}})^{8k} \leq \left(\frac{ex}{8k}\right)^{8k} e^{-2\ell} \leq e^{8k \ln \frac{x}{k} - 2\ell} \leq e^{-4k \ln \frac{x}{k}}$$

There are at most $\binom{x}{k} \leq e^{2 \ln \frac{x}{k}}$ possible sets K, and by a union bound over all of them, the probability some set K does not satisfy the condition is at most $e^{-2k \ln \frac{x}{k}}$. By another union bound, the probability that the codewords are not unique or the condition is not satisfied is at most $e^{-2k \ln \frac{x}{k}} + \frac{1}{2} < 1$. Since there is a non-zero probability that C is a valid (k, X)-choice superimposed code, such a code must exist. □

We now describe how choice superimposed codes can be used for multi-broadcast, in Algorithm 8.

We perform the same 'set-up' phase as in Algorithms 6 and 7, electing a leader and obtaining knowledge of diameter D and message length $\log M$. Using the leader election algorithm of [24], though, this only requires $O(D + \log L + \log M)$ time.

Next, we repeatedly perform *rounds* in which we attempt to collect the source messages, encoded using choice superimposed codes. The rounds have a parameter j which doubles each time, starting at a value such that $j \log \frac{M}{j} = D$ (since the rounds will have running time $\Theta(D + j \log \frac{M}{j})$, and we wish to start with the two factors equal). For each round, let C_j be a $(j, [M])$-choice superimposed code of length $9j \ln \frac{M}{j}$. We perform CollectMessages$(v, S, C_j(m(S)), 9j \ln \frac{M}{j})$ and broadcast the resulting string using Beep-Wave. By the properties of choice superimposed codes, if we have $j \geq k$, then the size of the set Y of dominated codewords is at most $9j$.

When this is the case, we proceed to a final call of CollectMessages. Each source node creates a string $\tilde{m}_S$ of length $|Y|$, where the b^{th} bit of the string corresponds to the b^{th} codeword in Y (in lexicographical order). It sets the bit corresponding to the codeword it used for its message to **1**, and all others to **0**. CollectMessages, performed on these strings and re-broadcast, then informs all nodes of the codewords (and hence the source messages) in use.

Theorem 12. *Algorithm 8 correctly performs multi-broadcast without provenance within $O(k \log \frac{M}{k} + D + \log L)$ time-steps if $k < M$, and $O(M + D + \log L)$ time-steps if $k \geq M$.*

Proof. The three sub-procedure calls in initial "set-up" phase take a total of $O(D + \log L + \log M)$ time-steps, and provide a leader node and knowledge of D and $\log M$.

A round of the algorithm's loop with parameter j takes $O(D + j \log \frac{M}{j})$ time, since it consists of performing CollectMessages on strings of length

Algorithm 8 NON-EXPLICIT MULTI-BROADCAST$(S, m(S))$

```
v ← ELECTLEADER
D ← ESTIMATEDIAMETER(v)
log M ← GETMESSAGELENGTH(v, S, m(S))
Let j satisfy j log M/j = D
repeat
    Z_j ← COLLECTMESSAGES(v, S, C_j(m(S)), 9j ln M/j)
    perform BEEP-WAVE(v, Z_j)
    i ← |Y|
    j ← 2j
until i ≤ 9j
m̃ ← COLLECTMESSAGES(v, S, m̃_S, i)
perform BEEP-WAVE(v, m̃)
```

$O(j \log \frac{M}{j})$, and then BEEP-WAVE on a string of the same length. The loop terminates when $j \leq k$ (assuming $k \leq \frac{M}{2}$; if $k \geq \frac{M}{2}$ it terminates when $j \leq \frac{M}{2}$, since $|Y| \leq M$).

Let j' be the initial value of j, i.e. $j' \log \frac{M}{j'} = D$.

We analyze running time of the loop and final COLLECTMESSAGES call, separating into three cases:

(1) $j' \geq \min k, \frac{M}{2}$;

(2) $j' \leq k < \frac{M}{2}$;

(3) $j' \leq \frac{M}{2} \leq k$;

Case 1: $j' \geq \min k, \frac{M}{2}$**.** The loop terminates after the first round, taking $O(j' \log \frac{M}{j'}) = O(D)$ time. The final COLLECTMESSAGES call takes $O(j') = O(D)$ time.

Case 2: $j' \leq k < \frac{M}{2}$**.** Total running time of the loop is at most

$$
\begin{aligned}
c \sum_{q=\log j'}^{\log k} \left(D + 2^q \log \frac{M}{2^q} \right) &\leq c(D \log \frac{k}{j'} + \sum_{q=1}^{\log k} (2^q \log M - q2^q)) \\
&\leq c(j' \log \frac{M}{j'} \log \frac{k}{j'} + 2^{\log k+1} \log M - (\log k - 1)2^{\log k+1}) \\
&\leq c(k \log \frac{M}{k} + \log \frac{2M}{k} 2^{\log k+1}) \\
&\leq 5ck \log \frac{M}{k} \ ,
\end{aligned}
$$

for some constant c.

The final CollectMessages call takes $O(k)$ time.

Case 3: $j' \leq \frac{M}{2} \leq k$**.** Total running time of the loop is at most

$$\begin{aligned} c \sum_{q=\log j'}^{\log \frac{M}{2}} \left(D + 2^q \log \frac{M}{2^q} \right) &\leq c(D \log \frac{M}{2j'} + \sum_{q=1}^{\log \frac{M}{2}} (2^q \log M - q2^q)) \\ &\leq c(j' \log \frac{M}{j'} \log \frac{M}{2j'} + 2^{\log M} \log M - (\log M - 2)2^{\log M}) \\ &\leq c(\frac{M}{2} + 2M) \\ &\leq 3cM \ , \end{aligned}$$

for some constant c.

The final CollectMessages call takes $O(M)$ time.

Combining these cases, we can see that Algorithm 8 performs multi-broadcast without provenance within $O(k \log \frac{M}{k} + D + \log L)$ time-steps if $k < \frac{M}{2}$, and $O(M + D + \log L)$ time-steps if $k \geq \frac{M}{2}$. □

We can use the same algorithm to perform multi-broadcast with provenance, simply by having each node v append its ID to its source message $m(v)$. Then, messages are drawn from the set $[L] \times [M]$ of size LM. Replacing M by LM in the statement and proof of Theorem 12 gives the following:

Theorem 13. *Algorithm 8 correctly performs multi-broadcast with provenance within $O(k \log \frac{LM}{k} + D)$ time-steps.*

3.5 Lower Bounds

In this section we give lower bounds for the main communications tasks we have considered: broadcasting (in undirected and directed networks) and multi-broadcast. All of these lower bounds follow a similar approach: given n, D, L, M, and for multi-broadcasting k, we first fix a network $\mathfrak{N}$ with n nodes and diameter D. Then, we specify a distribution of input instances by choosing uniformly at random the identifier assignment ID (from the set of all injective functions $[n] \to [L]$). Since we prove lower bounds against randomized algorithms, we will also assume that nodes have as input a random string y drawn independently from some distribution Y. Finally, we choose the input

message(s) m at random. The distribution we will choose from depends upon the task for which we give a lower bound:

- For broadcasting we choose a single message m uniformly at random from $[M]$;
- For multi-broadcasting without provenance we uniformly choose a size-k subset of messages from $[M]$, which we will denote by $m \in \binom{[M]}{k}$;
- For multi-broadcasting with provenance we instead draw m from the product of k independent (possibly non-unique) messages with a size-k subset of IDs, i.e. $m \in [M]^k \times \binom{[L]}{k}$. Source nodes use these IDs rather than those specified previously.

To encode node behavior, we will denote by $P_t^v \in \{\mathbf{B}, \mathbf{L}\}$ the behavior of a node v at a time-step t, where $\mathbf{B}$ means that v beeps, and $\mathbf{L}$ that it listens. We further denote $P_{\leq t}^v$ the sequence of v's behavior up to time-step t. We will also need to model what v would hear upon listening, which we denote $Q_t^v \in \{\mathbf{H}, \mathbf{S}\}$, where $\mathbf{H}$ means that v would hear a beep (i.e. has a neighbor u with $P_t^u = \mathbf{B}$), and $\mathbf{S}$ that it would hear silence. Likewise, we denote $Q_{\leq t}^v := \{Q_{t'}^v\}_{t' \leq t}$.

We then note that v's output after time-step t must depend entirely on $ID(v)$, y, $Q_{\leq t}^v$, and if v is a source, m_v. Our goal now will be to show that if insufficient time has passed, the probability that a node v's output is correct will be $o(1)$. We do this by arguing that $ID(v)$ and y are independent of v's correct output, and that $Q_{\leq t}^v$ provides insufficient information to reliably recover this output.

3.5.1 Undirected Networks

We will first show lower bound for broadcasting and multi-broadcast in undirected networks. The bound for broadcasting will be derived as a special case of the multi-broadcasting bound, so we begin with multi-broadcast without provenance.

The network $\mathfrak{N}$ we will use as a lower bound is the following: we place one node in each layer 1 to D, and all other nodes in layer 0. An edge will be present between nodes u and v if they are in consecutive layers, i.e. $|layer(u) - layer(v)| = 1$. (Note that we assume here that $n - D \geq k$, but since we are concerned with asymptotic results, if this is not the case we can simply use $k' = \frac{k}{2}$ and $D' = \frac{D}{2}$ instead.)

As described above, we choose uniformly at random $y \in Y$, $ID \in [n] \to [L]$, and $m \in \binom{[M]}{k}$ to generate our input distribution.

We show a lemma which states effectively that a node further than t steps from the source nodes cannot receive any information about source messages before time-step t:

Lemma 14. *For a time-step t, and for a node v in layer i with $i > t$, $P^v_{\leq t}$ is independent of m.*

Proof. We prove the claim by induction on t. Trivially it is true when $t = 1$, since any node v in layer $i > 0$ is not a source, and P^v_0 is determined based only on $ID(v)$ and y, which are independent of m.

Assuming the claim is true for $t = j$, and proving for $t = j + 1$, for a node v in layer $i > j + 1$, $P^v_{\leq j+1}$ is dependent entirely on $ID(v)$, y, and $Q^v_{\leq j}$. $Q^v_{\leq j}$ is dependent only on the values $P^u_{\leq j}$ for neighbors u of v, and since these nodes are in layers at least $i - 1 > j$, these $P^u_{\leq j}$ values are also independent of m by the inductive assumption. □

An easy corollary gives an $\Omega(D)$ lower bound:

Corollary 15. *Any multi-broadcast algorithm running on $\mathfrak{N}$ has $o(1)$ success probability, conditioned on it terminating in fewer than $T < D - 1$ time-steps. Asymptotic behavior refers to when $M \to \infty$.*

Proof. Consider a node v in layer D. The output of v after time-step T must depend entirely on $ID(v)$, y, and $Q^v_{\leq T}$. $Q^v_{\leq T}$ depends only on $P^u_{\leq T}$ for neighbors u of v, and since these nodes are in layers at least $D-1 > T$, by Lemma 16 this is independent of m. So, since m is chosen uniformly from $\binom{[M]}{k}$ independently of v's output, the probability that the output is correct is at most $\binom{M}{k}^{-1}$. □

We now show the $\Omega(k \log \frac{M}{k})$ term of the lower bound by arguing that if an algorithm terminates faster than this, $Q_{\leq T}$ contains insufficient information to correctly recover m:

Lemma 16. *Any multi-broadcast algorithm running on $\mathfrak{N}$ has $o(1)$ success probability, conditioned on it terminating in $T \leq \frac{k}{2} \log \frac{M}{k}$ time-steps.*

Proof. The output of any non-source node v at time-step T must depend entirely on $ID(v)$, y, and $Q_{\leq T}$. $ID(v)$ and y are independent of m, and $Q_{\leq T}$ takes one of only $2^T \leq \left(\frac{M}{k}\right)^{\frac{k}{2}}$ values.

For each m, let q_m maximize $\mathbf{Pr}\left[OUTPUT_v = m | Q_{\leq T} = q_m\right]$. Note that $\sum_{m\in[M]} \mathbf{Pr}\left[OUTPUT_v = m | Q_{\leq T} = q_m\right] \leq \frac{M}{k}^{\frac{k}{2}}$, since each possible value of $Q_{\leq T}$ contributes at most 1 in total. Then,

$$\begin{aligned}
\mathbf{Pr}\left[OUTPUT_v \text{ is correct}\right] &= \binom{M}{k}^{-1} \sum_{\mathbf{m}\in\binom{[M]}{k}} \mathbf{Pr}\left[OUTPUT_v = \mathbf{m} | m = \mathbf{m}\right] \\
&\leq \binom{M}{k}^{-1} \sum_{\mathbf{m}\in\binom{[M]}{k}} \mathbf{Pr}\left[OUTPUT_v = \mathbf{m} | m = \mathbf{m}, Q_{\leq T} = q_{\mathbf{m}}\right] \\
&\leq \binom{M}{k}^{-1} \left(\frac{M}{k}\right)^{\frac{k}{2}} \leq \left(\frac{M}{k}\right)^{-k} \left(\frac{M}{k}\right)^{\frac{k}{2}} = \left(\frac{M}{k}\right)^{-\frac{k}{2}} .
\end{aligned}$$

□

Since these results were proven on the same input distribution, we can combine them:

Theorem 17. *For $k < \frac{M}{2}$, any multi-broadcast without provenance algorithm running on $\mathfrak{N}$ has $o(1)$ success probability, conditioned on it terminating within $\frac{1}{4}(D + k \log \frac{M}{k})$ time-steps.*

Proof. From Corollary 15 and Lemma 16, an algorithm has $o(1)$ success probability conditioned on it terminating within $\max\{D - 1, \frac{k}{2} \log \frac{M}{k}\} \geq \frac{1}{4}(D + k \log \frac{M}{k})$ time-steps. □

With slight adjustments, we can also obtain a lower bound when $k \geq \frac{M}{2}$:

Theorem 18. *For $k \geq \frac{M}{2}$, any multi-broadcast without provenance algorithm running on $\mathfrak{N}$ has $o(1)$ success probability, conditioned on it terminating within $\frac{1}{8}(D + M)$ time-steps.*

Proof. We follow the same lines as the proof of Theorem 17, but when specifying our input distribution we only randomly select messages for $k' = \frac{M}{2}$ of the sources (the rest we can choose arbitrarily, and in fact can assume are known by all nodes a priori). Then, we reach the same result as Theorem 17 for for algorithms terminating within $\frac{1}{4}(D + k' \log \frac{M}{k'}) = \frac{D}{4} + \frac{M}{8}$ time-steps. □

We can also easily adapt for multi-broadcast with provenance:

Theorem 19. *Any multi-broadcast with provenance algorithm running on* $\mathfrak{N}$ *has* $o(1)$ *success probability, conditioned on it terminating within* $\frac{1}{4}(D+k\log\frac{LM}{k})$ *time-steps.*

Proof. We again follow the same lines as the proof of Theorem 17, but when specifying our input distribution we now take source inputs to the product of non-unique messages and unique IDs, i.e. m is drawn uniformly at random from the set $[M]^k \times \binom{[L]}{k}$, which has size $M^k \cdot \binom{L}{k} \geq \left(\frac{LM}{k}\right)^k$. We can then show analogously that conditioning on termination within $\frac{k}{2}\log\frac{LM}{k}$ time-steps, probability of correct output is at most $\left(\frac{LM}{k}\right)^{-\frac{k}{2}}$. The proof that $D-1$ time-steps are required is unchanged. So, an algorithm has $o(1)$ success probability conditioned on it terminating within $\max\{D-1, \frac{k}{2}\log\frac{LM}{k}\} \geq \frac{1}{4}(D+k\log\frac{LM}{k})$ time-steps. □

An asymptotically optimal lower bound for broadcasting is a special case of Theorem 17:

Theorem 20. *Any broadcasting algorithm running on* $\mathfrak{N}$ *has* $o(1)$ *success probability, conditioned on it terminating within* $\frac{1}{4}(D+\log M)$ *time-steps.*

Proof. Follows from Theorem 17, by setting $k=1$. □

3.5.2 Directed Networks

Next, we show that our algorithm for broadcasting in the directed beep model is also optimal.

Our network $\mathfrak{N}$ will be as follows, given parameters n and D, we again divide the nodes into $D+1$ *layers*; this time layer 0 contains only the source node s, layers 1 to $D-1$ each contain a single non-source node, and layer D contains all other nodes. Then we let a *directed* edge (u,v) be present in the network if $layer(u) \geq layer(v)-1$, with the exception that we do not put edges between pairs of nodes in layer D. That is, a node has edges to the node in the next layer, and to all nodes in previous layers.

Consider a fixed broadcasting algorithm running on $\mathfrak{N}$ for t time-steps. We will denote variable sets X_t^i to be the set of time-steps at most t in which a node in layer i beeps and no nodes in later layers do. We now show, in effect, that all information a node receives about the source message is contained within these sets:

Lemma 21. *For time-step t, a layer i, and for any node v in layer $j > i$, $P^v_{\leq t}$ is dependent entirely on ID, y, and X^i_{t-1}.*

Proof. We prove the claim by induction on t. The base case $t = 0$ is trivially true, since all non-source nodes' input in the time-step 0 is included in ID and y, and so their choice to beep is also fully dependent on these.

For the inductive step, assuming the claim is true for $t < t'$, we prove for t'. $P^v_{\leq t}$ is dependent entirely on ID, y, and $Q^v_{\leq t'-1}$. This latter term is dependent on the value of $P^u_{\leq t'-1}$ for all in-neighbors u of v.

If $i < j - 1$, these values are dependent entirely on ID, y, and $X^i_{t'-2}$ by the inductive assumption. All information in $X^i_{t'-2}$ is contained in $X^i_{t'-1}$, so the claim holds.

If $i = j - 1$, however, the inductive assumption cannot be applied to $P^w_{\leq t'-1}$, where w is the in-neighbor of v in layer $j - 1$. In this case, $P^w_{\leq t'-1}$ can be determined entirely from $X^i_{t'-1}$ and the values $P^u_{\leq t'-1}$ for all nodes u in layers at least j. By the inductive assumption, these values $P^u_{\leq t'-1}$, are dependent entirely upon ID, y, and $X^i_{t'-1}$, hence so are $P^w_{\leq t'-1}$ and $P^v_{\leq t'}$. □

We can node prove our lower bound by arguing that if running time is too short, these sets X^i_t contain insufficient information to recover m:

Theorem 22. *Any algorithm for broadcasting in directed networks which runs in $o(\frac{D \log M}{\log D})$ expected time has $o(1)$ success probability.*

Proof. We consider a node v in layer D. Let $c \geq 3$ be an arbitrarily large constant. By Lemma 21, the output of such a node v after $T \leq \frac{D \log M}{c^2 \log D}$ time-steps can be expressed as a function of ID, y (both of which independent of m), and X^i_{T-1}, for any $i < D$. We will denote random variable $x^i = |X^i_{T-1}|$.

For each $m \in [M]$, let $Xmax^i_{\mathbf{m}}$ be the value of X^i_{T-1} which maximizes

$$\mathbf{Pr}\left[OUTPUT_v = \mathbf{m} | X^i_{T-1} = Xmax^i_{\mathbf{m}}, m = \mathbf{m}\right]$$

subject to $x^i \leq \frac{\log M}{c \log D}$.

There are at most

$$\sum_{x^i=1}^{\frac{\log M}{c \log D}} \binom{T}{x^i} \leq \frac{\log M}{c \log D} \left(\frac{Te}{\frac{\log M}{c \log D}}\right)^{\frac{\log M}{c \log D}} = \frac{\log M}{c \log D} \left(\frac{eD}{c}\right)^{\frac{\log M}{c \log D}} \leq 2^{\frac{\log M}{c}} = M^{\frac{1}{c}}$$

possible values of X^i_{t-1} with $x^i \leq \frac{\log M}{c \log D}$. Therefore,

$$\sum_{\mathbf{m} \in [M]} \mathbf{Pr}\left[OUTPUT_v = \mathbf{m} | X^i_{T-1} = Xmax^i_{\mathbf{m}}, m = \mathbf{m}, x^i \leq \frac{\log M}{c \log D}\right] \leq M^{\frac{1}{c}} \ ,$$

since each possible value of X^i_{T-1} contributes at most 1 in total to the sum.

Then, for any i,

$$\begin{aligned}
&\mathbf{Pr}\left[OUTPUT_v \text{ is correct} | x^i \leq \frac{\log M}{c \log D}\right] \\
&\quad \leq \frac{1}{M} \sum_{\mathbf{m} \in [M]} \mathbf{Pr}\left[OUTPUT_v = \mathbf{m} | m = \mathbf{m}, x^i \leq \frac{\log M}{c \log D}\right] \\
&\quad \leq \frac{1}{M} \sum_{\mathbf{m} \in [M]} \mathbf{Pr}\left[OUTPUT_v = \mathbf{m} | X^i_{T-1} = Xmax^i_{\mathbf{m}}, m = \mathbf{m}, x^i \leq \frac{\log M}{c \log D}\right] \\
&\quad \leq \frac{M^{\frac{1}{c}}}{M} = M^{\frac{1}{c}-1} \ .
\end{aligned}$$

Now assume for the sake of contradiction that a broadcasting algorithm finishes within expected time $\mathbf{E}[T] \leq \frac{D \log M}{c^2 \log D}$ and succeeds with probability $p \geq \frac{2}{c}$. Then, for all $i < D$ we must have that $\mathbf{Pr}\left[x^i > \frac{\log M}{c \log D}\right] \geq p - M^{\frac{1}{c}-1} > \frac{p}{2}$, and so $\mathbf{E}\left[x^i\right] > \frac{p \log M}{2c \log D}$. Since the sets X^i_t are disjoint, $\sum_{i<D} x^i \leq T$, and therefore

$$\mathbf{E}[T] > \frac{pD \log M}{2c \log D} = \frac{D \log M}{c^2 \log D} \ .$$

This is a contradiction, and so it must be the case that $\mathbf{E}[T] > \frac{D \log M}{c^2 \log D}$. Since c is arbitrarily large, we have shown that any algorithm with $\Omega(1)$ success probability must take $\Omega(\frac{D \log M}{\log D})$ expected time, and conversely, any algorithm taking $o(\frac{D \log M}{\log D})$ expected time has $o(1)$ success probability. □

3.6 Discussion and Open Problems

Models for networks of very weak devices, such as the beep model, are growing in popularity as such devices become cheaper and more commercially viable; examples of their use include sensor networks and RFID tagging. Our aim here is to provide the first systematic study of algorithms for global tasks in such a model. Our running times are mostly optimal, with the only major grounds for improvement being an optimal *explicit* multi-broadcast algorithm. However,

there are several other interesting aspects of the beep model which could merit further research.

One crucial concern in networks of this type is that *energy* is often highly constrained: we may wish to minimize the amount of times nodes transmit (and possibly even listen; we could introduce a third option of 'do nothing' in a time-step). A study of how little energy is required to complete communication tasks in the beep model would be interesting.

Another research direction is further weaken the assumptions of the model, in order to make it as widely applicable as possible. The major assumption remaining is that time-steps are *synchronous*, i.e. that nodes local clocks all 'tick' at the same rate and beeps are heard immediately. There are several possible ways of modeling *asynchronicity*, and exploring what can be done in asynchronous beeping networks. One work in this direction is [37].

Chapter 4

Deterministic Radio Communication

In this chapter we present deterministic algorithms for broadcasting and wake-up, in single-hop (multiple access channel) and multi-hop radio networks, with and without parameter knowledge.

Recall that in the task of *wake-up*, nodes begin in a dormant state, and some non-empty subset of nodes spontaneously 'wake up' at arbitrary (adversarially chosen) time-steps. Nodes are also woken up if they receive messages. Nodes cannot participate (by transmitting) until they are woken up, and our goal is to ensure that eventually all nodes are awake. We assume nodes have access only to a *local clock*: they can count the number of time-steps since they woke up, but there is no global awareness of an absolute time-step number.

The task of *broadcasting* is usually described as follows: one node begins with a message, and it must inform all other nodes of this message via transmissions. However, to enable our results to transfer from multiple access channels (single-hop radio networks) to multi-hop radio networks, in this chapter we will instead use broadcasting to refer to a more generalized task. Our broadcasting task will be defined similarly to wake-up, with the only difference being that nodes have access to a *global clock*, informing them of the absolute time-step number. (In multiple access channels, this task is usually also referred to as wake-up, specifying global clock access, but here we will call it broadcasting to better differentiate.)

Notice that the standard broadcasting task in radio networks is a special case

of this task, in which only one node spontaneously wakes up. A global clock can be simulated by appending the number of the current global time-step to each transmitted message (and since all message chains originate from the same source node, these time-step numbers will agree).

4.1 Related Work

We survey the history of the study of deterministic broadcasting and wake-up in radio networks.

4.1.1 Wake-up.

The wake-up problem (with only local clocks) has been studied in both multiple access channels and multi-hop radio networks (often separately, though the results usually transfer directly from one to the other). It has been commonly assumed in the literature that network parameters are known, and that IDs are small ($L = n^{O(1)}$).

The first sub-quadratic deterministic wake-up protocol for radio networks was given in by Chrobak et al. [13], 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 for the wake-up problem. Since then, there have been several improvements in running time, making use of the radio synchronizer machinery: firstly to $O(n^{3/2} \log n)$ [7], and then to $O(n \log^2 n)$ [6]. However, without the assumption of small labels, all of these running times are increased. The algorithm of [7] with label size as a parameter would give $O(n \log^2 L)$ time. All of these algorithms, like those we present here, are non-explicit.

There has been some work on wake-up in multiple access channels without knowledge of network parameters: firstly an $O(L^4 \log^5 L)$ algorithm [31], and then an improvement to $O(L^3 \log^3 L)$ [49]. It was believed that this algorithms in this setting were necessarily much slower than those for when parameters were known; for example, [49] states "a crucial assumption is whether the processors using the shared channel are aware of the total number n of processors sharing the channel, or some polynomially related upper bound to such number. When such number n is known, much faster algorithms are possible."

There are no direct results for wake-up in radio networks with unknown parameters, but the algorithm of [49] can be applied to give $O(nL^3 \log^3 L)$ time.

4.1.2 Broadcasting

The task of *broadcasting* has been extensively studied for various network models for many decades. For the most relevant model here, directed radio networks with unknown structure and without collision detection, the first sub-quadratic deterministic broadcasting algorithm was proposed by Chlebus et al. [10], who gave an $O(n^{11/6})$-time broadcasting algorithm. After several small improvements [11, 48], Chrobak et al. [14] 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 [43] improved this bound to obtain an algorithm of complexity $O(n \log n \log D)$ and Czumaj and Rytter [23] gave a broadcasting algorithm running in time $O(n \log^2 D)$. Finally, De Marco [47] designed an algorithm that completes broadcasting in $O(n \log n \log \log n)$ time steps. These results approached the best lower bound known, $\Omega(n \log D)$ time due to Clementi et al. [15]. Again, however, these results generally assume *small node labels* ($L = O(n)$, though some of the earlier results only require $L = O(n^c)$ for some constant c), and their running time results do not hold otherwise. The situation where node labels can be large is less well-studied, though it is easy to see that the algorithm of [14] still works, requiring $O(n \log^2 L)$ time. In undirected networks, broadcast can be performed in $O(n \log D)$ time [42], and in multiple access channels, a $O(n \log \frac{L}{n})$ time algorithm exists [15]. All of these algorithms are, like those presented here, non-explicit.

These results also *intrinsically require parameter knowledge*. Without knowledge of n, L, or D, the fastest algorithm known is the $O(L)$ time algorithm of [31] for multiple access channels. This algorithm is explicit, but has the strong added restriction that the first node wakes up at global time-step 0. It also does not transfer to multi-hop radio networks, so the best running time therein is the $O(DL^3 \log^3 L)$ given by the algorithm of [49].

4.1.3 Previous Approaches

Almost all deterministic broadcasting protocols with sub-quadratic complexity (that is, since [10]) 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 $[n] := \{1, 2, \ldots, n\}$ 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 nodes cannot alter their behavior based on the time since they were informed, which might be desirable in order to optimize running time. 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. [13] introduced the concept of *radio synchronizers*, which they used to design an efficient wake-up algorithm. These are a development of selective families which allow nodes to begin their behavior as soon as they are informed, rather than waiting for the start of the next application of the selective family. A further extension to *universal synchronizers* in [7] allowed effectiveness across all in-neighborhood sizes. The downside of this new approach is that having nodes begin immediately gives rise to a far greater number of possible starting-time scenarios that have to be accounted for during the probabilistic proof, which in turn requires the synchronizers to be larger. This is the cause of the gap between the $O(n \log^2 n)$-time synchronizer-based wake-up algorithm of [7] and the $O(n \log^2 D)$-time selective family-based broadcasting algorithm of [23].

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. [38, 8] for explicit construction of selective families).

Returning to the problem of broadcasting, a breakthrough came in 2010 with a paper by De Marco [47] which took an approach inspired by radio synchronizers but adapted to make use of a global clock. As with radio synchronizers, nodes begin their own transmission patterns immediately upon being informed, constructed with transmission probabilities that decay over time. These be-

havior patterns are collated into a transmission matrix, about which appropriate selective properties can be proven. The difference from synchronizers is that the global clock is used to coordinate behavior within short *phases*, within which all nodes' transmission probabilities decay exponentially. Because of this, the $O(n \log^2 n)$ synchronizer running time of [6] could be improved to $O(n \log n \log \log n)$, which beats the $O(n \log^2 D)$ time algorithm of [23] for networks of large diameter.

4.2 Our Results

We present three algorithms in this chapter. The first, Algorithm 11, is a broadcasting algorithm for the most traditional setting: directed multi-hop radio networks with parameter knowledge and small node labels. We achieve $O(n \log D \log \log D)$ running time, improving over the $O(n \log^2 D)$ and $O(n \log n \log \log n)$ running times of [23, 47] and coming within a log-logarithmic factor of the $\Omega(n \log D)$ lower bound.

Our approach can be seen as a compromise between classical selective families and the transmission matrix formulation of De Marco [47]. Rather than have nodes begin their behavioral patterns immediately (as in the latter), or wait (possibly up to $\Theta(n)$ time) for the start of the next selective family as in the former, we instead divide time into *blocks* of size $\tilde{\Theta}(\frac{n}{D})$, and have nodes wait until the start of the next *block* to participate. The block size is sufficiently small that total time spent waiting does not become a dominant factor, but sufficiently large that we significantly restrict the set of circumstances we must account for (which we will call *instances*), and by doing so improve running time.

The second and third algorithms are designed for *blind* networks, in which nodes have no parameter knowledge. This model presents several new difficulties, since most previous work relies heavily upon nodes being able to base their behavior on n and D. We show that even in this more restrictive setting efficient communication can be achieved: we show the existence of a broadcasting algorithm requiring time $O(n \log L \log \log n)$, and a wake-up algorithm requiring time $O(n \log L \frac{\log n}{\log \log n})$, in both single-hop and multi-hop networks.

These running times match or even improve over the previous fastest algorithms for networks with parameter knowledge: the broadcasting result matches that of De Marco [47], and the wake-up algorithm surpasses the $O(n \log^2 L)$ time

algorithm of [7].

4.3 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. Then, in Sections 4.4–4.5, we will demonstrate in detail how these combinatorial objects can be used to obtain fast algorithms for broadcasting and wake-up.

The broad purposes of these combinatorial objects are the same: we define a collection of all possible circumstances (which we call *instances*) that we need to *hit* (have intersection of size one, corresponding to a successful transmission in our algorithms) with our combinatorial object. We show that there exists objects that have this property by the probabilistic method: we randomly generate a candidate, find a lower bound on the probability that it hits any particular instance, and then take a union bound over all instances of the probability that they are not hit. We can then find that with non-zero probability all instances are hit, so an object hitting all instances must exist.

4.3.1 Selective Families

We begin with a discussion about *selective families*, whose importance in the context of broadcasting was first observed by Chlebus et al. [10]. A selective family is a family $U = \{U_1, U_2, \ldots, U_h\}$ of subsets of $[n]$ 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, we will instead use the equivalent definition of a function $S : [n] \to \{0,1\}^h$. The correspondence between these two formulations is that $S(v)_j = 1 \iff v \in U_j$.

Definition 23. *A function $S : [n] \to \{0,1\}^h$ is an (n,k)-**selective family** of size h if for any $K \subseteq [n]$ with $1 \leq |K| \leq k$, there exists j, $1 \leq j \leq h$, such that $\sum_{v \in K} S(v)_j = 1$. We say that such j* hits *K.*

Our *instances* in this case are just the subsets K.

The following standard lemma (see, e.g., [15]) posits the existence of (n,k)-selective families of size $O(k \log \frac{n}{k})$. This has been shown to be asymptotically

optimal [15].

Lemma 24 (**Small selective families**). *For any $1 \leq k \leq n$, there exists an (n, k)-selective family of size $O(k \log \frac{n}{k})$.*

Proof. We wish to randomly generate a random candidate selective family S, and maximize the probability of it hitting each instance (set K). For each $v \in [n]$, we will choose the bits of $S(v)$ independently at random, but with different probabilities: since we need to hit sets between sizes 1 and k, we must choose the probabilities of bits being 1 to be between $\Theta(1)$ and $\Theta(\frac{1}{k})$. So, letting c be a constant to be determined later, we will divide the bits into $\log k$ *ranges*, for which the i^{th} range contains $c2^i \log \frac{n}{k}$ bits which are independently 1 with probability $\frac{1}{2^i}$ and 0 otherwise. Then, the total size h of the selective family is at most $2ck \log \frac{n}{k}$.

The purpose of choosing probabilities in this way is that if, for any set K, the sum over K of the probabilities of $S(v)_j$ being 1 is close to constant, then the probability that exactly one $S(v)_j$ is 1 is also close to constant. To formalize this, we show Lemma 25, variants of which have been used in several previous works, such as [47].

Lemma 25. *Let x_v, $v \in K$, be independent $\{0, 1\}$-valued random variables with $\mathbf{Pr}[x_v = 1] \leq \frac{1}{2}$, and let $f = \mathbf{E}\left[\sum_{v \in K} x_v\right] = \sum_{v \in K} \mathbf{Pr}[x_v = 1]$. Then $\mathbf{Pr}\left[\sum_{v \in K} x_v = 1\right] \geq f4^{-f}$.*

Proof.

$$\begin{aligned}
\mathbf{Pr}\left[\sum_{v \in K} x_v = 1\right] &= \sum_{u \in K} \mathbf{Pr}\left[x_u = 1 \wedge x_v = 0 \forall v \neq u\right] \\
&\geq \sum_{u \in K} \mathbf{Pr}[x_u = 1] \cdot \mathbf{Pr}[x_v = 0 \forall v] \\
&\geq f \cdot \mathbf{Pr}[x_v = 0 \forall v] = f \cdot \prod_{v \in K} (1 - \mathbf{Pr}[x_v = 1]) \\
&\geq f \cdot \prod_{v \in K} 4^{-\mathbf{Pr}[x_v = 1]} = f \cdot 4^{-\sum_{v \in K} \mathbf{Pr}[x_v = 1]} = f4^{-f} \quad . \quad \square
\end{aligned}$$

Let K be a fixed subset of $[n]$ with size $k' \leq k$, and let j be a fixed column in the i^{th} range. $\sum_{v \in K} \mathbf{Pr}[S(v)_j = 1] = \frac{k'}{2^i}$, so by Lemma 25, the probability that j hits K is at least $\frac{k'}{2^i} 4^{-\frac{k'}{2^i}}$. Note that when $i \geq \log k'$, this probability is at least $\frac{k'}{2^i} 4^{-1} = \frac{k'}{2^{i+2}}$. Since the columns j are all independent, the probability

that no column hits K is at most

$$\begin{aligned}\prod_{i=1}^{\log k}\left(1-\frac{k'}{2^i}4^{-\frac{k'}{2^i}}\right)^{c2^i\log\frac{n}{k}} &\leq \prod_{i=\lceil\log k'\rceil}^{\log k}\left(1-\frac{k'}{2^{i+2}}\right)^{c2^i\log\frac{n}{k}}\\ &\leq \prod_{i=\lceil\log k'\rceil}^{\log k} e^{-\frac{k'c2^i\log\frac{n}{k}}{2^{i+2}}}\\ &\leq e^{-\frac{k'\log\frac{k}{2k'}c\log\frac{n}{k}}{4}}\\ &\leq e^{-\frac{k'c\log\frac{n}{2k'}}{8}}\\ &\leq e^{-\frac{k'c\log\frac{n}{k'}}{9}}\quad .\end{aligned}$$

We now set $c = 27$, so that $\mathbf{Pr}[K \text{ is not hit}] \leq e^{-3k'\log\frac{n}{k'}}$

The number of subsets K of size k' is $\binom{n}{k'} \leq e^{k'\log\frac{n}{k'}}$. Taking a union bound over these, the probability that any set K of size k' is not hit is at most:

$$e^{k'\log\frac{n}{k'}}\cdot e^{-3k'\log\frac{n}{k'}} = e^{-2k'\log\frac{n}{k'}}$$

We take another union bound, this time over all values of k' between 1 and k. Thus, the probability that any set K is not hit is at most:

$$\sum_{k'=1}^{k} e^{-2k'\log\frac{n}{k'}} < 1$$

So, there is a non-zero probability that S does indeed hit all K, i.e. meets the criteria for an (n,k)-selective family of size at most $O(k\log\frac{n}{k})$. Therefore, such an object must exist. □

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., [10, 15]). 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 application 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 [10].

4.3.2 Block Transmission Schedules

Next, we introduce *block transmission schedules*, which are a new type of combinatorial object designed for use in our fast broadcasting algorithm. The major difference between block transmission schedules and the transmission schedules of De Marco [47] is that we divide time into *blocks* of some length B, and say that nodes only become *active* (and start transmitting) at the beginning of the next block after they wake up.

As mentioned, our aim with all of our combinatorial is to *hit* all possible *instances*. We assume that n and D are fixed, and define our instances here.

Definition 26. *For $k \leq n$, a k-**instance** X is a set $K \subseteq [n]$, $|K| = k$, with activation function $\omega : K \to \mathbb{N}$ satisfying $|\{v \in K : \omega(v) \leq \omega(K) + 1\}| \geq \frac{n}{D}$.*

The *activation function* ω maps each node to the *block* in which it becomes active when our algorithm is run on this instance. This means that the activation function depends on the algorithm, but there is no circular dependency: whether nodes become active in time-step j only depends on the algorithm's behavior in previous time-steps, and the algorithm's behavior at time-step j only depends on the activation function up to j. We will also extend ω to sets of nodes in the instance by $\omega(K) := \min_{v \in K} \omega(v)$.

The condition $|\{v \in K : \omega(v) \leq \omega(K) + 1\}| \geq \frac{n}{D}$ is because our block transmission schedules are only designed to inform nodes with active in-neighborhoods of size at least $\frac{n}{D}$. We will use selective families to inform smaller neighborhoods, and will take less than B time to do so. So, the instances we need to cover with our block transmission schedules are those that contain at least $\frac{n}{D}$ active nodes within the first B time-steps, and so fail to be informed by selective family.

Having defined our instances, we can now explain the reason for introducing the blocks: since we only care about the block in which nodes become active, rather than the exact time-step, we will be able to significantly cut down the number of instances we must consider.

We now define block transmission schedules:

Definition 27. *For a function $h : [n] \to \mathbb{N}$, an (n, D)-**block transmission schedule of delay** h is a function $T : [n] \to \{0,1\}^{h(n)}$ such that for any k-**instance**, there is some time-step $j \leq B\omega(K)+h(k)$ with $\sum_{v \in K} T(v)_{j-B\omega(v)} = 1$.*

Node v will refer to entry $T(v)_{j-B\omega(v)}$ to determine whether to transmit in time-step j. Then, for any set of nodes K with $\frac{n}{D} \leq k$, some nodes in the set first become active at time-step $B\omega(K)$, and so an (n, D)-block transmission schedule ensures that one node transmits alone within the next $h(k)$ time-steps.

Existence of small block transmission schedules

We will show the existence of small block transmission schedules in the following theorem.

Theorem 28. *There exists an (n, D)-**block transmission schedule of delay** h, where $h(k) = O(k \log D \log\log D)$.*

We will prove the existence of a small block transmission schedules by randomly generating a candidate T, and proving that it indeed has the required properties with positive probability.

Let c be a constant to be chosen later. Let $z = 2\log\log D$ be the *phase* length: within every z-length phase, our construction probabilities will decay exponentially to allow nodes to correctly 'guess' in-neighborhood size. To this end, let $\rho(j) = j \bmod z$, i.e. ρ is a function which tells us how far through its phase a time-step j is.

We will set our block size $B = c\frac{n}{D} z \log D$. Recall that nodes wait until the next block after waking up to become active. We will set our delay function $h(k) = c^2 kz \log D$, which is $O(k \log D \log\log D)$ as claimed in Theorem 28.

Our candidate object T will be generated as follows: for each node v, we independently choose each entry $T(v)_i$ to be 1 with probability

$$\frac{cz \log D}{(B+i)2^{\rho(i)+1}} \ .$$

.

Our first step is to cut down the number of k-instances we must consider. There are $\binom{n}{k}$ possible choices of K, and this cannot be reduced. We can, however, reduce the number of possible activation functions ω. Fix a basic time-step j, let K_j be the set of nodes awake by time-step j, and let $k_j = |K_j|$. We make the following observations:

- We can assume $\omega(K) = 0$, since otherwise we simply subtract $\omega(K)$ from all activation times without affecting node behavior.
- We need only consider instances with $j \leq h(k_j) + B \ \forall j < h(k)$. This is because if there is some $j < h(k)$ with $j > h(k_j) + B$, then we can *curtail* the k-instance to be a k_j-instance consisting only of the nodes in K_j. If this latter instance is hit within $h(k_j)$ time, then the former instance will also be hit. Note that it is not possible that $k_j < \frac{n}{D}$ (which would make the curtailed instance invalid), because we assumed in Definition 26 that $|\{v \in K : \omega(v) \leq 1\}| \geq \frac{n}{D}$.

Henceforth we consider only curtailed instances with $\omega(K) = 0$.

We would like to have a measure of the expected number of transmissions in that time-step, with respect to some fixed k-instance. We will call this the *load*:

Definition 29. *The **load** $f(j)$ of a time-step j is given by:*

$$f(j) := \sum_{v \in K_j} \frac{cz \log D}{(B + j - B\omega(v))2^{\rho(j)+1}} \ .$$

Our aim is to ensure that the load is within some constant range as often as possible, since if the load $f(j)$ is bounded by constants then there is a good probability that j hits the instance, which we can show using Lemma 25. since

Load is just a sum of independent $\{0, 1\}$-valued variables $T(v)_i$, and since for all nodes v and $j \geq B\omega(v)$ we have

$$\frac{cz \log D}{(B + j - B\omega(v))2^{\rho(j)+1}} \leq \frac{cz \log D}{2B} \leq \frac{1}{2} \ ,$$

we can plug in $x_v = T(v)_{j - B\omega(v)}$ to Lemma 25 and find that the probability of hitting a k-instance in time-step j is at least $f(j)4^{-f(j)}$.

We bound the load of basic time-steps from below:

Lemma 30. *All basic time-steps j have $f(j) \geq \frac{1}{3c}$.*

Proof. Due to our curtailing of instances, we have $j \leq h(k_j) + B$. So,

$$\begin{aligned} f(j) &= \sum_{v \in K_j} \frac{cz \log D}{2(B + j - B\omega(v))} \\ &\geq \sum_{v \in K_j} \frac{cz \log D}{2(2B + h(k_j))} \\ &= \frac{ck_j z \log D}{2(2cz\frac{n}{D} \log D + c^2 k_j z \log D)} \\ &\leq \frac{1}{3c} \qquad \text{(assuming } c \geq 4\text{).} \end{aligned}$$

□

We next examine time-steps at the end of phases, i.e., with $j \bmod z \equiv -1$, and call these *ending* time-steps. Note that for ending time-steps,

$$f(j) = \sum_{v \in K_j} \frac{cz \log D}{(B + j - B\omega(v))2^z} \ .$$

We bound the load of (a constant fraction of) ending time-steps from above:

Lemma 31. *Let $\mathcal{F}$ denote the set of ending time-steps j such that $f(j) \leq 1$. Then $|\mathcal{F}| \geq \frac{c^2 k \log D}{2}$.*

Proof. The total load over all ending time-steps can be bounded as follows:

$$\sum_{\text{ending timestep } j} f(j) \leq \sum_{i=1}^{h(k)/z} f(iz-1) = \sum_{i=1}^{c^2 k \log D} f(iz-1) \ .$$

Applying the definition of f, $f(iz-1)$ is equal to:

$$\sum_{v \in K_{iz-1}} \frac{cz \log D}{(B + iz - 1 - B\omega(v))2^z} \ ,$$

So,

$$\begin{aligned}
\sum_{\text{ending timestep } j} f(j) &\leq \sum_{i=1}^{c^2 k \log D} \sum_{v \in K_{iz-1}} \frac{cz \log D}{(B + iz - 1 - B\omega(v))2^z} \\
&\leq \frac{cz \log D}{2^z} \sum_{v \in K} \sum_{i=1}^{c^2 k \log D} \frac{1}{B + iz - 1} \\
&\leq \frac{ckz \log D}{\log^2 D} \int_0^{c^2 k \log D} \frac{di}{B + iz - 1} \\
&= \frac{ckz}{\log D} \cdot \frac{\ln(B + c^2 zk \log D - 1) - \ln(B - 1)}{z} \\
&= \frac{ck \ln \frac{B + c^2 zk \log D - 1}{B - 1}}{\log D} \\
&\leq \frac{ck \ln 2cD}{\log D} \leq ck \ .
\end{aligned}$$

Therefore, at most ck ending time-steps have load higher than 1, and so at least $c^2 k \log D - ck \geq \frac{c^2 k \log D}{2}$ ending time-steps have $f(j) \leq 1$. □

We can use Lemma 31 to show that we have sufficiently many time-steps with load within the interval $[\frac{1}{3c}, 1]$:

Lemma 32. *Let $\mathcal{F}$ be the set of time-steps $B\omega(K) < j \leq B\omega(K) + h(k)$ with $\frac{1}{3c} \leq f(j) \leq 1$. Then $|\mathcal{F}| \geq \frac{c^2 k \log D}{2}$.*

Proof. It can be seen that load decreases by at most a multiplicative factor of 3 between consecutive time-steps (since the contribution of each node decreases by at most a factor of 3). So, since every base time-step has load at least $\frac{1}{3c}$, for every ending timestep j with $f(j) \leq 1$, there is some j' in the preceding phase with $\frac{1}{3c} \leq f(j') \leq 1$. □

Since these time-steps have constant load, they have constant probability of hitting:

Lemma 33. *For any time-step $j \in \mathcal{F}$, the probability that j hits is at least $\frac{1}{5c}$.*

Proof. By Lemma 25, the probability that j hits is at least $f(j)4^{-f(j)}$. This is minimized over the range $[\frac{1}{3c}, 1]$ at $f(j) = \frac{1}{3c}$ and is therefore at least $\frac{4^{-\frac{1}{3c}}}{3c} \geq \frac{1}{5c}$. □

We now need to know how many unique (r, k)-instances we must hit. Since we are only concerned with the first $h(k)$ time-steps after the first node wakes up, we need only consider (k)-instances which are unique within this time range.

Lemma 34. *For any (sufficiently large) k, the number of unique (up to the first $h(k)$ steps after activation) k-instances is at most $2^{2k \log D}$.*

Proof. There are $\binom{n}{k} \leq (\frac{en}{k})^k$ possible choices for set K. The number of possible wake-up functions $\omega : K \to [h(k)/B]$ for a fixed K is at most $(\frac{h(k)}{B})^k = (\frac{ckD}{n})^k$, and so the total number of k-instances is at most $(\frac{en}{k} \frac{ckD}{n})^k = (ecD)^k \leq 2^{2k \log D}$. □

This allows us to use a union bound over all k-instances to show that the probability that any are not hit is small.

Lemma 35. *For any (sufficiently large) k, the probability that T does not hit all k-instances is at most $2^{-2k \log D}$.*

Proof. Fix some k-instance. The probability that it is not hit within $h(k)$ time-steps is at most

$$\prod_{j \in \mathcal{F}} (1 - \frac{1}{5c}) \leq e^{-\frac{|\mathcal{F}|}{5c}} \leq e^{-\frac{ck \log D}{10}} = 2^{-\frac{ck \log D}{10 \ln 2}} \quad .$$

Hence, if we set $c = 14$, by a union bound the probability that any k-instance is not hit is at most $2^{2k \log D} \cdot 2^{-\frac{14k \log D}{10 \ln 2}} \leq 2^{-2k \log D}$. □

We can now prove our main theorem on block transmission schedules (Theorem 28):

Proof. By Lemma 35 and a union bound over k, the probability that T does not hit all instances is at most $\sum_{k=\frac{n}{D}}^{n} 2^{-2k \log D} < 1$. Therefore T is an (n, D)-block transmission schedule of delay h with non-zero probability, so such an object must exist. □

4.3.3 Unbounded Universal Synchronizers

For the task of wake-up in blind radio networks, i.e., in the absence of a global clock or parameter knowledge, we will define an object called an **unbounded universal synchronizer** for use in our algorithm. This will be an infinite object, in which a node's behavior depends entirely on its ID and the amount of time since it woke up.

We begin by defining the instances our algorithm must account for:

Definition 36. *An (r, k)-**instance** X is a set K of k nodes with*

$$\sum_{v \in K} \log v = r$$

and wake-up function $\omega : K \to \mathbb{N}$.

(By using v as an integer here, we are abusing notation to mean the ID of node v.)

Here r is the main parameter we will use to quantify how 'large' our input instance is. By using the sum of logarithms of IDs (which approximates the total number of bits needed to write all IDs in use), we capture both the number of participating nodes and the length of IDs. We require r to be an integer, so we round down accordingly, but we omit floor functions for clarity since they do not affect the asymptotic result. Since we are only concerned with asymptotic behavior, we can also assume that r is larger than a sufficiently large constant.

The *wake-up function* ω maps each node to the *time-step* (not the *block*, as for block transmission schedules) it wakes up (either spontaneously or by receiving a transmission) when our algorithm is run on this instance. As before, this means that the wake-up function depends on the algorithm, but there is no circular dependency: whether nodes wake-up in time-step j only depends on the algorithm's behavior in previous time-steps, and the algorithm's behavior at time-step j only depends on the wake-up function up to j.

We now define the combinatorial object that will be the basis of our algorithm:

Definition 37. *For a function $h : \mathbb{N} \times \mathbb{N} \to \mathbb{N}$, an **unbounded universal synchronizer of delay** h is a function $S : \mathbb{N} \to \{0, 1\}^{\mathbb{N}}$ such that for any (r, k)-instance, there is some time-step $j \leq \omega(K) + h(r, k)$ with $\sum_{v \in K} S(v)_{j - \omega(v)} = 1$.*

The *unbounded universal synchronizer* S is a function mapping node IDs to a sequence of 0s and 1s, which tell nodes when to listen and transmit respectively. In our wake-up algorithm, a node v becomes *active* immediately upon waking up, and so refers to entry $S(v)_{j-\omega(v)}$ to determine whether to transmit in time-step j. It does not know the value j, but can count from $\omega(v)$.

We will apply this object to perform wake-up as follows: each node v transmits a message in time-step j (with its time-step count starting upon waking up) iff $S(v)_j = 1$. Then, the property guarantees that at some time-step j within

$h(r,k)$ time-steps of the first node waking up, any (r,k)-instance will have a successful transmission. We call this S *'hitting' the* (r,k)*-instance at time-step* j. So, our aim is to show the existence of such an object, with h growing as slowly as possible.

Our main technical result in this section is the following:

Theorem 38. *There exists an* ***unbounded universal synchronizer of delay*** h*, where* $h(r,k) = O\left(\frac{r \log k}{\log \log k}\right)$.

Our approach to proving Theorem 38 will be to randomly generate a candidate synchronizer, and then prove that with positive probability it does indeed satisfy the required property. Then, for this to be the case, at least one such object must exist.

Our candidate $S : \mathbb{N} \to \{0,1\}^{\mathbb{N}}$ will be generated by independently choosing each $S(v)_j$ to be $\mathbf{1}$ with probability $\frac{c \log v}{j + 2c \log v}$ and $\mathbf{0}$ otherwise, where c is a constant to be chosen later.

While S is drawn from an uncountable set, we will only be concerned with events that depend upon a finite portion of it, and countable unions and intersections thereof. Therefore, we can use as our underlying σ-algebra that generated by the set of all events $E_{v,j} = \{S : S(v)_j = 1\}$, where $v, j \in \mathbb{N}$, with the corresponding probabilities $\mathbf{Pr}[E_{v,j}] = \frac{c \log v}{j + 2c \log v}$.

We set delay function $h(r,k) = \frac{c^2 r \log k}{\log \log k}$.

We can again make some observations which allow us to restrict the instances we consider:

- We can assume $\omega(K) = 0$, since otherwise we simply subtract $\omega(K)$ from all activation times without affecting node behavior.
- We need only consider instances with $j \leq h(r_j, k_j)\ \forall j < h(r,k)$, because if there is some $j < h(k)$ with $j > h(r_j, k_j)$ then we can *curtail* the (r,k)-instance to be an (r_j, k_j)-instance consisting only of the nodes in K_j. If this latter instance is hit within $h(r_j, k_j)$ time, then the former instance will also be hit.

To analyze the probability of hitting (r,k)-instances, again define the *load* of a time-step $f(j)$ to be the expected number of transmissions in that time-step:

Definition 39. *For a fixed* (r,k)*-instance, the* ***load*** $f(j)$ ***of a time-step*** j *is defined as*

$$\sum_{v \in K_j} \mathbf{Pr}[v \text{ transmits}] = \sum_{v \in K_j} \frac{c \log v}{j - \omega(v) + 2c \log v} \ .$$

We now seek to bound the load from above and below, since when the load is close to constant we have a good chance of hitting.

Lemma 40. *All time-steps $j \leq h(r,k)$ have $f(j) \geq \frac{\log\log k}{2c\log k}$.*

Proof. Fix a time-step $j \leq h(r,k)$, let K_j be the set of nodes awake by time-step j, and let $k_j = |K_j|$ and $r_j = \sum_{v\in K_j} \log v$, analogous to r and k. Due to our curtailing of instances, we have $j \leq h(r_j, k_j)$. So,

$$\begin{aligned} f(j) &\geq \sum_{v\in K_j} \frac{c\log v}{j + 2c\log v} \geq \frac{cr_j}{h(r_j,k_j) + 2cr_j} \\ &\geq \frac{cr_j}{\frac{2c^2 r_j \log k_j}{\log\log k_j}} \geq \frac{\log\log k_j}{2c\log k_j} \geq \frac{\log\log k}{2c\log k} \quad . \end{aligned}$$ □

Having bounded load from below, we also seek to bound it from above. Unfortunately, the load in any particular time-step can be very high, but we can get a good bound on at least a constant fraction of the columns.

Lemma 41. *Let F denote the set of time-steps $j \leq h(r,k)$ such that $\frac{\log\log k}{2c\log k} \leq f(j) \leq \frac{\log\log k}{3}$. Then $|F| \geq \frac{c^2 r\log k}{2\log\log k}$.*

Proof. The total load over all time-steps can be bounded as follows:

$$\begin{aligned} \sum_{j\leq h(r,k)} f(j) &= \sum_{j\leq h(r,k)} \sum_{v\in K_j} \frac{c\log v}{j - \omega(v) + 2c\log v} \\ &\leq \sum_{v\in K} \sum_{\omega(v)<j\leq h(r,k)} \frac{c\log v}{j - \omega(v) + 2c\log v} \\ &\leq \sum_{v\in K} c\log v \sum_{j\leq h(r,k)} \frac{1}{j + 2c\log v} \\ &\leq \sum_{v\in K} c\log v \ln\frac{2h(r,k)}{4c\log v} \quad . \end{aligned}$$

We require a good upper bound for this last expression, but this is tricky since we don't know which nodes are in K. So, we partition K by ranges of node ID, and deal with each of the ranges separately.

Let $K^i = \{v \in K : \frac{r}{k\cdot 2^i} \leq \log v < \frac{r}{k\cdot 2^{i-1}}\}$, for $i \in \mathbb{N}$, and $K' = \{v \in K : \log v \geq \frac{r}{k}\}$

We next show that each of the ranges cannot contribute too much of the

total of IDs. If, for any i, we have $\sum_{v\in K_i} \log v > \frac{r}{2^i}$, then

$$r < 2^i \sum_{v\in K_i} \log v \leq 2^i \sum_{v\in K_i} \frac{r}{k\cdot 2^i} \leq r \ .$$

This gives a contradiction, so we must have $\sum_{v\in K_i} \log v \leq \frac{r}{2^i}$. Then,

$$\begin{aligned}
\sum_{j\leq h(r,k)} f(j) &\leq \sum_{v\in K} c\log v \ln \frac{2h(r,k)}{4c\log v} \\
&\leq \sum_{i\geq 1}\sum_{v\in K_i} c\log v \ln \frac{h(r,k)}{2c\log v} + \sum_{v\in K'} c\log v \ln \frac{h(r,k)}{2c\log v} \\
&\leq \sum_{i\geq 1}\sum_{v\in K_i} c\log v \ln \frac{h(r,k)}{2c\frac{r}{k\cdot 2^i}} + \sum_{v\in K'} c\log v \ln \frac{h(r,k)}{2c\frac{r}{k}} \\
&= \sum_{i\geq 1}\sum_{v\in K_i} c\log v \ln \frac{ck2^{i-1}\log k}{\log\log k} + \sum_{v\in K'} c\log v \ln \frac{ck\log k}{2\log\log k} \\
&\leq \sum_{i\geq 1} cr2^{-i}(2\ln k + (i-1)\ln 2) + 2cr\ln k \leq 5cr\ln k \leq 8cr\log k \ .
\end{aligned}$$

Therefore, at most $\frac{24cr\log k}{\log\log k}$ time-steps have load higher than $\frac{\log\log k}{3}$. Since by Lemma 40 all time-steps have load at least $\frac{\log\log k}{2c\log k}$,we have $|F| \geq h(r,k) - \frac{24cr\log k}{\log\log k} \geq \frac{c^2 r\log k}{2\log\log k}$ (provided we pick $c \geq 7$). □

We can use Lemma 25 to show that the probability that we hit on our 'good' time-steps (those in F) is high:

Lemma 42. *For any time-step $j \in F$, the probability that j hits is at least $\frac{\log\log k}{3c\log k}$.*

Proof. $\frac{\log\log k}{2c\log k} \leq f(j) \leq \frac{\log\log k}{3}$, and so $f(j)4^{-f(j)}$ is minimized at $f(j) = \frac{\log\log k}{2c\log k}$ and is therefore at least $\frac{\log\log k}{2c\log k}4^{-\frac{\log\log k}{2c\log k}} \geq \frac{\log\log k}{3c\log k}$. □

We now bound the number of possible instances we must hit:

Lemma 43. *For any (sufficiently large) r, the number of unique (r,k)-instances is at most 2^{5r}.*

Proof. The total number of bits used in all node IDs in the instance is at most r. There are at most 2^{r+1} possible bit-strings of length at most r, and at most 2^r ways of dividing each of these into substrings (for individual IDs), giving at most

2^{2r+1} sets of node IDs. The number of possible wake-up functions $\omega : K \to \mathbb{N}$ is at most $h(r,k)^k$, since all nodes must be awake by $h(r,k)$ time or the instance would have been curtailed.

$$h(r,k)^k = 2^{k \log h(r,k)} \leq 2^{1.1k \log r} = 2^{1.1(k \log k + k \log \frac{r}{k})} \leq 2^{1.3(k \log(k^{0.9}) + r)} \leq 2^{2.9r} \ .$$

So, the total number of possible (r,k)-instances is at most $2^{2r+1+2.9r} \leq 2^{5r}$. □

Lemma 44. *For any (sufficiently large) r, the probability that S does not hit all (r,k)-instances is at most 2^{-3r}*

Proof. Fix some (r,k)-instance. The probability that it is not hit within $h(k,r)$ time-steps is at most

$$\prod_{j \in F} (1 - \frac{\log \log k}{3c \log k}) \leq e^{-|F| \frac{\log \log k}{3c \log k}} \leq e^{-\frac{2}{3}cr} = 2^{-\frac{2cr}{3 \ln 2}} \ ,$$

by Lemma 42. Hence, if we set $c = 9$, by a union bound the probability that any (r,k)-instance is not hit is at most $2^{5r} \cdot 2^{-\frac{18r}{3 \ln 2}} \leq 2^{-3r}$. □

We can now prove our main theorem on unbounded universal synchronizers (Theorem 38):

Proof. By Lemma 44 and a union bound over r, the probability that S does not hit all instances is at most $\sum_{r \in \mathbb{N}} 2^{-3r} < 1$. Therefore S is an unbounded universal synchronizer of delay g with non-zero probability, so such an object must exist. □

4.3.4 Unbounded Transmission Schedules

Our final combinatorial structure is the **unbounded transmission schedule**, for broadcasting in blind networks. This combines some aspects of block transmission schedules and some of unbounded universal synchronizers:

Since we have access to a global clock we can use absolute time-step values, as we did for block transmission schedules. However, we cannot use blocks, since in blind networks nodes do not know what the appropriate block size should be. One might think that we can also not use phases (the short periods over which transmission probability rapidly decays), since the length of these is also based on network parameter n, but it transpires that we can instead have phase

length depend on the current time-step number, though this does significantly complicate the calculations.

As for unbounded universal synchronizers, our combinatorial object must be infinite, since nodes know a bound on completion time.

We formally unbounded transmission schedules below. We use the same definition of (r, k)-instances as in the previous section, for unbounded universal synchronizers.

Definition 45. *For a function $h : \mathbb{N} \times \mathbb{N} \to \mathbb{N}$, an **unbounded transmission schedule of delay** h is a function $T : \mathbb{N} \times \mathbb{N} \to \{0, 1\}^{\mathbb{N}}$ such that $T(v, \omega(v))_j = 0$ for any $j < \omega(v)$, and for any (r, k)-instance there is some time-step $j \leq \omega(K) + h(r, k)$ with $\sum_{v \in K} T(v, \omega(v))_j = 1$.*

The difference here from an unbounded universal synchronizer is that nodes now know the global time-step in which they wake up, and so their transmission patterns can depend upon it. This is the second argument of the function T. The other difference in the meaning of the definition is that the output of T now corresponds to absolute time-step numbers, rather than being relative to each node's wake-up time as for unbounded universal synchronizers. That is, the j^{th} entry of a node's output sequence tells it how it should behave in global time-step j, rather than j time-steps after it wakes up.

Our existence result for unbounded transmission schedules is the following:

Theorem 46. *There exists an **unbounded transmission schedule of delay** h, where $h(r, k) = O(r \log \log k)$.*

Our method will again be to randomly generate a candidate unbounded transmission schedule T, and then prove that it satisfies the required property with positive probability, so some such object must exist.

Let c be a constant to be chosen later. Our candidate object T will be generated as follows: for each node v, we generate a transmission sequence $s_{v,j}$, $j \in \mathbb{N}$, with $s_{v,j}$ independently chosen to be 1 with probability $\frac{c \log v \log \log j}{j + 2c \log v \log \log j}$ and 0 otherwise.

However, these will not be our final probabilities for generating T. To make use of our global clock, we will divide time into short phases during which transmission probability will decay exponentially. The lengths of these phases will be based on a function $z(j) := 2^{\lceil 1 + \log \log \log j \rceil}$, i.e., $\log \log j$ rounded up to the next-plus-one power of 2. An important property to note is that for all i, $z(i) | z(i + 1)$. We also generate a sequence $p_{v,j}$, $j \in \mathbb{N}$ of *phase values*, each

chosen independently and uniformly at random from the real interval $[0, 1]$. These, combined with the global time-step number and current phase length, will give us our final generation probabilities.

We set $T(v, \omega(v))_j$ to equal 1 iff $s_{v,j-\omega(v)} = 1$ and $p_{v,j-\omega(v)} \leq 2^{-j \bmod z(j-\omega(v))}$.

It can then be seen that

$$\mathbf{Pr}\left[T(v, \omega(v))_j = 1\right] = \frac{c \log v \log\log(j - \omega(v))}{(j - \omega(v) + 2c \log v \log\log(j - \omega(v)))2^{j \bmod z(j-\omega(v))}} \ .$$

The reason we split the process of random generation into two steps (using our phase values) is that now, if we shift all wake-up times in an (r, k)-instance by the same multiple of $z(x)$, then node behavior in the first x time-steps after $\omega(K)$ does not change. We will require this property when analyzing T.

Our probabilistic analysis is with respect to the σ-algebra generated by all events $E_{v,\omega(v),j} = \{T : T(v, \omega(v))_j = 1\}$, with $v, \omega(v), j \in \mathbb{N}$, and with the corresponding probabilities given above.

Let **load $f(j)$ of a time-step** j be again defined as the expected number of transmissions in that time-step:

$$f(j) := \sum_{v \in K_j} \frac{c \log v \log\log(j - \omega(v))}{(j - \omega(v) + 2c \log v \log\log(j - \omega(v)))2^{j \bmod z(j-\omega(v))}} \ .$$

We will set our delay function $h(r, k) = c^2 r \log\log k$.

Again we make some observations that allow us to narrow the circumstances we must consider: firstly that we can again assume that r is larger than a sufficiently large constant, and secondly that we need only look at curtailed instances (i.e., we can assume $j - \omega(K) \leq h(r_j, k_j) \forall j < h(r, k)$). This time, however, we cannot shift instances to begin at time-step 0, because node behavior is dependent upon global time-step number.

We follow a similar line of proof as before, except with some extra complications in dealing with phases. We first consider only time-steps at the beginning of each phase, i.e., time-steps $\omega(K) < j \leq \omega(K) + h(r, k)$ with $j \bmod z(h(r, k)) \equiv 0$, and we will call these *basic* time-steps. Notice that for basic time-steps,

$$f(j) = \sum_{v \in K_j} \frac{c \log v \log\log(j - \omega(v))}{j - \omega(v) 2c \log v \log\log(j - \omega(v))} \ .$$

We bound the load of basic time-steps from below:

Lemma 47. *All basic time-steps j have $f(j) \geq \frac{1}{2c}$.*

Proof. Fix a basic time-step j, let K_j be the set of nodes awake by time-step j, and let $k_j = |K_j|$ and $r_j = \sum_{v \in K_j} \log v$, analogous to r and k. If $k_j = k$, then

$$\begin{aligned} f(j) &\geq \sum_{v \in K} \frac{c \log v \log\log(j - \omega(v))}{j - \omega(v) + 2c \log v \log\log(j - \omega(v))} \\ &\geq \sum_{v \in K} \frac{c \log v \log\log h(r,k)}{h(r,k) + 2c \log v \log\log h(r,k)} \\ &\geq \sum_{v \in K} \frac{c \log v \log\log k}{2c^2 r \log\log k} \geq \frac{cr}{2c^2 r} = \frac{1}{2c} \ . \end{aligned}$$

If $k_j < k$, then due to our curtailing of instances, we have $j - \omega(K) \leq h(r_j, k_j)$. So,

$$\begin{aligned} f(j) &\geq \sum_{v \in K_j} \frac{c \log v \log\log(j - \omega(v))}{j - \omega(v) + 2c \log v \log\log(j - \omega(v))} \\ &\geq \sum_{v \in K} \frac{c \log v \log\log h(r_j,k_j)}{h(r_j,k_j) + 2c \log v \log\log h(r,k)} \\ &\geq \sum_{v \in K} \frac{c \log v \log\log k_j}{2c^2 r_j \log\log k_j} \geq \frac{cr_j}{2c^2 r_j} = \frac{1}{2c} \ . \end{aligned}$$

□

We next examine time-steps at the end of phases, i.e., with $\omega(K) < j \leq \omega(K) + h(r,k)$ and $j \bmod z(h(r,k)) \equiv -1$, and call these *ending* time-steps. Note that for ending time-steps,

$$f(j) = \sum_{v \in K_j} \frac{c \log v \log\log(j - \omega(v))}{(j - \omega(v) + 2c \log v \log\log(j - \omega(v)))2^{z(j-\omega(v))-1}} \ .$$

We bound the load of (a constant fraction of) ending time-steps from above:

Lemma 48. *Let $\mathcal{F}$ denote the set of ending time-steps j such that $f(j) \leq 1$. Then $|\mathcal{F}| \geq \frac{c^2 r}{2}$.*

Proof. Let t be the first ending time-step. The total load over all ending time-steps can be bounded as follows:

$$\sum_{\text{ending timestep } j} f(j) \leq \sum_{i=0}^{h(r,k)/z(h(r,k))} f(t+iz(h(r,k))) \leq \sum_{i=0}^{c^2 r} f(t+iz(h(r,k))) \ .$$

Applying the definition of f, $f(t+iz(h(r,k)))$ is equal to:

$$\sum_{v \in K_{t+iz(h(r))}} \frac{c \log v \log\log(t+iz(h(r,k))-\omega(v)) 2^{-z(t+iz(h(r,k))-\omega(v))-1}}{t+iz(h(r,k))-\omega(v)+2c\log v \log\log(t+iz(h(r,k))-\omega(v))} \ ,$$

which is bounded from above when $t-\omega(v)=0$:

$$\begin{aligned} f(t+iz(h(r,k))) &\leq \sum_{v \in K_{t+iz(h(r))}} \frac{c \log v \log\log(iz(h(r))) 2^{-z(iz(h(r,k)))}}{iz(h(r,k))+2c\log v \log\log(iz(h(r,k)))} \\ &\leq \sum_{v \in K_{t+iz(h(r,k))}} \frac{c \log v \log\log(iz(h(r,k)))}{iz(h(r,k)) 2^{z(iz(h(r,k)))}} \ . \end{aligned}$$

So,

$$\begin{aligned} \sum_{\text{ending timestep } j} f(j) &\leq \sum_{i=0}^{c^2 r} \sum_{v \in K_{t+iz(h(r,k))}} \frac{c \log v \log\log(iz(h(r,k)))}{iz(h(r,k)) 2^{z(iz(h(r,k)))}} \\ &\leq \sum_{v \in K} \sum_{i=0}^{c^2 r} \frac{c \log v \log\log(iz(h(r,k)))}{iz(h(r,k)) 2^{z(iz(h(r,k)))}} \\ &\leq \sum_{v \in K} \sum_{i=0}^{c^2 r} \frac{2c \log v \log\log(iz(h(r,k)))}{iz(h(r,k)) \log^2(iz(h(r,k)))} \\ &\leq \sum_{v \in K} 2c \log v \sum_{i=0}^{\infty} \frac{\log\log(iz(h(r,k)))}{iz(h(r,k)) \log^2(iz(h(r,k)))} \\ &\leq 10cr \ . \end{aligned}$$

Here the last inequality follows since the second sum converges to a constant less than 5, which can be seen by inspection of the first three terms and using the integral bound $\int_2^\infty \frac{\log\log x}{x \log^2 x} < 2$.

Therefore, at most $10cr$ ending time-steps have load higher than 1, and so at least $c^2 r - 10cr \geq \frac{c^2 r}{2}$ (provided we set $c \geq 5$) ending time-steps have $f(j) \leq 1$. □

We can use Lemma 48 to show that we have sufficiently many time-steps with load within the interval $[\frac{1}{2c}, 1]$:

Lemma 49. *Let $\mathcal{F}$ be the set of time-steps $\omega(K) < j \leq \omega(K) + h(r,k)$ with $\frac{1}{2c} \leq f(j) \leq 1$. Then $|\mathcal{F}| \geq \frac{c^2 r}{2}$.*

Proof. It can be seen that load decreases by at most a multiplicative factor of 3 between consecutive time-steps (since the contribution of each node decreases by at most a factor of 3). So, since every base time-step has load at least $\frac{1}{2c}$, for every ending timestep j with $f(j) \leq 1$, there is some j' in the preceding phase with $\frac{1}{2c} \leq f(j') \leq 1$. □

Since these time-steps have constant load, they have constant probability of hitting:

Lemma 50. *For any time-step $j \in \mathcal{F}$, the probability that j hits is at least $\frac{1}{3c}$.*

Proof. By Lemma 25, the probability that j hits is at least $f(j)4^{-f(j)}$. This is minimized over the range $[\frac{1}{2c}, 1]$ at $f(j) = \frac{1}{2c}$ and is therefore at least $\frac{4^{-\frac{1}{2c}}}{2c} \geq \frac{1}{3c}$. □

We now need to know how many unique (r,k)-instances we must hit. Since we are only concerned with the first $h(r,k)$ time-steps after the first node wakes up, we need only consider (r,k)-instances which are unique within this time range.

Lemma 51. *For any (sufficiently large) r, the number of unique (up to the first $h(r,k)$ steps after activation) (r,k)-instances is at most 2^{5r}.*

Proof. As before (in Lemma 43) there are at most 2^{2r+1} sets of node IDs. The number of possible wake-up functions $\omega : K \to \mathbb{N}$ for a fixed $\omega(K)$ is again at most $h(r,k)^k$, and though we are using a different delay function to the previous section, the calculations used to prove Lemma 43 still hold.

$$h(r)^k = 2^{k \log h(r,k)} \leq 2^{1.1k \log r} = 2^{1.1(k \log k + k \log \frac{r}{k})} \leq 2^{1.3(k \log(k^{0.9}) + r)} \leq 2^{2.9r} \ .$$

However, now our object does depend on $\omega(K)$, though as we noted we can shift all activation times by a multiple of $z(h(r,k))$ and behavior during the time-steps we analyze is unchanged. So our total number of instances to consider is multiplied by $z(h(r,k))$, and is upper bounded by $2^{2r+1+2.9r} z(h(r,k)) \leq 2^{5r}$. □

Lemma 52. *For any (sufficiently large) r, the probability that T does not hit all (r,k)-instances is at most 2^{-3r}.*

Proof. Fix some (r,k)-instance. The probability that it is not hit within $h(r,k)$ time-steps is at most

$$\prod_{j\in\mathcal{F}}(1-\frac{1}{3c}) \leq e^{-\frac{|\mathcal{F}|}{3c}} \leq e^{-\frac{cr}{6}} = 2^{-\frac{cr}{6\ln 2}} \ .$$

Hence, if we set $c = 34$, by a union bound the probability that any (r,k)-instance is not hit is at most $2^{5r}\cdot 2^{-\frac{34r}{6\ln 2}} \leq 2^{-3r}$. □

We can now prove our main theorem on unbounded transmission schedules (Theorem 46):

Proof. By Lemma 52 and a union bound over r, the probability that T does not hit all instances is at most $\sum_{r\in\mathbb{N}} 2^{-3r} < 1$. Therefore T is an unbounded transmission schedule of delay h with non-zero probability, so such an object must exist. □

4.4 Algorithms for Multiple Access Channels

Armed with our combinatorial objects, our algorithms are now extremely simple, and are the same for multiple access channels as for multi-hop radio networks.

Let:

- W be an $(\tilde{n},\tilde{D})$-block transmission schedule of delay h_1, where $\tilde{n}\geq n$ and $\tilde{D}$ are parameters to be chosen later and $h_1(k) = O(k\log\tilde{D}\log\log\tilde{D})$,
- X be an unbounded universal synchronizer of delay h_2, where $h_2(r,k) = O\left(\frac{r\log k}{\log\log k}\right)$,
- Y be an unbounded transmission schedule of delay h_3, where $h_3(r,k) = O(r\log\log k)$
- Z be an $(\tilde{n},\frac{\tilde{n}}{\tilde{D}})$-selective family of size $O(\frac{\tilde{n}}{\tilde{D}}\log\tilde{D})$.

Our algorithms are simply applications of these objects. We give the algorithms for blind networks first, since their application is the most straightforward:

Algorithm 9 Blind wake-up at a node v

for j from 1 to ∞, in time-step $\omega(v) + j$, **do**
  v transmits iff $X(v)_j = 1$
end for

Theorem 53. *Algorithm 9 performs wake-up in multiple access channels in time* $O\left(\frac{n \log L \log n}{\log \log n}\right)$*, without knowledge of* n *or* L*.*

Proof. By the definition of an unbounded universal synchronizer, there is some time-step within

$$h_2(r, n) = O\left(\frac{r \log n}{\log \log n}\right) = O\left(\frac{n \log L \log n}{\log \log n}\right)$$

time-steps of the first activation in which only one node transmits, and this completes wake-up. □

Algorithm 10 Blind broadcasting at a node v

for j from 1 to ∞, in time-step j, **do**
  v transmits iff $Y(v, \omega(v))_j = 1$
end for

Theorem 54. *Algorithm 10 performs broadcasting in multiple access channels in time* $O(n \log L \log \log n)$*, without knowledge of* n *or* L*.*

Proof. By the definition of an unbounded transmission schedule, there is some time-step within $h_3(r, n) = O(r \log \log n) = O(n \log L \log \log n)$ time-steps of the first activation in which only one node transmits, and this completes broadcasting. □

The natural application of the block transmission schedule is as follows:

Algorithm 11 Broadcasting at a node v - main process

for j from 1 to $h_1(n)$, in time-step $B\omega(v) + j$, **do**
  v transmits iff $W(v)_j = 1$
end for

However, recall that unlike universal synchronizers and transmission schedules, we had some restrictions on our definition of k-instances that our block

transmission schedule would hit. In particular, we require that $k_B \geq \frac{\tilde{n}}{\tilde{D}}$, i.e. at least $\frac{\tilde{n}}{\tilde{D}}$ nodes are active by time-step B. To ensure that this is always the case, we use a *background process* which is run concurrently with the main process (e.g. by performing the main process in odd time-steps and the background process in even ones). This background process is simply a repeated application of the $(\tilde{n}, \frac{\tilde{n}}{\tilde{D}})$-selective family Z:

Algorithm 11 Broadcasting at a node v - background process

for time-step j from start of next selective family after v wakes up, to $h_1(n)$, **do**

v transmits iff $Z(v)_{j \bmod |Z|} = 1$

end for

I.e. when nodes wake up, they wait until the start of the next selective family application and participate from then on. By the definition of a selective family, within $2|Z| = O(\frac{\tilde{n}}{\tilde{D}} \log \tilde{D}) \leq B$ time-steps, there will have been a successful transmission, or at least $\frac{\tilde{n}}{\tilde{D}}$ nodes will be active. Thus, we satisfy the necessary condition for k-instances.

Theorem 55. *If $\tilde{n} \geq n$, Algorithm 11 performs broadcasting in multiple access channels in time $O((\frac{\tilde{n}}{\tilde{D}} + n) \log \tilde{D} \log \log \tilde{D})$.*

Proof. Due to the background process, by time-step B at least $\frac{\tilde{n}}{\tilde{D}}$ nodes will be active. Then, by the definition of an $(\tilde{n}, \tilde{D})$-block transmission schedule, there is some time-step within the first $h_1(n) = O(k \log \tilde{D} \log \log \tilde{D})$ in which only one node transmits, and this completes broadcasting. So, total time is at most $O(B + n \log \tilde{D} \log \log \tilde{D}) = O((\frac{\tilde{n}}{\tilde{D}} + n) \log \tilde{D} \log \log \tilde{D})$. □

Unlike Theorems 53 and 54, we do not obtain a running time improvement in multiple access channels, since an $O(n)$-time algorithm already exists (assuming linear labels) [15], and this is what Theorem 55 will give if we set $\tilde{n} = n$, $\tilde{D} = 1$. However, this formulation allows us to transfer our results directly to multi-hop radio networks, where we do achieve a running time improvement.

4.5 Analysis for Multi-hop Radio Networks

To see how our results on multiple access channels (Theorems 53, 54, and 55) transfer directly to multi-hop radio networks, we apply an analysis method that reduces a multi-hop instance to a succession of single-hop instances.

The idea is that we fix a shortest path $p = (p_0, p_1, \ldots p_d)$ from some *source* node s to some *target* node v, and then classify all nodes into *layers* depending on the furthest node along the path to which they are an in-neighbor, i.e., layer $L_i = \{u : \max j \text{ such that } (u, p_j) \in E = i\}$. Note that some nodes may not be in any layer; these can be discounted from the analysis.

We wish to quantify how long a layer can remain *leading*, that is, the furthest layer to contain awake nodes. The key point is that we can regard these layers as multiple access channels: though they are not necessarily cliques, all we need is for a single node in the layer to transmit and then the layer ceases to be leading. Once the final layer ceases to be leading, the target node v must be informed, and since this node was chosen arbitrarily we obtain a time-bound for informing the entire network. Thereby the problem is reduced to a sequence of at most D single-hop instances, whose sizes sum to at most n (since each node can be in at most one layer).

Therefore we can use the following lemma (Lemma 56) to analyze how our algorithms perform in radio networks:

Lemma 56. *Let $h : [n] \to \mathbb{N}$ be a non-decreasing function, and define Y to be the supremum of the function $\sum_{i=1}^{D} h(k_i)$, where non-negative integers k_i satisfy the constraint $\sum_{i=1}^{n} k_i \leq n$. If a broadcast or wake-up protocol ensures that any layer of size k remains leading for no more than $h(k)$ time-steps, then all nodes wake up within Y time-steps.*

Proof. Let $k_i = |L_i|$. Layer L_d must be leading (and thus node v active) once no other layers are leading, and so this occurs within $\sum_{i=1}^{d} h(k_i)$ time-steps after layer L_1 becomes leading. Since $\sum_{i=1}^{d} h(k_i) \leq \sum_{i=1}^{D} h(k_i)$ and $\sum_{i=1}^{D} k_i \leq n$, this is no more than Y time-steps.

Since v was chosen arbitrarily, all nodes must be active within Y time-steps of s becoming active. □

Theorem 57. *Algorithm 9 performs wake-up in radio networks in time $O(\frac{n \log L \log n}{\log \log n})$, without knowledge of n or L.*

Proof. By Theorem 53, any layer of size k remains leading for no more than $h(k)$ time-steps, where $h(k) = O(\frac{k \log L \log k}{\log \log k})$. $\sum_{i=1}^{D} h(k_i)$ is then maximized by setting $k_1 = n$ and $k_i = 0$ for every $i > 1$. So, by Lemma 56, wake-up is performed for the entire radio network in $O(\frac{n \log L \log n}{\log \log n})$ time. □

Theorem 58. *Algorithm 10 performs broadcasting in radio networks in time $O(n \log L \log \log n)$, without knowledge of n or L.*

Proof. By Theorem 54, any layer of size k remains leading for no more than $h(k)$ time-steps, where $h(k) = O(k \log L \log\log k)$. $\sum_{i=1}^{D} h(k_i)$ is then maximized by setting $k_1 = n$ and $k_i = 0$ for $i > 1$. So, by Lemma 56, broadcasting is performed for the entire radio network in $O(n \log L \log\log n)$ time. □

Recall that the combinatorial objects used for Algorithm 11 depended on parameters n and D, which for multiple access channels we treated as free variables. In multi-hop radio networks, they of course denote network size and diameter, and these are the values we use for the combinatorial objects.

Theorem 59. *Algorithm 11, using $\tilde{n} = n$ and $\tilde{D} = D$, performs broadcasting in radio networks in time $O(n \log D \log\log D)$.*

Proof. By Theorem 55, any layer of size k remains leading for no more than $h(k)$ time-steps, where $h(k) = O((\frac{n}{D} + k) \log D \log\log D)$. The value of the sum $\sum_{i=1}^{D} h(k_i)$ is $O(n \log D \log\log D)$ for any choice of k_i. So, by Lemma 56, broadcasting is performed for the entire radio network in $O(n \log D \log\log D)$ time. □

4.6 Discussion and Open Problems

We present the fastest known deterministic algorithms for broadcasting and wake-up, both with and without parameter knowledge.

None of these algorithms are known to be optimal. The best lower bound for both broadcasting and wake-up is $\Omega(n \log D)$ [15]; our broadcasting algorithm therefore comes within a log-logarithmic factor, but our wake-up algorithm remains a logarithmic factor away. Algorithms meeting the lower bound would be interesting, as would an $\Omega(n \log D)$ lower bound for *undirected* networks (currently the bound only holds for directed networks).

As mentioned, almost all deterministic broadcasting protocols with subquadratic complexity have relied on selective families or variants thereof, and have been non-explicit results. Our work here is also non-explicit, but while this is standard for deterministic radio network algorithms, it may present more of an issue in blind networks, since our combinatorial structures are infinite (though if parameters *are* known, the algorithms can be run using truncated structures to avoid this issue). It is not clear how the protocols we present could be performed by devices with bounded memory, and as such this work is more of a proof-of-concept than a practical algorithm. However, it is possible that

our method could be adapted so that nodes' behavior could be generated by a finite-size (i.e. a function of ID) program; this would be a natural and interesting extension to our work, and would overcome the problem. Of course, an efficient explicit algorithm would be even more desirable, but for now this seems out of reach: the best explicit algorithm known to date, even with parameter knowledge, is a long way from optimality [38].

Another issue with Algorithms 9 and 10 in blind networks is that nodes must perform the protocol indefinitely, and never become aware that broadcasting has been successfully completed. However, this is unavoidable in the blind model: Chlebus et al. [10] prove that *acknowledged* broadcasting without parameter knowledge is impossible.

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 [28]. Collision detection does remove the requirement of spontaneous transmissions for the use of the $O(n)$ algorithm of [10], but a synchronized global clock would still be required. Wake-up can also be considered in this model, though only if hearing a collision does not wake up a node, since otherwise the problem is trivial.

Chapter 5

Randomized Blind Broadcasting

In this chapter we turn our attention to randomized algorithms for broadcasting.

5.1 Related Work

The first major randomized broadcasting result was a seminal paper of Bar-Yehuda et al. [4], who designed the DECAY protocol and used it to obtain an almost optimal randomized broadcasting algorithm achieving the running time of $O((D + \log n) \cdot \log n)$, and succeeding with high probability. This bound was later improved by Czumaj and Rytter [23], and independently Kowalski and Pelc [44], who gave randomized broadcasting algorithms that complete the task in $O(D \log \frac{n}{D} + \log^2 n)$ time with high probability. This running time matched a known $\Omega(D \log \frac{n}{D} + \log^2 n)$ lower bound for the task [2, 46]. All of these results hold for directed networks as well as undirected ones.

More recently, Ghaffari, Haeupler and Khabbazian [28] showed that collision detection can be used to surpass this lower bound, attaining an $O(D + \log^6 n)$ time algorithm. In 2016, work by Haeupler and Wajc [34] demonstrated that even without collision detection, the lower bound could be beaten assuming *spontaneous transmissions* were permitted; that is, nodes have access to a global clock and are allowed to transmit before receiving the source message. We discuss and extend this work in Chapter 7. However, these algorithms only work in undirected networks.

All of these results also intrinsically require parameter knowledge, and algorithms that do not require such knowledge have been little studied. The closest analogue in the literature is the work of Jurdzinski and Stachowiak [39], who give algorithms for wake-up in single-hop radio networks under a wide range of node knowledge assumptions. Their Use-Factorial-Representation algorithm is the most relevant; the running time is given as $O((\log n \log\log n)^3)$ for high-probability wake-up with a global clock (a slightly stronger task than broadcasting) in single-hop networks, but a similar analysis as we present here would demonstrate that the algorithm also performs broadcasting in multi-hop networks in $O((D + \log n)\log^2 \frac{n}{D} \log^3 \log \frac{n}{D})$ time.

5.2 Our Results

We present a randomized algorithm for broadcasting in (directed or undirected) blind networks without collision detection which succeeds with high probability within time $O(D \log \frac{n}{D} \log^2 \log \frac{n}{D} + \log^2 n)$. This improves over the $O((D + \log n)\log^2 \frac{n}{D} \log^3 \log \frac{n}{D})$ time that could be obtained by applying our analysis method to the Use-Factorial-Representation algorithm of Jurdzinski and Stachowiak [39], and comes within a poly-log log factor of the $\Omega(D \log \frac{n}{D} + \log^2 n)$ lower bound.

We also present an algorithm for directed blind networks with collision detection, whose $O(D \log \frac{n}{D} \log\log\log \frac{n}{D} + \log^2 n)$ running time comes even closer to the lower bound (we note that, to our knowledge, it has not been proven that the lower bound still holds in this setting[1], though it would be very surprising if it did not).

Finally, we make the observation that in undirected networks with collision detection, the $O(D + \log^6 n)$-time algorithm of [28] can be simulated without parameter knowledge at no extra cost.

5.3 Overview of Approach

We will begin by applying our analysis approach to show how the existing algorithms for networks with parameter knowledge [4, 23] achieve their running times, and then describe the changes that need to be made for blind networks.

[1] Newport [52] proves a similar lower bound, but in a setting where nodes remain dormant until hearing the source message, i.e. are not woken by collisions.

5.3.1 Analysis of Decay

The idea of the classic DECAY protocol is to have nodes transmit with exponentially decaying probabilities, between $\frac{1}{2}$ and $\frac{1}{n}$. Then, in some step, the inverse of the transmission probability will be within a constant factor of the number of participating nodes, in which case, by Lemma 62, we have a constant probability of successful transmission.

One *round* of the DECAY protocol is as follows (Algorithm 12):

Algorithm 12 DECAY at a node v

for $i = 1$ **to** $\log n$ **do**
  v transmits its message with probability 2^{-i}.
end for

We analyze the DECAY protocol by fixing some uninformed node v with some number δ of active in-neighbors. There will be at least one time-step during the round of DECAY such that $2^{i-1} \leq \delta \leq 2^i$. Then, the expected number of in-neighbors of v that transmit is $\delta \cdot 2^{-i}$, which is in the interval $[\frac{1}{2}, 1]$. Since all nodes are transmitting with probability at most $\frac{1}{2}$, we can apply Lemma 25 to find that exactly one transmits with probability at least $\frac{1}{4}$. So, every $\log n$ time-steps we independently have probability at least $\frac{1}{4}$ of a successful transmission. The total time taken to inform v is then stochastically majorized by $\log n + X$, where X is a geometric random variable with parameter $\frac{1}{4 \log n}$.

What we now need to obtain a running time for broadcasting in multi-hop networks is a randomized analogue of Lemma 56, where instead of some fixed bound h on the time taken to inform a single-hop instance, we instead have a geometric random variable. We achieve this by exploiting a lemma from [23], which gives a concentration bound on the sum of geometric random variables:

Lemma 60 (Lemma 3.5 of [23])**.** *Let $X_1, \ldots, X_D$ be a sequence of independent integer-valued random variables, each X_i geometrically distributed with parameter p_i, $0 < p_i < 1$. For every i, let $\mu_i = 1/p_i$, and let M be the set of unique μ_i, i.e. $M = \{\mu_i : 1 \leq i \leq D\}$. If $\sum_{i=1}^{d} \mu_i \leq N$, then for any positive real β,*

$$\mathbf{Pr}\left[\sum_{i=1}^{D} X_i \leq 2 \cdot N + 8 \ln(|M|/\beta) \cdot \sum_{z \in M} z\right] \geq 1 - \beta \ .$$

We adapt this slightly for our purposes:

Corollary 61. *Let $X_1, \ldots, X_D$ be a sequence of independent integer-valued random variables, each X_i geometrically distributed with parameter $1/\mu_i$, $\mu_i \in \mathbb{N}$. Let μ_{max} be the maximum μ_i. If $\sum_{i=1}^{D} \mu_i \leq N$, then for any positive $\beta \leq \frac{1}{\log \mu_{max}}$,*

$$\mathbf{Pr}\left[\sum_{i=1}^{D} X_i \leq 4N + 65\mu_{max} \ln(1/\beta)\right] \geq 1 - \beta \ .$$

Proof. Let M be the set of all powers of 2 up to μ_{max}, i.e. $M = \{2^i : 1 \leq i \leq \lceil \log \mu_{max} \rceil\}$; then $|M| = \lceil \log \mu_{max} \rceil$ and $\sum_{z \in M} z \leq 4\mu_{max}$. For all i, let X_i' be a geometric random variable with μ_i' equal to μ_i rounded up to the next power of 2 (and $p_i' = 1/\mu_i'$ accordingly). Note that X_i' majorizes X_i. Then, by Lemma 60, for positive $\beta \leq \frac{1}{\log \mu_{max}}$,

$$\begin{aligned}
1 - \beta &\leq \mathbf{Pr}\left[\sum_{i=1}^{D} X_i' \leq 2 \cdot \sum_{i=1}^{D} \mu_i' + 8 \ln(|M|/\beta) \cdot \sum_{z \in M} z\right] \\
&\leq \mathbf{Pr}\left[\sum_{i=1}^{D} X_i' \leq 4 \cdot \sum_{i=1}^{D} \mu_i + 8 \ln(\lceil \log \mu_{max} \rceil/\beta) \cdot 4\mu_{max}\right] \\
&\leq \mathbf{Pr}\left[\sum_{i=1}^{D} X_i \leq 4N + 65\mu_{max} \ln(1/\beta)\right] \ .
\end{aligned}$$

□

Using this bound, we have a recipe for getting from these geometric random variables to a running time for a broadcasting algorithm:

Lemma 62. *Let $\mu : [D] \times [n] \to [n]$ be a function which is non-decreasing in its second argument. Let N be the maximum of $\sum_{d=1}^{D} \mu_{max}(d, \delta_d)$, subject to $\sum_{d=1}^{D} \delta_d \leq n$, and let μ_{max} be the maximum value of μ. If an algorithm for broadcasting guarantees that any node v at distance d_v from the source with neighborhood of size δ_v is informed within time X_v of a neighbor being informed, where X_v is stochastically majorized by a geometric random variable with parameter $\frac{1}{\mu(d_v, \delta_v)}$, then with probability at least $\frac{1}{n}$, broadcasting is completed in the whole network within $4N + 91\mu_{max} \log n$ time.*

Proof. Fix some arbitrary target node u and some shortest (s, u) path $p = (s = p_0, p_1, \ldots, p_{d_v} = v)$. Classify nodes into layers as follows: let layer L_i be the set of all nodes whose latest out-neighbor path node is p_i. We call a layer *leading* if it is the furthest layer containing an active node. We wish to bound the time

t_i that a layer L_i can remain leading, since the total time taken to inform u is then at most $\sum_{i=1}^{D} t_i$. This time t_i is stochastically majorized by geometric random variable with parameter $\frac{1}{\mu(i,|L_i|)}$, since only nodes in the intersection of u's neighborhood and L_i can participate in informing u while L_i is leading. Then, applying Corollary 61,

$$\mathbf{Pr}\left[\sum_{i=1}^{D} t_i \leq 4N + 65\mu_{max}\ln(n^2)\right] \geq 1 - n^{-2} \ ,$$

i.e. with probability at least $1 - n^{-2}$, u is informed within time $4N + 91\mu_{max}\log n$, and taking a union bound over all nodes u, the whole network is informed within this time with probability at least $1 - n^{-1}$. □

We will be using Lemma 62 to analyze our blind broadcasting algorithms, and it is somewhat more complex than is necessary to analyze Decay; the function μ in Lemma 62 allows the bound for single-hop instances to depend upon number of active nodes and distance from the broadcasting source s, which is unnecessary for Decay. However, using the same analysis framework will highlight the differences between our algorithms and preceding work.

Returning to Decay, we found that time to inform single-hop instances is majorized by $\log n + X$, where X is a geometric random variable with parameter $\frac{1}{4\log n}$. So, by plugging $\mu(d,\delta) = 4\log n$ for all d, δ into Lemma 62, we find that broadcasting is completed in $D\log n + 4\sum_{i=1}^{D} 4\log n + 91 \cdot 4\log^2 n = O((D + \log n)\log n)$ time with high probability.

5.3.2 Randomized Selecting Sequences

As described, a phase of Decay 'guesses' active in-neighborhood size δ in increasing order: in the first time-step it is successful if $1 \leq \delta \leq 2$, in the second inf $2 \leq \delta \leq 4$, and so on for all $\log n$ 2-factor ranges of δ between 1 and n. The order is important, since δ can increase during the course of the round of Decay as new nodes wake up; for example if guesses were made in reverse order, then a situation in which $\delta = 1$ for the first $\frac{\log n}{2}$ time-steps, and then $\delta = n - 1$ for the remaining ones, would prevent any step from having good success probability.

This becomes an issue if we wish to alter the *rate* with which ranges of δ are guessed. Currently all ranges are guessed once every $\log n$ time-steps, i.e. we use probability 2^{-i} with rate $\frac{1}{\log n}$ for all $i \in [\log n]$. We will see that by adjusting these rates we can improve the algorithmic running time. If we do

so, however, it becomes less obvious in what order we should make our guesses. However, as noticed by Czumaj and Rytter [23], if we use a *random* order, or *selecting sequence*, then we still guess correctly with frequency roughly equal to the rate, with high probability. We can implement a random selecting sequence by having the source node generate it and append it to the source message.

The DECAY algorithm with random selecting sequence would look as follows:

Algorithm 13 DECAY - random selecting sequence

for each $j \in [T]$, s generates a random variable x_j uniformly from $[\log n]$.
s appends variables x_j to the source message.
for j from 1 to T, in time-step j, **do**
    active nodes transmit with probability 2^{-x_j}.
end for

Here T is some estimate for how long the algorithm should be run, which we would set to be $O((D + \log n) \log n)$. To analyze this algorithm, we note that there is a $\frac{1}{\log n}$ probability of choosing the correct x_j, in which case we have a successful transmission with probability at least $\frac{1}{4}$ as before. So, the time taken for a single-hop instance is majorized by a geometric random variable with parameter $\frac{1}{4 \log n}$, and we can apply Lemma 62 and find that broadcasting is performed in $O((D + \log n) \log n)$ time as before.

5.3.3 Rates for Optimal Broadcasting

Now that we are using a randomized selection sequence, we can easily adjust the rates with which we choose each neighborhood size range. That is, we can modify Algorithm 13 so that the distribution of the x_j is no longer uniform. We will still need the support of the distribution to be $[\log n]$, since we need to be able to inform any neighborhood between sizes 1 and n eventually. However, the observation made by Czumaj and Rytter [23] which allowed them to achieve optimal $O(D \log \frac{n}{D} + \log^2 n)$ time was that neighborhoods larger than $\frac{n}{D}$ can be chosen less frequently, since there cannot be as many of them.

Specifically, we use a distribution Y which is a simplified version of that given in [23]:

$$\mathbf{Pr}[x_i = y] = \begin{cases} \frac{1}{4 \log \frac{n}{D}}, & \text{if } 1 \leq y < 2 \log \frac{n}{D} \\ \frac{1}{4 \min\{y^2, \log n\}}, & \text{if } 2 \log \frac{n}{D} \leq y \leq \log n \end{cases},$$

with $x_i = 0$ with any remaining probability. This is a well-defined distribution since the sum of the rates is at most 1 (and we can let x_i take an arbitrary value with any remaining weight):

$$\sum_{y=1}^{2\log\frac{n}{D}-1} \frac{1}{4\log\frac{n}{D}} + \sum_{y=2\log\frac{n}{D}}^{\log n} \left(\frac{1}{4y^2} + \frac{1}{4\log n}\right) \leq \frac{1}{2} + \sum_{y=2}^{\infty} \frac{1}{4y^2} + \frac{1}{4} \leq 1 \ .$$

We can use this distribution Y for a broadcasting algorithm as follows (Algorithm 14):

Algorithm 14 Broadcasting [23]

for each $j \in [T]$, s generates a random variable x_j from distribution Y.
s appends variables x_j to the source message.
for j from 1 to T, in time-step j, **do**
  active nodes transmit with probability 2^{-x_j}.
end for

Our analysis framework can then be applied to this algorithm; by recalling that when $2^{x_j} = \Theta(\delta)$ we have a constant probability of successful transmission, we have a function μ to plug into Lemma 62 which now depends on in-neighborhood size:

$$\mu(d,\delta) = \begin{cases} O(\log\frac{n}{D}), & \text{if } 1 \leq \delta < (\frac{n}{D})^2 \\ O(\min\{\log^2\delta, \log n\}), & \text{otherwise} \end{cases} .$$

Lemma 62 tells us that we complete broadcasting in time $O(N+\mu_{max})$; here μ_{max} is clearly $O(\log n)$. N is $O(D\log\frac{n}{D})$, since this is clearly an upper bound for the contribution of the layers with $\delta < (\frac{n}{D})^2$, and the contribution of larger layers is at most $O(\frac{D^2}{n}\log^2(\frac{n^2}{D^2})) = O(D\cdot\frac{D}{n}\log^2\frac{n}{D}) = O(D)$.

So, broadcasting is completed in $O(D\log\frac{n}{D} + \log^2 n)$ time, succeeding with high probability.

5.3.4 Difficulties in Blind Networks

The $O(D\log\frac{n}{D} + \log^2 n)$ time algorithm of [23] is optimal (assuming no collision detection or spontaneous transmissions), due to the matching lower bound of [46]. However, it intrinsically requires knowledge of parameters n and D, since

both the time bound T and the distribution X depend upon them. In blind networks, we must find some way of avoiding this dependence. For T this is not a major problem; we can simply use a 'double-and-test' type technique, where we try the algorithm with $T = 1, 2, 4, 8, \ldots$ until we are successful.

Finding a distribution Y which does not depend on n and D is more difficult, and this is the main technical challenge of the work. We will see that we can still approach the optimal $O(D \log \frac{n}{D} + \log^2 n)$ running time using a distribution which is only dependent upon T, and, when collision detection is available, distance d from the source (though the latter causes more complications, as we will see shortly).

Our algorithm for blind broadcasting without collision detection, where node transmission probabilities only depend on the doubling parameter T, would fit into the following framework (Algorithm 15):

Algorithm 15 Blind Broadcast Framework - Take 1

for $t = 1$ to ∞ **do**
  let $T = 2^t$
  for each $j \in [T]$, s generates a random variable x_j from distribution Y.
  s appends variables x_j to the source message.
  for j from 1 to T, in time-step j, **do**
    active nodes v transmit with probability 2^{-x_j}.
  end for
  reset time-step numbers and set non-source nodes to inactive.
end for

As one can see, this framework is similar to Algorithm 14 except that T is iteratively doubling, and that we must set Y to be independent of parameters n and D. However, this is not the final framework we will use, because, as mentioned, we can improve running time when collision detection is available by using nodes' distance from the source.

A simple application of beep waves, as we saw in Chapter 3, can inform nodes of their exact distance from the source node within $O(D)$ time, even in directed networks: in time-step 1, the source node emits a 'beep' (a transmission with arbitrary content), and in every subsequent step, all nodes who hear either a transmission or a collision in a time-step themselves 'beep' in the next time-step. In this way, the wave of beeps emanates out from the source, one distance hop per time-step, and the time-step number in which a node hears its first beep

is equal to its distance from the source. If we perform this beep-wave procedure before our main algorithm, we can assume every node v knows its distance d_v from the source (actually, since we cannot tell when the procedure has ended, we must use time multiplexing, for example performing beep waves during odd time-steps and the main algorithm during even ones, but this does not affect asymptotic running time).

We then wish to have nodes base their transmission probabilities on d_v as well as T. However, we cannot incorporate this information into the distribution Y, since Y is global, whereas d_v is different for each node. So, we must instead use d_v when nodes choose their local transmission probabilities, i.e. where they previously used probability 2^{-x_j}, we now need some expression involving d_v.

So, our final framework looks as follows (Algorithm 16):

Algorithm 16 Blind Broadcast Framework - Take 2

for $t = 1$ to ∞ **do**
  let $T = 2^t$
  s randomly generates a sequence $S \in [C]^T$ with uniform i.i.d. entries.
  for each $j \in [T]$, s generates a random variable x_j from distribution $Y_{S_j,T}$.
  s appends S and variables x_j to the source message.
  for j from 1 to T, in time-step j, **do**
    active nodes v transmit with probability $p_{S_j,d_v,T}(x_j)$.
  end for
  reset time-step numbers and set non-source nodes to inactive.
end for

Here we have a constant number c of different global distributions, dependent on T, each of which is equipped with a local probability function, dependent on d_v and T. We call this pair of global distribution and local probability function a *protocol*. The protocol to use in each time step is randomly chosen by the source, and these choices are appended to the source message along with the variables x_j.

5.4 Protocols

We now specify the protocols, that is, pairs of global distribution and local probability function, which we use in Algorithm 16 to perform broadcast.

5.4.1 Networks Without Collision Detection

In networks without collision detection, we use two different protocols, i.e. $c = 2$. We will call the protocol designed to account for most circumstances **General-Broadcast**; in this protocol, the source 'guesses' a neighborhood size from 1 to ∞ in each time-step, with a probability that decreases in neighborhood size in order for the total probability to sum to at most 1. Transmission probabilities are independent of parameter T. If we used only this protocol, we would obtain a running time of $O((D + \log n) \log \frac{n}{D} \log^2 \log \frac{n}{D})$. In low diameter networks (when $D < \log n$), we improve upon this with **Shallow-Broadcast** protocol, which informs networks of low diameter in $O(\log^2 n)$ time by assuming that $T \approx \log^2 n$ and using this to approximate the maximum in-neighborhood size to account for.

Shallow-Broadcast Distribution Y_1 is given by $\mathbf{Pr}[x_j = y] = \frac{c_1}{\sqrt{T}}$ for all $y \in [\frac{\sqrt{T}}{c_1}]$. Probability function $p_{1,d_v,T}(x_j) = 2^{-x_j}$.

Lemma 63. *If $D \leq \log n$, **Shallow-Broadcast** performs broadcasting in $O(\log^2 n)$ time with high probability.*

Proof. We consider the iteration in which $(c_1 \log n)^2 \leq T \leq 2(c_1 \log n)^2$. Fix a time-step j, and let u be an inactive node with a set Δ of active neighbors, $\delta = |\Delta| \geq 1$. With probability $\frac{1}{c}$, j is a **Shallow-Broadcast** time-step. Then, with probability $\frac{c_1}{\sqrt{T}} \geq \frac{1}{\sqrt{2} \log n}$, x_j is chosen such that $2\delta \leq 2^{x_j} \leq 4\delta$, in which case $\frac{1}{4} \leq \sum_{u \in \Delta} \mathbf{Pr}[u \text{ transmits}] \leq \frac{1}{2}$, so by Lemma 25, $\mathbf{Pr}[v \text{ is informed}] \geq \frac{1}{4} \cdot 4^{-\frac{1}{4}} \geq \frac{1}{6}$. So, in each time-step, u is informed with probability at least $\frac{1}{c} \cdot \frac{1}{\sqrt{2} \log n} \cdot \frac{1}{6} \geq \frac{1}{9c \log n}$. Time taken to inform u is therefore stochastically majorized by a geometric random variable with parameter $\frac{1}{9c \log n}$. Using Lemma 62 with $\mu(d, \delta) = 9c \log n$ for all d, δ, we can conclude that the network will be informed within $4N + 91\mu_{max} \log n \leq 36cD \log n + 819c \log^2 n$ time with high probability. We set $c_1 = 30c$ to ensure that the iteration we analyze is sufficiently long, and thus broadcast is performed in $O(\log^2 n)$ time. □

General-broadcast Distribution Y_2 is given by $\mathbf{Pr}[x_j = y] = \frac{1}{3y \log^2 y}$ for all $y \in \mathbb{N}$ (and $x_j = 0$ with the remaining probability). Probability function $p_{2,d_v,T}(x_j) = 2^{-x_j}$.

We first check that Y_2 is a well-defined probability distribution, which is the case since $\sum_{y \in \mathbb{N}} \frac{1}{3y \log^2 y} \leq \frac{1}{2} + \int_2^\infty \frac{dy}{3y \log^2 y} \leq 1$.

Lemma 64. *If $D \geq \log n$, **General-Broadcast** performs broadcasting in $O(D \log \frac{n}{D} \log^2 \log \frac{n}{D})$ time with high probability.*

Proof. We consider the iteration in which

$$c_2 D \log \frac{n}{D} \log^2 \log \frac{n}{D} \leq T \leq 2c_2 D \log \frac{n}{D} \log^2 \log \frac{n}{D} \ .$$

Fix a time-step j, and let u be an uninformed node with a set Δ of informed neighbors, $\delta = |\Delta| \geq 1$. With probability $\frac{1}{c}$, j is a **General-Broadcast** time-step. Then, with probability at least $\frac{1}{3 \log(4\delta) \log^2 \log(4\delta)} \geq \frac{1}{6 \log \delta \log^2 \log \delta}$, x_j is chosen such that $2\delta \leq 2^{x_j} \leq 4\delta$, in which case $\frac{1}{4} \leq \sum_{u \in \Delta} \mathbf{Pr}[u \text{ transmits}] \leq \frac{1}{2}$, so by Lemma 25, $\mathbf{Pr}[v \text{ is informed}] \geq \frac{1}{4} \cdot 4^{-\frac{1}{4}} \geq \frac{1}{6}$. So, in each time-step, u is informed with probability at least $\frac{1}{c} \cdot \frac{1}{6 \log \delta \log^2 \log \delta} \cdot \frac{1}{6} \geq \frac{1}{36c \log \delta \log^2 \log \delta}$. Time taken to inform u is therefore stochastically majorized by a geometric random variable with parameter $\frac{1}{36c \log \delta \log^2 \log \delta}$. Using Lemma 62 with $\mu(d, \delta) = 36c \log \delta \log^2 \log \delta$ for all d, δ, we can conclude that the network will be informed within

$$\begin{aligned} 4N + 91\mu_{max} \log n &\leq 144cD \log \frac{n}{D} \log^2 \log \frac{n}{D} + 3276 \log^2 n \log^2 \log n \\ &\leq 3500cD \log \frac{n}{D} \log^2 \log \frac{n}{D} \end{aligned}$$

time with high probability. We set $c_2 = 3500c$ to ensure that the iteration we analyze is sufficiently long, and thus broadcast is performed in $O(D \log \frac{n}{D} \log^2 \log \frac{n}{D})$ time. □

To perform broadcasting in networks without collision detection, we apply the framework of Algorithm 16 using the **Shallow-Broadcast** and **General-Broadcast** protocols.

Theorem 65. *Broadcasting can be performed in networks without collision detection in $O(D \log \frac{n}{D} \log^2 \log \frac{n}{D} + \log^2 n)$ time, with high probability.*

Proof. Follows from Lemmas 63 and 64. □

5.4.2 Undirected Networks With Collision Detection

For directed networks, as mentioned, we first perform a beep wave which allows each node v to use its distance d_v from the source. This is the *only* way in which we use collision detection; afterwards we apply Algorithm 16 as before.

We add two new protocols to the two already defined, i.e. we now use $c = 4$. The main new protocol is **Deep-Broadcast**, which assumes that $T \approx D \log \frac{n}{D} \log\log\log \frac{n}{D}$ and $d_v \approx D$, and uses this to approximate $(\frac{n}{D})^2$, the largest neighborhood size for which it accounts. This only works well for nodes which do indeed have $d_v \approx D$ and $\delta_v \leq (\frac{n}{D})^2$, but for our analysis method this covers most nodes of importance, and we use **General-Broadcast** to deal with the remaining nodes. By including **Deep-Broadcast** we speed up broadcasting to $O(D \log \frac{n}{D} \log\log\log \frac{n}{D})$ when $D > \log n \log^2 \log n$, but when D is below this, the running time of **General-Broadcast** still dominates. So, we also add **Semi-Shallow-Broadcast**, which works quickly for networks with $\log n \leq D \leq \log n \log^2 \log n$.

Semi-Shallow-Broadcast Distribution $Y_{3,T}$ is given by

$$\mathbf{Pr}\left[x_j = y\right] = \begin{cases} \sqrt{\frac{c_3^2 \log^2 T \log\log T}{2T}}, & \text{if } 1 \leq y \leq \sqrt{\frac{T}{c_3^2 \log^2 T \log\log T}} \\ \frac{1}{3y \log\log T}, & \text{if } \sqrt{\frac{T}{c_3^2 \log^2 T \log\log T}} < y \leq \sqrt{\frac{T \log^2 T}{c_3^2 \log\log T}} \end{cases} .$$

We let $x_j = 0$ with any remaining probability.

Probability function $p_{3,d_v,T}$ is given by $p_{3,d_v,T}(x_j) = 2^{-x_j}$.

We first check that the distribution $Y_{3,T}$ is well defined, which is the case since

$$\sum_{y=1}^{\sqrt{\frac{T}{c_3^2 \log^2 T \log\log T}}} \sqrt{\frac{c_3^2 \log^2 T \log\log T}{2T}} < \frac{3}{4} ,$$

and

$$\begin{aligned} \sum_{y=\sqrt{\frac{T}{c_3^2 \log^2 T \log\log T}}+1}^{\sqrt{\frac{T \log^2 T}{c_3^2 \log\log T}}} \frac{1}{3y \log\log T} &\leq \int_{\sqrt{\frac{T}{c_3^2 \log^2 T \log\log T}}}^{\sqrt{\frac{T \log^2 T}{c_3^2 \log\log T}}} \frac{dy}{3y \log\log T} \\ &\leq \frac{\ln(\log^2 T)}{3 \log\log T} < \frac{1}{4} . \end{aligned}$$

Lemma 66. *If $\log n \leq D \leq \log n \log^2 \log n$, **Semi-Shallow-Broadcast** performs broadcasting in $O(D \log n \log\log\log n)$ time with high probability.*

Proof. We consider the iteration in which

$$c_3 D \log n \log\log\log n \leq T \leq 2c_3 D \log n \log\log\log n \ .$$

Fix a time-step j, and let u be an uninformed node with a set Δ of informed neighbors, $\delta = |\Delta| \geq 1$. With probability $\frac{1}{c}$, j is a **Semi-Shallow-Broadcast** time-step. The probability that x_j is chosen such that $2\delta \leq 2^{x_j} \leq 4\delta$ is at least

$$\begin{aligned}
\sqrt{\frac{c_3^2 \log^2 T \log\log T}{2T}} &\geq \sqrt{\frac{c_3^2 \log^2(2c_3 D \log n \log\log\log n) \log\log(2c_3 D \log n \log\log\log n)}{4c_3 D \log n \log\log\log n}} \\
&\geq \sqrt{\frac{c_3^2 \log^2 \log n \log\log\log n}{4c_3 D \log n \log\log\log n}} \\
&\geq \sqrt{\frac{c_3 \log^2 \log n}{4 \log^2 n \log^2 \log n}} = \frac{\sqrt{c_3}}{2 \log n} \ ,
\end{aligned}$$

if $\log(4\delta) \leq \sqrt{\frac{T}{c_3^2 \log^2 T \log\log T}}$, and

$$\frac{1}{3\log(4\delta)\log\log T} \geq \frac{1}{3\log(4n)\log\log T} \geq \frac{1}{4\log n \log\log\log n} \ ,$$

otherwise. Note that it is not possible that $\log(4\delta) > \sqrt{\frac{T \log^2 T}{c_3^2 \log\log T}}$, since that would give

$$\begin{aligned}
\log(4\delta) &\geq \sqrt{\frac{c_3 D \log n \log\log\log n \log^2 T}{c_3^2 \log\log(c_3 D \log n \log\log\log n)}} \\
&\geq \sqrt{\frac{c_3 \log^2 n \log^2 \log n \log\log\log n}{4c_3^2 \log\log\log n}} \\
&\geq \frac{\log n \log\log n}{2\sqrt{c_3}} > \log(4n) \ ,
\end{aligned}$$

i.e. $\delta > n$, which is a contradiction.

So, an appropriate value of x_j is chosen with probability at least $\frac{1}{4\log n \log\log\log n}$, in which case $\frac{1}{4} \leq \sum_{u \in \Delta} \mathbf{Pr}[u \text{ transmits}] \leq \frac{1}{2}$, so by Lemma 25, $\mathbf{Pr}[v \text{ is informed}] \geq \frac{1}{4} \cdot 4^{-\frac{1}{4}} \geq \frac{1}{6}$. Therefore, in each time-step, u is informed with probability at least $\frac{1}{c} \cdot \frac{1}{4\log n \log\log\log n} \cdot \frac{1}{6} \geq \frac{1}{24c \log n \log\log\log n}$.

Time taken to inform u is therefore stochastically majorized by a geometric random variable with parameter $\frac{1}{24c \log n \log\log\log n}$. Using Lemma 62 with $\mu(d,\delta) = 24c \log n \log\log\log n$ for all d, δ, we can conclude that the network will be informed within

$$\begin{aligned} 4N + 91\mu_{max} \log n &\leq 96cD \log n \log\log\log n + 2184c \log^2 n \log\log\log n \\ &\leq 2280cD \log n \log\log\log n \end{aligned}$$

time with high probability. We set $c_3 = 2280c$ to ensure that the iteration we analyze is sufficiently long, and thus broadcast is performed in $O(D \log n \log\log\log n)$ time. □

Deep broadcast Distribution $Y_{4,T}$ is given by $\mathbf{Pr}\left[x_j = y\right] = \frac{1}{T}$ for all $y \in [T]$. Probability function $p_{4,d_v,T}$ is given by $p_{4,d_v,T}(x_j) = 2^{-\frac{1}{c_4 d_v \log\log \frac{T}{d_v}}}$.

Lemma 67. *If $D \geq \log n (\log\log n)^2$, **Deep-Broadcast** and **General-Broadcast** together complete broadcasting in $O(D \log \frac{n}{D} \log\log\log \frac{n}{D})$ time, with high probability.*

Proof. We consider the iteration in which

$$c_4^2 D \log \frac{n}{D} \log\log\log \frac{n}{D} \leq T \leq 2c_4^2 D \log \frac{n}{D} \log\log\log \frac{n}{D} \ .$$

Fix a time-step j, and let v be an uninformed node with a set Δ of informed neighbors, $\delta = |\Delta|$. Denote by d v's distance from the source.

If $d < \frac{D}{\log^2 \log \frac{n}{D}}$ or $\delta > (\frac{n}{D})^2$, we analyze **General-Broadcast** time-steps, and conclude, as in the proof of Lemma 64, that the time taken to inform v is stochastically majorized by a geometric random variable with parameter $\frac{1}{36c \log\delta \log^2 \log\delta}$.

Otherwise, we analyze **Deep-Broadcast** time-steps:

There is some real value $x' \in [1,T]$ such that $\sum_{u \in \Delta} p_{4,d_v,T}(x') = \frac{1}{2}$, since the value of this sum is continuous in x, is at least

$$\sum_{u \in \Delta} 2^{-\frac{1}{c_4 d_u \log\log \frac{T}{d_u}}} \geq 2^{-\frac{1}{c_4 (d-1) \log\log \frac{T}{d-1}}} > 2^{-1} = \frac{1}{2} \ ,$$

when $x = 1$, and is at most

$$\begin{aligned}\sum_{u\in\Delta} 2^{-\frac{T}{c_4 d_u \log\log\frac{T}{d_u}}} &\le \sum_{u\in\Delta} 2^{-\frac{c_4^2 D\log\frac{n}{D}\log\log\log\frac{n}{D}}{D\log\log\frac{c_4^2 D\log\frac{n}{D}\log\log\log\frac{n}{D}}{D}}} \\ &\le \sum_{u\in\Delta} 2^{-\frac{c_4^2 D\log\frac{n}{D}\log\log\log\frac{n}{D}}{2D\log\log\log\frac{n}{D}}} \\ &\le (\frac{n}{D})^2\cdot 2^{-\frac{1}{2}c_4^2\log\frac{n}{D}} = 2^{(2-\frac{1}{2}c_4^2)\log\frac{n}{D}} \\ &\le 2^{-1} = \frac{1}{2}\ ,\end{aligned}$$

when $x = T$.

Then the value of the sum at $x' + c_4 d\log\log\log\frac{n}{D} + 1$ is at least

$$\begin{aligned}\sum_{u\in\Delta} 2^{-\frac{x'+c_4 d\log\log\log\frac{n}{D}+1}{c_4 d_u\log\log\frac{T}{d_u}}} &\ge \sum_{u\in\Delta} 2^{-\frac{x'}{c_4 d\log\log\frac{T}{d}}}\cdot 2^{-\frac{c_4 d\log\log\log\frac{n}{D}+1}{c_4(d-1)\log\log\frac{T}{d-1}}} \\ &\ge 2^{-1-\frac{d\log\log\log\frac{n}{D}+1}{(d-1)\log\log\frac{T}{d-1}}} \\ &\ge 2^{-1-\frac{2d\log\log\log\frac{n}{D}}{d\log\log\log\frac{n}{D}}} = \frac{1}{8}\ .\end{aligned}$$

If x_j is chosen to be any of the integer values between x' and $x' + c_4 d\log\log\log\frac{n}{D} + 1$, then by Lemma 25, the probability that v is informed is at least $\frac{1}{8}\cdot 4^{-\frac{1}{8}} \ge \frac{1}{10}$. So, the overall probability that v is informed in time-step j is at least

$$\frac{1}{c}\cdot\frac{1}{10}\cdot\frac{c_4 d\log\log\log\frac{n}{D}}{T} \ge \frac{c_4 d\log\log\log\frac{n}{D}}{20cc_4^2 D\log\frac{n}{D}\log\log\log\frac{n}{D}} = \frac{d}{20cc_4 D\log\frac{n}{D}}\ .$$

We are now in a position to apply Lemma 62 with

$$\mu(d,\delta) = \begin{cases}\frac{20cc_4 D\log\frac{n}{D}}{d}, & \text{if } d \ge \frac{D}{\log^2\log\frac{n}{D}} \text{ and } \delta \le (\frac{n}{D})^2 \\ 36c\log\delta\log^2\log\delta & \text{otherwise.}\end{cases}$$

We can bound N, the maximum of $\sum_{d=1}^{D}\mu(d,\delta_d)$, subject to $\sum_{d=1}^{D}\delta_d \le n$, as follows: the maximum contribution of terms falling under the first case is at most

$$\sum_{d=\frac{D}{\log^2 \log \frac{n}{D}}}^{D} \frac{20cc_4 D \log \frac{n}{D}}{d} \leq 20cc_4 D \log \frac{n}{D}(1 + \int_{\frac{D}{\log^2 \log \frac{n}{D}}}^{D} \frac{di}{i})$$
$$\leq 20cc_4 D \log \frac{n}{D}(1 + \ln(\log^2 \log \frac{n}{D}))$$
$$\leq 48cc_4 D \log \frac{n}{D} \log \log \log \frac{n}{D} \ ,$$

the maximum contribution of terms where $d < \frac{D}{\log^2 \log \frac{n}{D}}$ is at most

$$\frac{D}{\log^2 \log \frac{n}{D}} \cdot 36c \log \frac{n \log^2 \log \frac{n}{D}}{D} \log^2 \log \frac{n \log^2 \log \frac{n}{D}}{D}$$
$$\leq \frac{37cD \log \frac{n}{D} \log^2 \log \frac{n}{D}}{\log^2 \log \frac{n}{D}}$$
$$= 37cD \log \frac{n}{D} \ ,$$

and the maximum contribution of terms where $\delta \geq (\frac{n}{D})^2$ is at most

$$\frac{D^2}{n} \cdot 36c \log(\frac{n}{D})^2 \log^2 \log(\frac{n}{D})^2 \leq 73cD \cdot \frac{D}{n} \log \frac{n}{D} \log^2 \log \frac{n}{D} \leq 73cD \ .$$

So, summing these contributions,

$$N \leq 48cc_4 D \log \frac{n}{D} \log \log \log \frac{n}{D} + 37cD \log \frac{n}{D} + 73cD$$
$$\leq 50cc_4 D \log \frac{n}{D} \log \log \frac{n}{D} \ .$$

We can also see that the maximum μ value μ_{max} is at most $20cc_4 \log n \log^2 \log n$. So, applying Lemma 62, we can conclude that the network will be informed within

$$4N + 91\mu_{max} \log n \leq 200Cc_4 D \log \frac{n}{D} \log \log \log \frac{n}{D} + 1820Cc_4 \log^2 n \log^2 \log n$$
$$\leq 200cc_4 D \log \frac{n}{D} \log \log \log \frac{n}{D} + 1820cc_4 \cdot 2D \log \frac{n}{D}$$
$$\leq 3840cc_4 D \log \frac{n}{D} \log \log \log \frac{n}{D}$$

time with high probability. Here in the second inequality, we are using that $D \geq \log n (\log \log n)^2$. We set $c_4 = 3840c$ to ensure that the iteration we analyze

is sufficiently long, and thus broadcast is performed in $O(D \log \frac{n}{D} \log\log\log \frac{n}{D})$ time.

□

Theorem 68. *Broadcasting can be performed in networks with collision detection in $O(D \log \frac{n}{D} \log\log\log \frac{n}{D} + \log^2 n)$ time, with high probability.*

Proof. We perform a beep-wave to ensure that all nodes v know their distance d_v from the source, and then apply the framework of Algorithm 16 using **Shallow-Broadcast**, **General-Broadcast**, **Semi-Shallow-Broadcast**, and **Deep-Broadcast**, i.e. $c = 4$. If $D \leq \log n$, then by Lemma 63 we complete broadcasting in $O(\log^2 n)$ time. If $\log n \leq D \leq \log n \log^2 \log n$, then by Lemma 66 we complete broadcasting in $O(D \log n \log\log\log n) = O(D \log \frac{n}{D} \log\log\log \frac{n}{D})$ time. If $D > \log n \log^2 \log n$, then by Lemma 67 we complete broadcasting in $O(D \log \frac{n}{D} \log\log\log \frac{n}{D})$ time. This gives a total asymptotic running time of $O(D \log \frac{n}{D} \log\log\log \frac{n}{D} + \log^2 n)$. □

5.5 Broadcast in Undirected Networks with Collision Detection

In undirected networks in which collision detection is available, the fastest algorithm with known network parameters is the $O(D + \log^6 n)$-time result of [28]. This algorithm involves utilizing beep-wave type methods to set up a structure known as a *gathering spanning tree*, which arose in work on known-topology radio networks [32] and admits a fast broadcasting schedule atop it. To achieve the $O(D+\log^6 n)$-time bound, constant-factor upper bounds on D and $\log n$ are required. However, since the running time does not contain a product of these two quantities, the algorithm can be simulated without parameter knowledge by using a doubling parameter T, where T is used as an upper bound for both D and $\log^6 n$, and the algorithm is run for T time in each iteration. Then, when T exceeds both D and $\log^6 n$, which happens within $O(D + \log^6 n)$ time, the algorithm will succeed.

5.6 Discussion and Open Problems

We present the first dedicated *randomized* broadcasting algorithms for networks without parameter knowledge, and show near-optimal running times. There are

still complexity gaps to close; in particular, algorithms reaching $O(D \log \frac{n}{D} + \log^2 n)$ would be interesting, as would a matching lower bound for directed networks with collision detection. An improvement we would like our algorithms to have is a deterministic selecting sequence (similarly to [23]), which would allow them to be used with multiple sources. Another possible extension to our work here is to achieve *acknowledged* broadcasting, i.e. to ensure that by some time-step, all nodes know that broadcasting has been successfully completed and can cease transmissions. This would solve the issue with our algorithms here that nodes must continue transmissions indefinitely. However, it is not clear if acknowledged broadcasting is possible in this model; as mentioned in Chapter 4, Chlebus et al. [10] show that, using deterministic algorithms and without collision detection, it is not.

Chapter 6

Randomized Leader Election

We now consider the problem of leader election, and design randomized algorithms for networks with and without collision detection.

6.1 Related Work

Work on leader election in radio networks started in the 1970s with the single-hop network model. In this setting, in the model with collision detection, leader election can be performed deterministically in $O(\log n)$ time, which was proven to be optimal by Greenberg and Winograd [30]. While randomization provides no benefit if a high probability bound is required, as a $\Omega(\log n)$ lower bound also exists [29, 51], Willard [55] gave an algorithm with the expected running time of $O(\log \log n)$ and showed that this bound is also asymptotically optimal. (We note, however, that this result assume that nodes do not know the value of n, nor a linear upper bound.) For the single-hop network model without collision detection, deterministic leader election has complexity $\Theta(n \log n)$ [15, 45], and randomized leader election has expected time complexity $O(\log n)$ [46] and high probability time complexity $O(\log^2 n)$ [39].

While the complexity of leader election in single-hop networks is now well understood, the complexity of the problem in multi-hop networks has been less developed.

In the seminal work initiating the study of the complexity of communication

protocols in multi-hop radio networks, Bar-Yehuda et al. [3] developed a general randomized framework of simulating single-hop networks with collision detection by multi-hop networks without collision detection. The framework yields leader election algorithms for multi-hop networks (in directed and undirected networks) running in $O(T_{BC} \cdot \log \log n)$ expected time and $O(T_{BC} \cdot \log n)$ time w.h.p., where T_{BC} is the time required to broadcast a message from a single source to the entire network. The same authors also gave a randomized broadcasting algorithm running in $O(D \log n + \log^2 n)$ time w.h.p., thereby yielding a leader election algorithm taking $O((D \log n + \log^2 n) \log \log n)$ expected time and $O(D \log^2 n + \log^3 n)$ time w.h.p.

The next improvement came with faster algorithms for broadcast due to Czumaj and Rytter [23], and independently Kowalski and Pelc [44], which require only $O(D \log \frac{n}{D} + \log^2 n)$ time w.h.p. Combining these algorithms with the simulation framework of Bar-Yehuda et al., one obtains leader election algorithms (even in the model without collision detection, and in both directed and undirected graphs) running in $O((D \log \frac{n}{D} + \log^2 n) \log \log n)$ expected time and $O((D \log \frac{n}{D} + \log^2 n) \log n)$ time w.h.p.

Recently, Ghaffari and Haeupler [27] took a new approach, which yielded faster leader election algorithms in *undirected networks*. The main idea of this work is to randomly select a small (logarithmic) number of candidates for the leader and then repeatedly run "debates" to reduce the number of candidates to one. Standard random sampling technique allows one to choose in constant time a random set of $\Theta(\log n)$ candidates, with high probability. Then, by running a constant number of broadcasting computations and neighborhood exploration algorithms (this phase is called a "debate" in [27] and it relies heavily on the assumption that the network is undirected), one can reduce the number of candidates by a constant factor. Using this approach, Ghaffari and Haeupler gave a leader election algorithm (in undirected networks) that in $O((D \log \frac{n}{D} + \log^3 n) \cdot \min\{\log \log n, \log \frac{n}{D}\})$ rounds elects a single leader w.h.p. For the model with collision detection, the same work [27] used a similar approach to elect a leader in $O((D + \log n \log \log n) \cdot \min\{\log \log n, \log \frac{n}{D}\})$ rounds w.h.p. These algorithms are nearly optimal, and are the fastest currently known for undirected networks and for worst-case running time.

6.2 Our Results

We present a framework for leader election in multi-hop radio networks which yield randomized leader elections taking O(broadcasting time) in expectation, and another which yields algorithms taking fixed $O(\sqrt{\log n})$-times broadcasting time. Both succeed with high probability.

We show how to implement these frameworks in radio networks without collision detection, and in networks with collision detection (in fact in the strictly weaker beep model). In doing so, we obtain the first optimal expected-time leader election algorithms in both settings, and also improve the worst-case running time in directed networks without collision detection by an $O(\sqrt{\log n})$ factor.

6.3 Leader Election Frameworks

We first give frameworks for leader election in radio network-like scenarios which are independent of the specific communication rules of the models involved.

6.3.1 Verify and Eliminate

Our leader election frameworks will be based upon the use of two subprocedures, which we will call Verify and Eliminate. In broad terms, these procedures are both means of utilizing a global broadcast to collect some information about the current state of a leader election attempt.

Verify The purpose of Verify is to determine whether the input set (which in our application will be a set of candidate leaders) is of size 0, 1, or greater than 1. Further, in the case that the set is of size exactly 1 (which is what we hope for), all nodes receive the ID of its sole member.

Formally, Verify(C) takes as input a subset of nodes C and outputs a pair $(m(v), b) \in [L] \times \{0, 1, 2\}$ to each node v, satisfying the following conditions:

- $b = \begin{cases} 0 & \text{if } |C| = 0; \\ 1 & \text{if } |C| = 1; \\ 2 & \text{otherwise;} \end{cases}$
- if C consists of only a single element c, then $m(v) = ID(c)$ for all nodes v.

Here b carries the information about the size of C, and if $|C| = 1$ as required, $m(v)$ gives the candidate ID. Otherwise, we make no restrictions on $m(v)$.

Eliminate ELIMINATE has stronger requirements than VERIFY. Rather than just returning information about the size of the input set, it instead returns an output set which is of size at least 1, and strictly smaller than the input set (as long as this was of size greater than 1). In our application, this will allow us to thin out a set of candidate leaders by (at least) one per iteration, until we reach a single leader.

Formally, ELIMINATE(C) takes as input a subset of nodes C and outputs a pair $(m(v), C') \in [L] \times 2^V$ to each node v, satisfying the following conditions:

- $1 \leq |C'| < \max\{|C|, 2\}$;
- if C consists of only a single element c, then $m(v) = ID(c)$ for all nodes v.

In both of these procedures, sets given as input or output are implicit; that is, that each node receives, or outputs, only the information of whether it is itself a member of the set (rather than full knowledge of the set). Furthermore, we will be considering randomized implementations of these procedures, and so will ensure that our implementations meet the specified requirements with high probability.

We will show how to implement these sub-procedures in our network models later; first we describe how to use them to build general leader election frameworks.

6.3.2 Leader Election Frameworks

In this section we will show how the VERIFY and ELIMINATE procedures can be combined into frameworks for leader election. While in this work we are focused on radio networks, in general these frameworks could extend to other distributed computing models.

Variable time

We first give the framework for a leader election algorithm whose running time is a random variable, which is O(broadcasting time) in expectation.

The idea is simple: we repeatedly randomly choose a set of candidate leaders, and terminate when the set we chose is of size 1.

Algorithm 17 Leader Election, variable time

loop
each $v \in V$ chooses to be in the set C of **candidates** with probability $\Theta(\frac{1}{n})$
$(m(V), b) \leftarrow$ VERIFY(C)
if $b = 1$ **then** output $m(V)$, terminate
end loop

Theorem 69. *If* VERIFY(C) *is implemented in t time to succeed with high probability, then Algorithm 17 performs leader election in $O(t)$ expected time, succeeding with high probability.*

Proof. Assume for the sake of the analysis that if the algorithm has run unsuccessfully for $\frac{n}{2}$ iterations, it terminates. With probability at least $1 - \frac{n^{-1}}{2}$, VERIFY returns the correct result for these $\frac{n}{2}$ iterations. Notice that in any particular iteration, the probability that $|C| = 1$ is bounded above by a constant, and denote this constant c. With probability at least $1 - c^{\frac{n}{2}}$ one of the iterations will have had $|C| = 1$, so the algorithm will correctly perform leader election with probability at least $(1 - \frac{n^{-1}}{2})(1 - c^{\frac{n}{2}}) \geq 1 - n^{-1}$. Since c is constant, expected number of iterations until $|C| = 1$ is also a constant k. Expected number of iterations until termination is therefore at most $k(1 - \frac{n^{-1}}{2}) + \frac{n}{2}\frac{n^{-1}}{2} < k + 1$, i.e., a constant, and since the running time of each iteration is dominated by that of VERIFY, expected running time is $O(t)$. □

Fixed time

Next we give another framework for leader election whose running time is fixed, and therefore has better worst-case performance. This framework is slightly more complex, and consists of two main phases. In the first, each node chooses to be a candidate with probability $\Theta(\frac{\log n}{n})$, which ensures that $\Theta(\log n)$ candidates are chosen with high probability. Then we repeatedly have candidates drop out with probability $\frac{1}{2}$, and use the VERIFY procedure to check that we did not remove all candidates. Doing this $\Theta(x)$ times (where x is some parameter between 1 and $\log n$ to be fixed presently) ensures that only $\Theta(\frac{\log n}{x})$ candidates remain, w.h.p. Then, in the second phase, we repeatedly use ELIMINATE to remove candidates one at a time, until only a single one remains.

The running time of the algorithm will be dominated by the $\Theta(x)$ calls to VERIFY and the $\Theta(\frac{\log n}{x})$ calls to ELIMINATE. Therefore, if we let t and u be the

Algorithm 18 Leader Election, fixed time

each $v \in V$ chooses to be in the set C of **candidates** with probability $\Theta(\frac{\log n}{n})$
loop $\Theta(x)$ times
  each $v \in C$ chooses to be in C' with probability $\frac{1}{2}$
  $(m(v), b) \leftarrow$ VERIFY(C')
  if $b \neq 0$ then $C \leftarrow C'$
end loop
loop $\Theta(\frac{\log n}{x})$ times
  $(m(v), C) \leftarrow$ ELIMINATE(C)
end loop
output $m(v)$, terminate

running times of VERIFY and ELIMINATE respectively, we see that to optimize our overall running time we should set $x = \sqrt{\frac{u}{t}\log n}$.

Theorem 70. *Algorithm 18 performs leader election in $O(\sqrt{tu\log n})$ time and succeeds with high probability.*

Proof. With high probability, $\Theta(\log n)$ nodes choose to be candidates in C. First we will analyze how many candidates remain after the first loop. Let y be the number of rounds of the loop during which $|C| > \frac{2\log n}{x}$, and consider only these rounds. We call such a round i *successful* if $\frac{5|C|}{6} > |C'| \geq 1$. The probability that any round is not successful is at most:

$$\begin{aligned}
\mathbf{Pr}\left[|C'| = 0\right] + \mathbf{Pr}\left[|C'| \geq \tfrac{5|C|}{6}\right] &\leq \left(\frac{1}{2}\right)^{|C|} + \binom{|C|}{\frac{5|C|}{6}}\left(\frac{1}{2}\right)^{\frac{5|C|}{6}} \\
&= \left(\frac{1}{2}\right)^{|C|} + \binom{|C|}{\frac{|C|}{6}}\left(\frac{1}{32}\right)^{\frac{|C|}{6}} \leq \left(\frac{1}{2}\right)^{|C|} + \left(\frac{e\cdot|C|}{\frac{|C|}{6}}\right)^{\frac{|C|}{6}}\cdot\left(\frac{1}{32}\right)^{\frac{|C|}{6}} \\
&= \left(\frac{1}{2}\right)^{|C|} + \left(\frac{3e}{16}\right)^{\frac{|C|}{6}} \leq 2\times 0.9^{|C|} \leq 0.9^{\frac{\log n}{x}} \quad .
\end{aligned}$$

Note that this is still true conditioned on the randomness of all previous rounds, and so the total number of unsuccessful rounds is majorized by a bino-

mially distributed variable $\mathbb{B}\text{in}(y,p)$, where $p = 0.9^{\frac{\log n}{x}}$. If $y \geq 20x$ then

$$\begin{aligned}\mathbf{Pr}\left[\text{at least } \tfrac{y}{2} \text{ rounds are unsuccessful}\right] &\leq \mathbf{Pr}\left[\mathbb{B}\text{in}(y,p) \geq \tfrac{y}{2}\right] \\ &\leq e^{-\frac{1}{2}y(\ln \frac{1}{2p} + \ln \frac{1}{2(1-p)})} \\ &\leq e^{-\frac{y}{2}\ln \frac{0.9^{-\frac{\log n}{x}}}{2}} \leq e^{-\frac{y}{2}\ln e^{\frac{0.1\log n}{x}}} \\ &= e^{-\frac{y\log n}{20x}} \leq n^{-\log e} \ .\end{aligned}$$

So either $y < 20x$ (i.e., after $20x$ rounds $|C| < \frac{2\log n}{x}$), or with high probability at least $10x$ of the first $20x$ rounds are successful and so $|C|$ has reduced by a factor of at least $\left(\frac{5}{6}\right)^{10x}$, and hence is below $\frac{\log n}{x}$.

Then, since ELIMINATE reduces $|C|$ by at least 1 per round, after $\Theta(\frac{\log n}{x})$ rounds only one candidate remains, and leader election is complete.

The running time is dominated by $\Theta(x)$ rounds of VERIFY and $\Theta(\frac{\log n}{x})$ rounds of ELIMINATE. With $x = \sqrt{\frac{u\log n}{t}}$, this gives a total running time of $O(\sqrt{tu\log n})$. □

6.4 Implementation

We now show how to implement VERIFY(C) and ELIMINATE(C) in the models we consider: radio networks without collision detection, and with collision detection. In the latter case, we will actually use the strictly weaker beep model, as any algorithm that works in the beep model also works in radio networks with collision detection.

6.4.1 Radio Networks Without Collision Detection

Our algorithms will make use of some existing techniques for the radio network model, namely methods for broadcasting messages within local neighborhoods and throughout the entire network.

Local and global broadcast

For local broadcasting, we will utilize the classical DECAY protocol of [4] (see Algorithm 12 from Chapter 5). Recall that each *round* of DECAY (taking $\log n$ time-steps) results in a successful transmission with probability at least $\frac{1}{4}$.

While Decay has a very localized effect, to achieve global tasks we will need more complex primitives. In particular, we will also need a method of propa-

gating information globally. For this we employ another previous result from the literature, which yields a PARTIAL MULTI-BROADCAST(S, f) algorithm with the following properties:

- $S \subseteq V$ is a set of *source nodes*;
- $f : S \to \{0,1\}^\ell$, where $\ell = O(\log n)$ is a message length parameter, is a function giving each source a message to broadcast; in our applications, this will either be a node ID, or a single bit "1";
- Each node v interprets some bit-string $m(v)$ upon completion;
  - if $S = \emptyset$, then $m(v) = \epsilon$ (the empty string) for all v;
  - if $S \neq \emptyset$, then $\forall v \in V\ \exists s \in S$ with $m(v) = f(s)$, i.e., each node interprets some source's message.

To achieve this, we use the broadcasting algorithm of Czumaj and Rytter [23], performed with every node in S operating as a single source and with nodes interpreting the first transmission they receive to be their output message $m(v)$. We require the *deterministic selecting sequence* version of their algorithm, rather than the randomized selecting sequence version discussed in Chapter 5, on which Algorithm 14 is based. This is because the latter requires global shared randomness, which is not available when we have multiple sources.

Lemma 71 (From [23])**.** *There is a Partial Multi-Broadcast algorithm running in time $O(D \log \frac{n}{D} + \log^2 n)$, which succeeds with high probability.* □

Armed with procedures for both local and global message dissemination, we can implement VERIFY and ELIMINATE:

Subprocedure implementation

In radio networks without collision detection, our implementations of VERIFY and ELIMINATE both run in time $T_{BC} = O(D \log \frac{n}{D} + \log^2 n)$ time, and we can in fact combine them into the same process (Algorithm 19; here the output triple contains the necessary outputs for both VERIFY and ELIMINATE). The idea of both is to make use of PARTIAL MULTI-BROADCAST to inform all nodes of at least one candidate ID, check if any neighboring nodes received different IDs, and performing PARTIAL MULTI-BROADCAST again to inform the network if this was the case. To meet the stricter requirements of ELIMINATE, we have

nodes who detected differing IDs broadcast the highest ID they heard in this final phase. Then, if there are multiple candidates, at least one of the candidates (the one with the lowest ID) becomes aware of another with a higher ID and drops out.

We note that this implementation is similar to the method used by Bar-Yehuda, Goldreich and Itai [3] to simulate algorithms for single-hop networks within multi-hop networks.

Algorithm 19 VERIFY&ELIMINATE(C), no-CD

```
m(v) ← PARTIAL MULTI-BROADCAST(C, ID(C))
if m(v) ≠ ε then
    for i = 1 to ID length do
        let v ∈ S_i if the i^th bit of v's message m(v)_i is 1
        perform four rounds of DECAY(S_i)
        if v receives a node ID but m(v)_i = 0 then
            v becomes a member of the set W of witnesses
            m(v) ← highest ID v knows
        end if
    end for
    p(v) ← PARTIAL MULTI-BROADCAST(W, m(W))
    if p(v) > ID(v) then v drops out of C
    if p(v) = ε then v outputs (m(v), C, 1), procedure terminates
    v outputs (m(v), C, 2), procedure terminates
else
    v outputs (m(v), C, 0), procedure terminates
end if
```

Theorem 72. *Algorithm 19 performs both* VERIFY *and* ELIMINATE *within time* $O(T_{BC}) = O(D \log \frac{n}{D} + \log^2 n)$*, and succeeds with high probability.*

Proof. If C is empty, the initial call of PARTIAL MULTI-BROADCAST will involve no transmissions, so all nodes will output $(\epsilon, \emptyset, 0)$, satisfying the requirements.

If $|C| = 1$ then all nodes will receive the ID of its sole member. Therefore there will be no iteration of the for loop in which one node performs DECAY and another does not, and so no nodes will become witnesses. Then, the second invocation of PARTIAL MULTI-BROADCAST will involve no transmissions, so nodes output $(m(v), C, 1)$ with $m(v)$ being the ID of C's member, as required.

If $|C| > 1$ then there will be at least one pair of neighboring nodes who received different IDs during the first step. Since IDs are $\Theta(\log n)$-bit strings chosen uniformly at random, with high probability every pair of IDs differs on $\Omega(\log n)$ positions (this can easily be seen by taking a union bound over all pairs).

For each such position, after four rounds of DECAY the node whose received ID has a **1** in the position will inform the neighboring node with constant probability. Since this event is independent for each of the $\Omega(\log n)$ positions, with high probability the DECAY phase succeeds in at least one of them. This results in a node becoming a witness. So, in the second call of PARTIAL MULTI-BROADCAST, all nodes receive some message and output $(m(v), C, 2)$. Clearly this satisfies the conditions of VERIFY. For ELIMINATE we require that at least one node dropped out of C. To see this, consider the node v in C which had the lowest ID before the procedure. In the second PARTIAL MULTI-BROADCAST call it receives an ID from some witness. The witness compared at least two IDs and picked the highest to broadcast. Therefore the ID it picked must have been higher than v's, so v will drop out of C.

The running time is dominated by two calls of PARTIAL MULTI-BROADCAST taking $O(D \log \frac{n}{D} + \log^2 n)$ time, and by $O(\log n)$ rounds of DECAY, taking $O(\log^2 n)$ time. Therefore total running is $O(D \log \frac{n}{D} + \log^2 n)$. □

6.4.2 Radio Networks With Collision Detection

When collision detection is available, we can speed up the VERIFY procedure by utilizing *beep waves* for broadcasting. As mentioned, our implementations for this setting actually work for the strictly weaker beep model.

Beep waves (see Chapter 3) are a means of propagating information through the network one bit at a time via waves of collisions. For our specific purposes, we require procedure MULTI-BEEP-WAVE(S, f) which satisfies the following:

- $S \subseteq V$ is a (possibly empty) set of source nodes;
- $f : S \to \{0,1\}^\ell$, where $\ell = O(\log n)$ is some message length parameter, is a function giving each source a bit-string to broadcast.
- Each node interprets a string $m(v)$;
  - If $S = \emptyset$, then $m(v) = \epsilon$ (the empty string) for all v;
  - If $S = \{s\}$, for some $s \in V$, then $\forall v \in V$, $m(v) = f(s)$;
  - If $|S| > 1$, then for all v, $m(v) \neq \epsilon$. Furthermore, there exists $w \in V$ and two distinct $u, v \in S$ (we allow $w \in \{u, v\}$) such that $m(w) = f(u) \vee f(v) \vee m$, for some bit-string m.

The last condition may seem convoluted; the reason for it is that, while we cannot guarantee messages are correctly received as in the single source case, we will at least need some means of telling that there were indeed multiple sources. This will be detailed later, but for now we require that, as well as all nodes receiving some non-empty message, at least one receives the logical **OR** of two source messages, possibly with some extra **1**s. We achieve these conditions with Algorithm 20.

Note that this method only works in *undirected networks*, since in directed networks it is impossible to prevent beep waves from interfering with subsequent ones, and consequently broadcasting in directed networks in the beep model is much slower (see Chapter 3).

Algorithm 20 MULTI-BEEP-WAVE(S, f) at a node v

```
initialize m(v)_i = 0 ∀i ∈ [ℓ]
if v ∈ S then
    v beeps in time-step 0
    for i = 1 to ℓ do
        if the i^th bit of f(v) is 1 then v beeps in time-step 3i and m(v)_i ← 1
        else v receives a beep in time-step 3i and m(v)_i ← 1
    end for
else
    Let j be the time-step in which v receives its first beep
    v beeps in time-step j + 1
    for i = 1 to ℓ do
        if v receives a beep in time-step j + 3i then
            v beeps in time-step j + 3i + 1 and m(v)_i ← 1
        else
            m(v)_i ← 0
        end if
    end for
end if
```

We prove that the algorithm has the desired behavior:

Lemma 73. *If* MULTI-BEEP-WAVE(S, f) *is run with* $S = \{s\}$*, then* $\forall v \in V$*,* $m(v) = f(s)$*.*

Proof. Partition all nodes into layers depending on their distance from the source s, i.e., layer $L_i = \{v \in V : dist(v, s) = i\}$. We first note that a node in layer i beeps for the first time in time-step i, since this first beep will propagate through the network one layer per time-step. The algorithm then ensures that such a node will beep only in time-steps equivalent to $i \bmod 3$, and only if a

beep was heard in the previous step. Since all neighbors of the node must be in layers $i-1, i$, and $i+1$, only messages from neighbors in layer $i-1$ can be relayed (as these are the only neighbors whose beeps are in time-steps equivalent to $i-1 \bmod 3$). It is then easy to see that layers act in unison, and beep if and only if the previous layer beeped in the previous time-step. □

Lemma 74. *If* Multi-Beep-Wave(S, f) *is run with* $|S| > 1$ *then there exists* $w \in V$ *and two distinct* $u, v \in S$ *(we allow* $w \in \{u, v\}$*) such that* $m(w) = f(u) \vee f(v) \vee m$*, for some bit-string* m*.*

Proof. Let u, v be the closest pair of sources in the graph. Let w be the midpoint on the shortest $u \to v$ path (if the path is of odd length, pick either midpoint arbitrarily).

If w is a source, then we can assume, without loss of generality, that $w = u$ and v is an adjacent source. Then if $f(w)_i = 1$ or $f(v)_i = 1$, w receives or transmits a beep in time-step $3i$ and sets $m(w)_i = 1$, so we are done.

Otherwise, assume without loss of generality that $dist(w, u) \leq dist(w, v) \leq dist(w, u) + 1$, and denote $j := dist(w, u) - 1$. w receives its first beep in time-step j. Then, since u is the closest source to every node along the shortest $u \to w$ path and v is the closest source to every node along the shortest $v \to w$ path, beeps from u and v will always be relayed along these paths. So, if $f(u)_i = 1$, w receives a beep in time-step $j + 3i$, and if $f(v)_i = 1$, w receives a beep in time-step $dist(v, w) - 1 + 3i = j + 3i$ or $j + 3i + 1$ (unless w beeps itself in time-step $j + 3i + 1$, i.e., it received a beep in time-step $j + 3i$). In either case, $m(w)_i$ is set to 1. □

Lemma 75. *Algorithm 20 correctly achieves the Beep-Wave conditions in* $O(D + \ell)$ *time-steps.*

Proof. Correctness for the cases $|S| = 1$ and $|S| > 1$ follow from Lemmas 73–74, and the case $S = \emptyset$ follows since no node ever beeps. To analyze the running time: clearly all sources will have ceased transmission after $O(\ell)$ time, and since beeps are propagated through the network one layer per time-step, it may be a further D time-steps before a source's last beep is heard by the whole network, yielding $O(D + \ell)$ time. □

Subprocedure implementation

We adapt Verify to make use of Multi-Beep-Wave, which in these circumstances is faster than Partial Multi-Broadcast.

Algorithm 21 VERIFY(C) in radio networks with collision detection

$m(v) \leftarrow$ BEEP-WAVE$(C, \text{ID}^*(C))$
if $m(v) \neq \epsilon$ **then**
  if $m(v)$ has more than $10 \log n$ **1**s **then** v becomes a member of witness set W
  $p(v) \leftarrow$ BEEP-WAVE$(W, 1)$
  if $p(v) = \epsilon$ **then** v outputs $(m(v), 1)$, procedure terminates
  else v outputs $(m(v), 2)$, procedure terminates
else
  v outputs $(m(v), 0)$, procedure terminates
end if

The idea of this algorithm is similar to that of Algorithm 19, except that now to identify ID clashes we use a property of MULTI-BEEP-WAVE rather than rounds of DECAY.

To do so we must make a change to candidate IDs: when we have a clash of IDs, what we will get is the logical **OR** superimposition of the bit-strings, as per the specification of MULTI-BEEP-WAVE. Hence, if all IDs have the same number of **1**s, we will be able to identify such a clash since there will be more **1**s than a single ID. So, in our VERIFY algorithm we use ID* to mean original ID with $10 \log n$ extra bits appended, with the purpose of padding the number of **1**s to *exactly* $10 \log n$.

Theorem 76. *Algorithm 21 performs* VERIFY *within time* $O(D + \log n)$.

Proof. If C is empty, the initial call of MULTI-BEEP-WAVE will involve no transmissions, so all nodes will output $(\epsilon, 0)$, satisfying the requirements.

If $|C| = 1$ then all nodes will receive the ID of its sole member, which has exactly $10 \log n$ **1**s. Therefore there will be no witnesses, the second invocation of MULTI-BEEP-WAVE will involve no transmissions, so nodes output $(m(v), 1)$ with $m(v)$ being the ID of C's member, as required.

If $|C| > 1$ then by the properties of MULTI-BEEP-WAVE at least one node will receive the logical **OR** superimposition of two IDs (and possibly some other string). Since the IDs are unique w.h.p. (and this remains true with our modification to ID*), this superimposition must have more than $10 \log n$ **1**s. So, at least one node will become a witness and will broadcast during the second invocation of BEEP-WAVE, and therefore all nodes will output $(m(v), 2)$ as required.

The running time is dominated by two calls of MULTI-BEEP-WAVE taking $O(D + \log n)$ time. □

We cannot use similar methods to speed up ELIMINATE, since the single

bit of information that witnesses broadcast is not sufficient to guarantee that at least one candidate will drop out, and we cannot quickly broadcast longer messages from multiple sources without interference. Furthermore, even with a faster ELIMINATE sub-procedure, our fixed running time framework would still yield an algorithm slower than the $O((D + \log n \log\log n) \min\{\log\log n, \log \frac{n}{D}\})$ time algorithm of [27]. Therefore we only use our variable-time framework for networks with collision detection.

6.5 Running Times

We can now plug the running times of our implementations of the VERIFY and ELIMINATE sub-procedures (given by Theorems 72 and 76) into our framework results (Theorems 69 and 70) to obtain the following leader election results:

Theorem 77. *Leader election can be performed in radio networks (either directed or undirected) without collision detection within $O(D \log \frac{n}{D} + \log^2 n)$ time* in expectation*, succeeding with high probability.*

Proof. Follows immediately from Theorems 69 and 72. □

This running time is asymptotically optimal, and improves by a $\Theta(\log\log n)$ factor over the previous best result of [27].

Theorem 78. *Leader election can be performed in radio networks (either directed or undirected) without collision detection in $O((D \log \frac{n}{D} + \log^2 n)\sqrt{\log n})$ time, succeeding with high probability.*

Proof. Follows immediately from Theorems 70 and 72. □

While this algorithm is slower than that of [27], it has the benefit of working in directed networks as well as undirected. In directed networks, it is a $\Theta(\sqrt{\log n})$ factor faster than previous results.

Though these results are given by two different frameworks, we can easily combine them into a single algorithm, either by interspersing steps of the two algorithms alternately, or by running Algorithm 17 for $O((D \log \frac{n}{D} + \log^2 n)\sqrt{\log n})$ time and then running Algorithm 18. Consequently, we can achieve good expected and worst case running times concurrently.

For undirected networks in the model with collision detection and in the beep model, we can obtain the following stronger bound:

Theorem 79. *Leader election can be performed in undirected radio networks with collision detection within $O(D+\log n)$ time in expectation, succeeding with high probability.*

Proof. Follows immediately from Theorems 69 and 76. □

6.6 Discussion and Open Problems

We present a framework for leader election in radio networks which yield randomized leader elections taking optimal O(broadcasting time) in expectation, and another which yields algorithms taking fixed $O(\sqrt{\log n})$-times broadcasting time. Both these algorithms succeed with high probability. We then showed implementations of the framework in radio networks with and without collision detection.

Some of these results have since been partially superseded: the $O(D+\log n)$-time deterministic leader election algorithm of [24], for the undirected beep model, improves over Theorem 79. Furthermore, we show a randomized leader election algorithm for undirected networks without collision detection taking $O(D\frac{\log n}{\log D} + \log^{O(1)} n)$ worst-case time in Chapter 7, which improves over Theorems 77 and 78 in most circumstances. However, these theorems remain the strongest results for leader election in directed networks.

Indeed, directed networks are the setting in which it seems further improvement should be most possible. Currently, there are no leader election algorithms known for directed networks which exploit collision detection at all, so an open question is whether collision detection adds any power here. Furthermore, while the expected running time of Theorem 77 is optimal, worst case running time is still a $\sqrt{\log n}$-factor away from the lower bound, and one would wish to close this gap.

Chapter 7

Spontaneous Transmissions

In Chapter 5 we saw an $O(D \log \frac{n}{D} + \log^2 n)$-time broadcasting algorithm based on those developed by Czumaj and Rytter [23] and Kowalski and Pelc [44]. We also mentioned that this running time is known to be optimal [2, 46]. However, Kushilevitz and Mansour's $\Omega(D \log \frac{n}{D})$ time lower bound crucially relies on the assumption that *spontaneous transmissions* are not allowed, i.e. that nodes can only participate once they have received the source message. In this chapter we describe how spontaneous transmissions can be exploited to improve running times and break this lower bound for broadcasting and leader election.

7.1 Related Work

We will be considering undirected multi-hop networks without collision detection, with a global clock and parameter knowledge. In this setting, the fastest randomized broadcasting algorithm until very recently was the $O(D \log \frac{n}{D} + \log^2 n)$-time algorithm of [23, 44], which is optimal in the absence of spontaneous transmissions. The fastest leader election algorithms were that of Ghaffari and Haeupler [27], attaining $O(D \log \frac{n}{D} + \log^3 n) \cdot \min\{\log \log n, \log \frac{n}{D}\}$ worst-case running time with high probability of success, and Algorithm 17 from Chapter 6, which runs in $O(D \log \frac{n}{D} + \log^2 n)$ expected time and also succeeds with high probability. For a more comprehensive history of broadcasting and leader election algorithms, see Chapters 5 and 6 respectively.

In 2016, Haeupler and Wajc [34] demonstrated that allowing spontaneous transmissions can lead to faster broadcasting algorithms, by designing a ran-

domized algorithm that completes broadcasting in $O(D\frac{\log n \log\log n}{\log D} + \log^{O(1)} n)$ time, succeeding with high probability. This is the only algorithm that beats the lower bound of $\Omega(D \log \frac{n}{D} + \log^2 n)$ [2, 46] in the model with no spontaneous transmissions, and indeed is the only non-trivial use of spontaneous transmissions in radio networks, to our knowledge (though they have previously been used for naive Round-Robin type procedures as part of deterministic algorithms, e.g. [10]). Given that for the model that allows spontaneous transmissions any broadcasting algorithm requires $\Omega(D + \log^2 n)$ time (cf. [2, 53]), the algorithm due to Haeupler and Wajc [34] is *almost optimal* (up to an $O(\log\log n)$ factor) whenever n is polynomial in D.

7.1.1 Our Results

We extend the approach recently developed by Haeupler and Wajc [34] to design a fast randomized algorithm for both broadcasting and leader election, running in time $O(D\frac{\log n}{\log D} + \log^{O(1)} n)$, and succeeding with high probability (Theorem 83). When $D = \Omega(\log^c n)$ for a sufficiently large constant c, these running time bounds improve the fastest previous algorithms for broadcasting and leader election by factors $O(\log\log n)$ and $O(\log n \log\log n)$, respectively. More importantly, whenever n is polynomial in D (i.e., $n = O(D^c)$, for some constant $c \geq 1$), this running time is $O(D)$, which is asymptotically optimal since time D is required for any information to traverse the network.

Our algorithm is the first to achieve optimality over this range of parameters, and is also the first instance (in this model) of leader election time being asymptotically equal to fastest broadcasting time, since the former is usually a harder task in radio network models.

Note: We assume throughout that $D = \Omega(\log^c n)$ for some sufficiently large constant c. If this is not the case, then the $O(D \log \frac{n}{D} + \log^2 n)$-time algorithm of [23, 44] should be used instead.

7.2 Overview of Approach

We describe how to design a single procedure which performs both broadcasting and leader election:

7.2.1 Broadcasting, Leader Election, and Compete

To perform both broadcasting and leader election using the same algorithm, we study an auxiliary problem, which we call COMPETE, which generalizes both tasks. Recall that in broadcasting, one particular node, called the *source*, has a message which must become known to all other nodes. Meanwhile, in leader election, all nodes must agree on a designated leader node.

COMPETE has a similar flavor to broadcasting, but instead of transmitting a single message from a single source to all nodes in the network, it takes as its input a source set $S \subseteq V$, in which every source $s \in S$ has a message (of integer value) it wishes to propagate, and guarantees that upon completion all nodes in $\mathfrak{N}$ know the *highest-valued* source message.

It is easy to see how the COMPETE process generalizes broadcasting: it is simply invoked with the source as the only member of the set S. To perform leader election, one can probabilistically generate a small set (e.g., of size $\Theta(\log n)$) of candidate leaders, and then perform COMPETE using this set, with IDs as the messages to be propagated. Therefore, to design efficient randomized broadcasting and leader election algorithms, it is sufficient to design a fast randomized algorithm for COMPETE.

Our approach to study COMPETE (and hence also broadcasting and leader election problems) follows the methodology recently applied for fast distributed communication primitives by Ghaffari, Haeupler, Wajc, and others (see, e.g., [27, 34]). In order to solve the problem, we split computations into three parts. First, all nodes in the network will communicate with their local neighborhood to create some clustering of the network. Then, using this clustering, the nodes will perform some computations within each cluster, so that all nodes in the cluster share some useful knowledge. Finally, the knowledge from the clusters will be utilized to efficiently perform global communication.

7.2.2 Clusterings, Partition, and Schedulings

To implement this approach efficiently, we follow a similar line to that of Haeupler and Wajc [34] and rely on a clustering procedure of Miller et al. [50], adapted for the radio network model. We consider a partitioning of the input network into clusters in distributed setting, such that

- each node identifies one particular node as its *cluster center*,

- any node which is a cluster center to any other must be cluster center to itself, and
- the subgraph of nodes identifying any particular node as their cluster center is connected.

Haeupler and Wajc provide a means of achieving such a clustering with useful properties. In what follows, the term *"strong diameter"* refers to diameter using only edges within the relevant cluster (i.e. disallowing 'short-cuts' which involve leaving the cluster and returning).

Lemma 80 (Lemma 3.1 of [34])**.** *Let $0 < \beta \leq 1$. Any network on n nodes can be partitioned into clusters such that:*

- *each cluster has strong diameter $O(\frac{\log n}{\beta})$ with high probability, and*
- *every edge is cut by this partition (has its endpoints in distinct clusters) with probability $O(\beta)$.*

This algorithm can be implemented in the radio network setting in $O(\frac{\log^3 n}{\beta})$ rounds.

We will not prove this lemma, but we give a brief explanation of the method:

Clustering Method

The clustering algorithm can be described very simply (though the radio network implementation is less simple): each node v independently generates δ_v, an exponentially distributed random variable with parameter β (i.e. a variable taking values in $\mathbb{R}_{\geq 0}$ with $\mathbf{Pr}[\delta_v \leq y] = 1 - e^{-\beta y}$). Then, v joins the cluster of the node u which minimizes $dist(u, v) - \delta_u$.

The purpose of the variables δ_v is to break symmetry, and cause some nodes to be more prone to becoming cluster centers than others. The exponential distribution is chosen because of its *memorylessness* property: if we condition on δ_v exceeding some threshold t, then the amount that δ_v exceeds t is distributed identically to δ_v's original distribution (formally, $\mathbf{Pr}[\delta_v \leq t + y | \delta_v \geq t] = \mathbf{Pr}[\delta_v \leq y]$).

This property can give us an intuition for why every edge is cut with probability $O(\beta)$. Fix an edge e and let v be an endpoint. For e to be cut, there must two distinct nodes u, w such that $0 \leq (dist(u, v) - \delta_u) - (dist(w, v) - \delta_w) \leq 1$ (this is necessary for w to be v's cluster center, and u to be the cluster center

of e's other endpoint). But by the memorylessness property, this is equal to $\mathbf{Pr}[\delta_w] \leq 1$, which is $1 - e^{-\beta} = O(\beta)$. Note that this is not a formal argument; for detailed proof see [34].

We can also see why each cluster has strong diameter $O(\frac{\log n}{\beta})$: with high probability, all δ_v will be $O(\frac{\log n}{\beta})$. Then, for any node u further from v than that, $dist(u,v) - \delta_u \geq 0 \geq dist(v,v) - \delta_v$, so u can never be v's cluster center. This means that weak cluster diameter is $O(\frac{\log n}{\beta})$; to see that the same is true of strong diameter, notice that if a node w is v's cluster center, then it must also be cluster center to all nodes on the (w,v)-shortest path. Therefore, the strong diameter of a cluster is at most twice the weak diameter.

Using the Clusterings

The clustering provided by the application of Lemma 80 will be denoted by Partition(β).

This framework will be used in our central result, Theorem 81, which states that upon applying Partition(β) with β randomly chosen from some range polynomial in D, with constant probability the expected distance from some fixed node to its cluster center is $O(\frac{\log n}{\beta \log D})$.

Theorem 81. *Let j be an integer chosen uniformly at random between $0.01 \log D$ and $0.1 \log D$, and let $\beta = 2^{-j}$. For any node v, with probability at least 0.55 (over choice of j), the expected distance from v to its cluster center upon applying* Partition(β) *is $O(\frac{\log n}{\beta \log D})$.*

We prove this result in Section 7.6.

Theorem 81 applies to the clustering method in any setting, not just radio networks, and hence it may well be of independent interest. It improves over the result of [34] that expected distance to cluster center is $O(\frac{\log n \log\log n}{\beta \log D})$.

The approach described above is combined with a means of communicating within clusters from [28] using the notion of *schedules*. These schedules are adapted from results on radio networks with known topology [32].

Lemma 82 (Lemma 2.1 of [34])**.** *A network of diameter D and with n nodes can be preprocessed in $O(D \log^{O(1)} n)$ rounds, yielding a* ***schedule*** *which allows for one-to-all broadcast of k messages in $O(D + k \log n + \log^6 n)$ rounds with high probability. This schedule satisfies the following properties:*

- *for some prescribed node r, the schedule transmits messages to and from nodes at distance ℓ from r in $O(\ell + \log^6 n)$ rounds with high probability;*

- *the schedule is periodic with period $O(\log n)$: it can be thought of as restarting every $O(\log n)$ steps.*

Whenever we refer to computing or using *schedules* during our algorithms, we mean using the method from Lemma 82. We note that, as shown in Lemma 4.2 of [34], we can perform this preprocessing in such a way that it succeeds with high probability despite collisions, at a multiplicative $O(\log^{O(1)} n)$ time cost.

7.2.3 Algorithm Structure

The general approach of our algorithm proceeds as follows: First there is a preprocessing phase, in which we partition the network using PARTITION(β) from Lemma 80, and compute schedules within the clusters using Lemma 82. Then we broadcast the message through the network using these computed schedules within clusters. Any shortest (u, v)-path p crosses $O(|p|\beta)$ clusters in expectation, and communication within these clusters takes $O(\frac{\log n}{\beta \log D})$ expected time, so total time required should be $O(|p|\frac{\log n}{\log D}) = O(D\frac{\log n}{\log D})$.

Of course, this omits many of the technical details, and we encounter several difficulties when trying to implement the approach. Firstly, Theorem 81 only bounds expected distance to cluster center with constant probability. However, by generating many different clusterings, with different random values of β, and curtailing application of the schedules after $O(\frac{\log n}{\beta \log D})$ time, we can ensure that we do make sufficient progress with high probability. A second issue is that these values of β must somehow be coordinated, which we solve by using an extra layer of "coarse" clusters, similarly to [34]. Thirdly, collisions can occur between nodes of different clusters during both precomputation and broadcasting phases. We take several measures to deal with these collisions in our algorithms and analysis.

7.2.4 Advances over Previous Works

The idea of performing some precomputation locally and then using this local knowledge to perform a global task occurs frequently in distributed computing. In our setting, the most similar prior work is the $O(D\frac{\log n \log\log n}{\log D} + \log^{O(1)} n)$-time broadcasting algorithm due to Haeupler and Wajc [34]. Here we summarize our main technical differences from that paper and other related works:

- It was known from [34] that when PARTITION(β) is run with $1/\beta$ randomly selected from a range polynomial in D, the expected distance from a node to its cluster center is $O(\frac{\log n \log\log n}{\beta \log D})$. We improve this result with

Theorem 81, which states that with constant probability this distance is $O(\frac{\log n}{\beta \log D})$.

- We demonstrate how, by switching clusterings frequently and curtailing their schedules after $O(\frac{\log n}{\beta \log D})$ time, we can improve the fastest time for broadcasting in radio networks.

- We show that, with a different method of analysis and an algorithmic background process to deal with collisions, we can extend this method to also complete leader election, a task usually considered to be more difficult.

7.3 Algorithm for Compete

As mentioned in Section 7.5, since our broadcasting and leader election protocols require the same asymptotic running time and use similar methods, we combine their workings into a single generalized procedure Compete.

Compete takes as input a source set $S \subseteq V$ of nodes, in which every source $s \in S$ has a message it wishes to propagate, and guarantees, with high probability, that upon completion all nodes know the highest-valued source message. The process takes $O(D\frac{\log n}{\log D} + |S|D^{0.125} + \log^{O(1)} n)$ time (cf. Theorem 83), which is within the $O(D\frac{\log n}{\log D} + \log^{O(1)} n)$ time claimed for broadcasting and leader election, as long as $|S| = O(D^{0.875})$. Here this constant exponent of D is somewhat arbitrary, and could be improved by modifying constants in our algorithm and analysis, but this value is sufficient for our needs.

Our efficient algorithm for Compete consists of two processes which run concurrently, alternating between steps of each. The main Compete process is designed to propagate messages quickly through most of the network, and the background process is slower, with the purpose of "papering over the cracks" in the main process; in this case that means passing messages across coarse cluster boundaries.

Algorithm 22 COMPETE(S)

1) Compute a *coarse clustering* using PARTITION(β') with $\beta' = D^{-0.5}$.

2) Compute a schedule within each coarse cluster.

3) Within each coarse cluster, for each integer $j \in [0.01 \log D, 0.1 \log D]$, compute $D^{0.2}$ different *fine-grained clusterings* using PARTITION(β) with $\beta = 2^{-j}$.

4) Compute schedules within all fine-grained clusterings.

5) Each coarse cluster center computes a D-length sequence of randomly chosen fine-grained clusterings to use.

6) Transmit this sequence within each coarse cluster, using the coarse cluster schedules.

7) For each fine-grained clustering in the sequence perform INTRA-CLUSTER PROPAGATION($O(\frac{\log n}{\beta \log D})$), with β corresponding to the clustering.

In the main process, we first compute a *coarse clustering*, that is, one with comparatively large clusters, which we need to spread shared randomness. Then, within the coarse clusters we compute many different *fine-grained clusterings*, i.e., sub-clusterings with smaller clusters. These are the clusterings we will use to propagate information through the network. We will henceforth refer to the clusters of the fine-grained clusterings as fine-grained clusters.

The coarse clusters generate and transmit a random sequence of these fine-grained clusterings, which tells their members in what order to use the fine-grained clusterings for this propagation (this was the sole purpose of the coarse clustering). We show that, upon using a clustering with β chosen at random and applying INTRA-CLUSTER PROPAGATION($O(\frac{\log n}{\beta \log D})$), we have a constant probability of making sufficient progress towards our goal of information propagation. We can treat the progress made during each application of INTRA-CLUSTER PROPAGATION as being independent, since we use a different random clustering each time (and with high probability, whenever we choose a clustering we have used before, we have made sufficient progress in between so that the clusters we are analyzing are far apart and behave independently). Therefore we can use a Chernoff bound to show that with high probability we make sufficient progress throughout the algorithm as a whole.

An issue with the main process, though, is that at the boundaries of the coarse clustering, collisions between coarse clusters can cause INTRA-CLUSTER PROPAGATION to fail. To rectify this, we *interleave steps of the main process with steps of a background process*, e.g., by performing the main process during

even time-steps and the background process during odd time-steps.

Algorithm 23 COMPETE(S) - BACKGROUND PROCESS

1) Compute $D^{0.2}$ different *fine-grained clusterings* using PARTITION(β) with $\beta = D^{-0.1}$.

2) Compute a schedule within each cluster, for each clustering.

3) Cycling through clusterings in round-robin order, perform INTRA-CLUSTER PROPAGATION($O(\frac{\log n}{\beta})$)

The background process is simpler: it follows a similar line to the main process, but does not use a coarse clustering, only fine-grained clusterings. This means that we do not have the shared randomness we use in the main process, so we cannot choose β randomly (we instead fix $\beta = D^{-0.1}$) and we cannot use a random ordering of fine-grained clusterings (we instead use a round-robin order). As a result, we must run INTRA-CLUSTER PROPAGATION for longer to achieve a constant probability of making good progress, and so the propagation of information is slower (if we were to rely on the background process alone, we would only achieve $O(D \log n + \log^{O(1)} n)$ time for COMPETE).

However, the upside is that there are no coarse cluster boundaries, and so the progress is made consistently throughout the network. Therefore, we can analyze the progress of our algorithm using the faster main process most of the time, and switching to analysis of the background process when the main process reaches a coarse cluster boundary. Since the coarse clusters are comparatively large, their boundaries are reached infrequently, and so we can show that overall the algorithm still makes progress quickly.

Both COMPETE processes make use of INTRA-CLUSTER PROPAGATION as a primitive, which utilizes the computed clusters and schedule to propagate information. Specifically, the procedure facilitates communication between the cluster center and nodes within ℓ hops.

Algorithm 24 INTRA-CLUSTER PROPAGATION(ℓ)

Broadcast the highest message known by the cluster center to all nodes within ℓ distance.

All such nodes which know a higher message participate in a broadcast towards the cluster center.

Broadcast the highest message known by the cluster center to all nodes within ℓ distance.

Here we apply Lemma 82: after computing schedules, it is possible to broadcast between the cluster center and nodes at distance at most ℓ in time $O(\ell + \log^{O(1)} n)$. That is, on an outward broadcast all nodes within distance ℓ of the cluster center hear its message, and on an inward broadcast the cluster center hears the message of *at least one* participating node. This would be sufficient in isolation, but since we perform Intra-Cluster Propagation within all fine-grained clusters at the same time, we will describe a background process to deal with collisions between fine-grained clusters in the same coarse cluster. As before, we intersperse the steps of the main process and background process, performing one step of each alternately.

Algorithm 24 Intra-Cluster Propagation Background Process

Repeat until main process is complete:
for $i = 1$ **to** $\log n$ **do**
with probability 2^{-i} (coordinated within clusters) perform one round of Decay;
otherwise remain silent for $\log n$ steps.
end for

The background process aims to individually inform nodes that border other fine-grained clusters, and therefore may have collisions that prevent them from participating properly in the main process. The goal is to ensure that eventually (we will bound the amount of time that we may have to wait), such a node's cluster will be the only neighboring cluster to perform Decay (algorithm 12), which ensures that the node will then hear its cluster's message (with constant probability).

7.4 Analysis of Compete Algorithm

In this section we prove the following guarantee on the behavior of Compete:

Theorem 83. Compete(S) *informs all nodes of the highest message in S within $O(\frac{D \log n}{\log D} + |S| D^{0.125} + \log^{O(1)} n)$ time-steps, with high probability.*

The precomputation phase of Compete, that is, steps 1–6 of the main process and steps 1-2 of the background process, requires $O(D^{0.5} \log^{O(1)} n) = O(D)$ time, and upon its completion we have all the schedules required to perform Intra-Cluster Propagation. As in [34], we can ignore collisions during

these precomputation steps, since we can simulate each transmission step with $O(\log n)$ rounds of DECAY to ensure their success without exceeding $O(D)$ total time.

We first prove a result that allows us to use INTRA-CLUSTER PROPAGATION to propagate messages through the network. During a fixed application of INTRA-CLUSTER PROPAGATION, we call a node *valid* if it can correctly send and receive messages to/from its cluster center despite collisions between fine-grained clusters.

Lemma 84. *For some constant c, upon applying* INTRA-CLUSTER PROPAGATION(ℓ) *with $\ell = D^{\Omega(1)}$, a fixed node u at distance at most $\frac{\ell}{c}$ from its cluster center is valid with probability at least* 0.99*.*

Proof. Let u be a node at distance d from its cluster center, and call nodes on the shortest path from u to the cluster center that border another fine-grained cluster *risky*. We make use of a result of [34] (a corollary of Lemma 3.6 used during the proof of Lemma 4.6) which states that any node is risky with probability $O(\beta)$. Therefore the expected number of risky nodes on the path is $O(d\beta)$.

Let v be a risky node bordering q fine-grained clusters, and consider how long v must wait to be informed if it has a neighbor in its own cluster that wishes to inform it. Whenever 2^{-i} is within a constant factor of $\frac{1}{q}$ during the background process, DECAY has $\Omega(\frac{1}{q})$ probability of informing v from its own cluster. This is because with probability $\Omega(\frac{1}{q})$, v's cluster is the only cluster bordering v to perform DECAY, and in this case v is informed with constant probability. Since this value of 2^{-i} recurs every $O(\log^2 n)$ steps, the time needed to inform v is $O(q \log^2 n)$ in expectation.

We use another result from [34], Corollary 3.9, which states that with high probability all nodes border $O(\frac{\log n}{\log D}) = O(\log n)$ clusters. Therefore the total amount of time spent informing risky nodes is $O(d\beta \cdot \log^3 n) = O(d)$ in expectation, and since $O(d + \log^{O(1)} n)$ time is required to inform non-risky nodes using the main process, u can communicate with its cluster center in $O(d + \log^{O(1)} n)$ expected time. By choosing sufficiently large c, by Markov's inequality v is valid with probability at least 0.99. □

This will allow us to use INTRA-CLUSTER PROPAGATION to propagate information locally. To make a global argument, we will analyze the COMPETE algorithm's progress along paths by partitioning said paths into length $D^{0.12}$

subpaths. We call the set of all nodes within distance $D^{0.11}$ of a subpath its *neighborhood*, and we call a subpath *good* if all nodes in its neighborhood are in the same coarse cluster (and *bad* otherwise). We will show that we pass messages along good subpaths quickly under the main Compete process, and along bad subpaths more slowly under the background process.

For each pair of vertices, fix a canonical shortest path between them. When we refer to 'all shortest paths' we mean just these canonical paths, not all others of the same length. To show that there are not too many bad subpaths along these shortest paths, we make use of the following result from [34]:

Lemma 85 (Corollary 3.8 of [34])**.** *After running* Partition(β) *the probability of a fixed node u having nodes from t distinct clusters at distance d or less from u is at most* $(1 - e^{-\beta(2d+1)})^{t-1}$. □

Therefore the probability of a node u having nodes from two different coarse clusters within $D^{0.11}$ distance is at most

$$1 - e^{-D^{-0.5}(2D^{0.11}+1)} \leq 1 - e^{-3D^{-0.39}} \leq 3D^{-0.39} \ .$$

Taking the union bound over all nodes in a subpath, we find that any length-$D^{0.12}$ subpath is bad with probability upper bounded by $D^{0.12} \cdot 3D^{-0.39} \leq D^{-0.26}$.

Lemma 86. *All shortest paths p have $O(D^{0.63})$ bad subpaths, with high probability.*

Proof. Fix some shortest path p. As in the proof of Lemma 4.3 of [34], we first condition on the event that all exponentially distributed random variables δ_v used when computing the coarse clustering are at most $2D^{0.5} \log n$, which is the case with high probability. Then, the events that two length-$D^{0.12}$ subpaths of distance at least $5D^{0.5} \log n$ apart are bad are independent, since they are not affected by any of the same δ_v. If we label the length-$D^{0.12}$ subpaths of p in order from one end of the path to the other, and group them by label mod $6D^{0.38} \log n$, then the badness of every subpath is independent from all the others in its group. Hence, the number of bad subpaths in each group is binomially distributed, and is $O(\frac{D}{D^{0.12} \cdot 6D^{0.38} \log n} \cdot D^{-0.26}) = O(D^{0.24})$ with high probability by a Chernoff bound. By the union bound over all of the groups, the total number of bad subpaths is $O(D^{0.62})$ with high probability. If we allow this amount to be as high as $O(D^{0.63})$, we can reduce the probability that we exceed

it to n^{-c} for an arbitrarily large constant c. We can then take a union bound over all n^2 shortest paths, and find that they all have $O(D^{0.63})$ bad subpaths with high probability. □

Having bounded the number of bad subpaths, we can show we can pass messages along them using the background process, quickly enough that we do not exceed the algorithm's stated running time in total. Note that here, and henceforth, we will refer to messages by their place in increasing lexicographical order out of all messages of nodes in S. That is, by *message j* we mean the j^{th} highest message in S.

Lemma 87 (***Bad subpaths***)**.** *Let p be any (u, v)-subpath of length at most $D^{0.12}$. Let j be the minimum, over all nodes v in p's neighborhood, of the highest message known by v at time-step t. If, at timestep t, u knows a message higher than j, then by time-step $t' = t+O(D^{0.121})$ all nodes in p know a message at least as high as $j + 1$ with high probability.*

Proof. We analyze only the background process, and consider separately each fine-grained clustering used in the sequence between time-steps t and t'. For any such clustering, let w be the furthest node along p which knows a message at least as high as $j + 1$. We call the clustering *good* if:

- all nodes in w's cluster are $O(D^{0.1} \log n)$ distance from the cluster center;
- the node x which is $\frac{D^{0.1}}{c}$ nodes along p from w is in its cluster as w;
- x and w are valid (recall that this means they succeed in Intra-Cluster Propagation).

By Lemma 80 the first event occurs with high probability, by Corollary 3.7 of [34] we can make the probability of the second event an arbitrarily high constant by our choice of c, and by Lemma 84 and a union bound, the third event occurs with probability at least $1 - 2(1 - 0.99) = 0.98$, conditioned on the first. Therefore the clustering is good with probability at least $\frac{1}{2}$, by applying a union bound again.

By a Chernoff bound, $\Omega(D^{0.02})$ of the clusterings applied between times t and t' will be good. Consider each good clustering in turn. After applying such a clustering, w's cluster will be informed of an ID higher than j. Every time this occurs, w advances at least $\frac{D^{0.1}}{c}$ steps, and so by time t' the entire path knows a message at least as high as $j + 1$. □

We now make a similar argument for the good subpaths, but since we can use the main COMPETE process without fear of collisions from other coarse clusters, we get a better time bound:

Lemma 88 (***Good subpaths***)**.** *Let p be any good (u,v)-path of length at most $D^{0.12}$. Let j be the minimum, over all nodes v within $D^{0.11}$ distance p, of the highest message known by v at time-step t. If, at timestep t, u knows a message higher than j, then by time-step $t' = t + O(D^{0.12}\frac{\log n}{\log D})$ all nodes in p know a message at least as high as $j+1$ with high probability.*

Proof. We analyze only the main procedure, and consider separately each fine-grained clustering used in the sequence between time-steps t and t'. For any such clustering, let w be the furthest node along p which knows a message at least as high as $j+1$. We call the clustering *good* if:

- w is at distance at most $c_1\frac{\log n}{\beta \log D}$ from its cluster center;
- the node x which is $\frac{D^{0.1}}{c}$ nodes along p from w is in the same cluster as w;
- x and w are valid (recall that this means they succeed in INTRA-CLUSTER PROPAGATION).

By Theorem 81, and using Markov's inequality, we can choose c_1 such that the first event occurs with probability at least 0.54, conditioned on all previous randomness. By Corollary 3.7 of [34], we can choose c_2 so that the second event occurs with probability at least 0.99, also conditioned on all previous randomness. By Lemma 84 the probability that x and w are valid, conditioned on the first event, is at least 0.98. Therefore each fine-grained clustering is good with probability at least $\frac{1}{2}$ (by the union bound).

Let S be the set of all clusterings applied between time-steps t and t'. We are interested in the quantity $\sum_{s\in S \text{ is good}} \beta_s^{-1}$. Note that this majorizes the quantity $\sum_{s\in S} x_s$, where the x_s are independent Bernoulli variables which take value β_s^{-1} with probability $\frac{1}{2}$ and 0 otherwise. The expected value of this quantity is $\frac{1}{2}\sum_{s\in S \text{ is good}} \beta_s^{-1} \geq \frac{c}{3}D^{0.12}$. By Hoeffding's inequality,

$$\mathbf{Pr}\left[\sum_{s\in S} x_s \leq \frac{c}{6}D^{0.12}\right] \leq e^{-\frac{2|S|^2(\frac{c}{6}D^{0.12})^2}{\sum_{s\in S}\beta_s^{-2}}} \leq e^{-\frac{2|S|(\frac{c}{6}D^{0.12})^2}{D^{0.1}}} \leq e^{-\log^2 n} .$$

By time t', w has advanced at least $\frac{\sum_{s\in S} x_s}{c_2} \geq \frac{c}{6}D^{0.12}$ steps along p, and so

by choosing a sufficiently large constant in the big-Oh notation for t', we can ensure that every node in p knows a message at least as high as $j+1$. □

We combine the results from Lemmas 86–88 to show how to propagate messages along any shortest path between two nodes.

Lemma 89 (***All shortest paths***)**.** *Let u and v be any nodes in $\mathfrak{N}$, p be the (canonical) shortest (u,v)-path, and let b be the number of bad length-$D^{0.12}$ subpaths of p. If u knows a message at least as high as i at time-step t, then by time-step $t + O(\frac{|p|\log n}{\log D} + (i+b)D^{0.125})$, v knows a message at least as high as i with high probability.*

Proof. Let k be the maximum of the constants implied by the asymptotic notation of Lemmas 87 and 88. We will prove the claim of the lemma at time-step $t + k(\frac{|p|\log n}{\log D} + (2i+b)D^{0.125})$, using a nested induction. Our 'outer' induction shall be on the value i.

Base case: $i = 1$. Path p trivially contains at most $\frac{|p|}{D^{0.12}}$ good sub-paths, and b bad sub-paths. Applying Lemmas 87–88, the time taken to inform v of a message at least as high as 1 is at most

$$\frac{|p|}{D^{0.12}} \cdot kD^{0.12}\frac{\log n}{\log D} + b \cdot kD^{0.121} \leq k\left(\frac{|p|\log n}{\log D} + (2i+b)D^{0.125}\right) \ .$$

Inductive step: We can now assume the claim for $i = \ell - 1$ (Inductive Assumption 1), and prove the inductive step $i = \ell$. We do this using a second induction, on $|p|$.

Induction on $|p|$. Base case: $|p| \leq D^{0.12}$. Path p is a single subpath. If p is good, then by Inductive Assumption 1, all nodes within $D^{0.11}$ of p know an ID at least as high as $\ell - 1$ by time-step

$$t + k\left(\frac{(|p| + D^{0.11})\log n}{\log D} + 2(\ell-1)D^{0.125}\right) \ .$$

Then, by Lemma 88, v knows an ID at least as high as ℓ by time-step

$$\begin{aligned} &t + k\left(\frac{(|p| + D^{0.11})\log n}{\log D} + 2(\ell-1)D^{0.125}\right) + kD^{0.12}\frac{\log n}{\log D} \\ \leq\ &t + k\left(\frac{|p|\log n}{\log D} + 2\ell D^{0.125}\right) \ . \end{aligned}$$

If p is bad then by Inductive Assumption 1, all nodes within $D^{0.11}$ of p know an ID at least as high as $\ell - 1$ by time-step

$$t + k\left(\frac{(|p| + D^{0.11})\log n}{\log D} + (2(\ell-1)+1)D^{0.125}\right) \quad .$$

Then, by Lemma 87, v knows an ID at least as high as i by time-step

$$\begin{aligned} & t + k\left(\frac{(|p| + D^{0.11})\log n}{\log D} + (2\ell-1)D^{0.125}\right) + kD^{0.121} \\ \leq\ & t + k\left(\frac{|p|\log n}{\log D} + (2\ell+1)D^{0.125}\right) \quad . \end{aligned}$$

Induction on $|p|$. Inductive step: Having proved the base case, we can now assume the claim for $i = \ell$ and $|p| < q$ (Inductive Assumption 2), and prove the inductive step $|p| = q$.

Let u' be the start node of the last subpath of p. If this subpath is good, then by Inductive Assumption 2, u' knows an ID at least as high as ℓ by time-step

$$t + k\left(\frac{(|p| - D^{0.12})\log n}{\log D} + (2\ell+b)D^{0.125}\right) \quad .$$

By Inductive Assumption 1, all nodes within $D^{0.11}$ of p know a message at least as high as $\ell - 1$ by time-step

$$\begin{aligned} & t + k\left(\frac{(|p| + D^{0.11})\log n}{\log D} + (2(\ell-1)+b+1)D^{0.125}\right) \\ \leq\ & t + k\left(\frac{(|p| - D^{0.12})\log n}{\log D} + (2\ell+b)D^{0.125}\right) \quad . \end{aligned}$$

Therefore, by Lemma 88, v knows a message at least as high as ℓ by time-step

$$\begin{aligned} & t + k\left(\frac{(|p| - D^{0.12})\log n}{\log D} + (2\ell+b)D^{0.125}\right) + kD^{0.12}\frac{\log n}{\log D} \\ =\ & t + k\left(\frac{|p|\log n}{\log D} + (2\ell+b)D^{0.125}\right) \quad . \end{aligned}$$

If the subpath is bad, then by Inductive Assumption 2, u' knows an ID at

least as high as ℓ by time-step

$$\begin{aligned} & t + k\left(\frac{(|p| - D^{0.12})\log n}{\log D} + (2\ell + b - 1)D^{0.125}\right) \\ \leq\ & t + k\left(\frac{(|p| + D^{0.11})\log n}{\log D} + (2\ell + b - 1)D^{0.125}\right) \ . \end{aligned}$$

By Inductive Assumption 1, all nodes within $D^{0.11}$ of p know a message at least as high as $\ell - 1$ by time-step

$$t + k\left(\frac{(|p| + D^{0.11})\log n}{\log D} + (2(\ell - 1) + b)D^{0.125}\right) \ .$$

Therefore, by Lemma 87, v knows a message at least as high as ℓ by time-step

$$\begin{aligned} & t + k\left(\frac{(|p| + D^{0.11})\log n}{\log D} + (2\ell - 2 + b)D^{0.125}\right) + kD^{0.121} \\ \leq\ & t + k\left(\frac{|p|\log n}{\log D} + (2\ell + b)D^{0.125}\right) \ . \end{aligned}$$

This completes the proof of Lemma 89 by induction. □

We are now ready to prove Theorem 83:

Proof of Theorem 83. The precomputation phase takes at most $O(D+\log^{O(1)} n)$ time. Upon beginning the Intra-Cluster Propagation phase, one node u knows the highest message. Therefore by Lemma 89, each node v hears this message within $O(\frac{dist(u,v)\log n}{\log D} + (|S| + b)D^{0.125})$ time-steps, with high probability. By Lemma 86, $b = O(D^{0.63})$ for all nodes v, and so total running time is $O(\frac{D\log n}{\log D} + |S|D^{0.125} + \log^{O(1)} n)$. □

7.5 Applying Compete to Broadcasting and Leader Election

It is not difficult to see that Compete can be used to perform both broadcasting and leader election.

Theorem 90. Compete($\{s\}$) *completes broadcasting in* $O(D\frac{\log n}{\log D} + \log^{O(1)} n)$ *time with high probability.*

Proof. COMPETE informs all nodes of the highest message in the message set in time $O(D\frac{\log n}{\log D} + \log^{O(1)} n)$, with high probability. Since this set contains only the source message, broadcasting is completed. □

Algorithm 25 LEADER ELECTION

1) Nodes choose to become candidates in C with probability $\Theta(\frac{\log n}{n})$.

2) Candidates randomly generate $\Theta(\log n)$-bit IDs.

3) Perform COMPETE(C).

Theorem 91. *Algorithm 25 completes leader election within time* $O(D\frac{\log n}{\log D} + \log^{O(1)} n)$, *with high probability*

Proof. With high probability $|C| = \Theta(\log n)$ and all candidate IDs are unique. Conditioning on this, COMPETE informs all nodes of the highest candidate ID within time $O(D\frac{\log n}{\log D} + \log^{O(1)} n)$, with high probability. Therefore leader election is completed. □

7.6 Clustering property: Proof of Theorem 81

In this section we prove the last remaining part of our analysis, a key property of the clustering method in our algorithm, PARTITION(β), as described in Theorem 81.

What we must show to prove Theorem 81 is that if j is an integer chosen uniformly at random from the interval $[0.01 \log D, 0.1 \log D]$, and if $\beta = 2^{-j}$, then in algorithm PARTITION(β) as described above, for any node v, with probability at least 0.55 (over choice of j), the expected distance from v to its cluster center upon applying PARTITION(β) is $O(\frac{\log n}{\beta \log D})$.

7.6.1 Bounding Expected Distance to Cluster Center

Our first step in proving Theorem 81 is to obtain a bound on the distance to the cluster center which is based upon the number of nodes at each distance layer from v. To this purpose, let $A_i(v)$ be the set of nodes at distance i from v and denote $x_i = |A_i(v)|$. Denote $\boldsymbol{x} \in \mathbb{N}_0^D$ to be the vector with these x_i as coefficients.

Denote $T_{\boldsymbol{x},\beta} = \sum_{i=0}^{D} i x_i e^{-i\beta}$ and $B_{\boldsymbol{x},\beta} = \sum_{i=0}^{D} x_i e^{-i\beta}$. Denote

$$S_{\boldsymbol{x},\beta} = \frac{T_{\boldsymbol{x},\beta}}{B_{\boldsymbol{x},\beta}} = \frac{\sum_{i=0}^{D} i x_i e^{-i\beta}}{\sum_{i=0}^{D} x_i e^{-i\beta}} \ .$$

These quantities will be used in the following key auxiliary lemma describing the expected distance from any fixed v to its cluster center after applying Partition(β).

Lemma 92. *For any fixed node v and value β with $D^{-0.01} \leq \beta \leq D^{-0.1}$, the expected distance from v to its cluster center upon applying* Partition(β) *is at most* $\frac{5\sum_{i=0}^{D} i x_i e^{-i\beta}}{\sum_{i=0}^{D} x_i e^{-i\beta}} = 5S_{\boldsymbol{x},\beta}$.

Proof. We compute the expected distance to cluster center:

$$\begin{aligned}
&\mathbb{E}[\text{distance from } v \text{ to its cluster center}] \\
&\qquad = \sum_{i=0}^{D} i \cdot \mathbf{Pr}\left[v\text{'s cluster center is distance } i \text{ away}\right] \\
&\qquad = \sum_{i=1}^{D} i \cdot \left(\sum_{u \in A_i(v)} \mathbf{Pr}\left[u \text{ is } v\text{'s cluster center}\right] \right) \ .
\end{aligned}$$

We concentrate on this latter probability and henceforth fix $u \in A_i(v)$ to be some node at distance i from v. For simplicity of notation, let $\mathcal{P}_{u,v}$ denote $\mathbf{Pr}\left[u \text{ is } v\text{'s cluster center}\right]$. We note that

$$\mathcal{P}_{u,v} = \int_{i}^{\infty} \beta e^{-\beta p} \mathbf{Pr}\left[u \text{ is } v\text{'s cluster center} | \delta_u = p\right] dp$$

by conditioning on the value of δ_u over its whole range and multiplying by the corresponding probability density function (we can start the integral at i since if $\delta_u < i$ then u cannot be v's cluster center).

Having conditioned on the value of δ_u, we can evaluate the probability that u is v's cluster center based on the random variables generated by other nodes. Since the events that each other node 'beats' u are now independent, $\mathcal{P}_{u,v}$ is equal to:

$$\int_{i}^{\infty} \beta e^{-\beta p} \prod_{w \neq u} \mathbf{Pr}\left[\delta_w - dist(v,w) < \delta_u - dist(v,u) | \delta_u = p\right] dp \ .$$

We can simplify by grouping the nodes w based on distance from v, though we must be careful to include a $\frac{1}{\mathbf{Pr}[\delta_u < p]}$ term to cancel out u's contribution to the resulting product:

$$\mathcal{P}_{u,v} = \int_i^\infty \frac{\beta e^{-\beta p}}{\mathbf{Pr}\left[\delta_u < p\right]} \prod_{k=0}^{D} \prod_{w \in A_k(v)} \mathbf{Pr}\left[\delta_w - k < p - i\right] dp \ .$$

Plugging in the cumulative distribution function for the δ_w yields:

$$\mathcal{P}_{u,v} = \int_i^\infty \frac{\beta e^{-\beta p}}{1 - e^{-\beta p}} \prod_{k=0}^{D} \prod_{w \in A_k(v)} 1 - e^{-\beta(p-i+k)} dp \ .$$

We use the standard inequality $1 - y \leq e^{-y}$ for $y \in [0, 1]$, here setting $y = e^{-\beta(p-i+k)}$, and account for the second product by taking the contents to the power of x_k:

$$\begin{aligned}
\mathcal{P}_{u,v} &\leq \int_i^\infty \frac{\beta e^{-\beta p}}{1 - e^{-\beta p}} \prod_{k=0}^{D} \prod_{w \in A_k(v)} e^{-e^{-\beta(p-i+k)}} dp \\
&= \int_i^\infty \frac{\beta e^{-\beta p}}{1 - e^{-\beta p}} \prod_{k=0}^{D} e^{-e^{-\beta(p-i+k)} x_k} dp \ .
\end{aligned}$$

We can also remove the remaining product by taking it as a sum into the exponent, and re-arranging some terms yields:

$$\begin{aligned}
\mathcal{P}_{u,v} &\leq \int_i^\infty \frac{\beta e^{-\beta p}}{1 - e^{-\beta p}} e^{-e^{\beta(i-p)} \sum_{k=0}^{D} x_k e^{-\beta k}} dp \\
&= \int_i^\infty \frac{\beta e^{-\beta p}}{1 - e^{-\beta p}} e^{-e^{\beta(i-p)} B_{\boldsymbol{x},\beta}} dp \ ,
\end{aligned}$$

where for succinctness we use our definition $B_{\boldsymbol{x},\beta} = \sum_{i=0}^{D} x_i e^{-i\beta}$.

At this point we split the integral and bound the parts separately, since they exhibit different behavior:

$$\mathcal{P}_{u,v} \leq J + K \ ,$$

where,

$$J = \int_i^{\frac{1}{\beta}} \frac{\beta e^{-\beta p}}{1 - e^{-\beta p}} e^{-e^{\beta(i-p)} B_{\boldsymbol{x},\beta}} dp \quad \text{and} \quad K = \int_{\frac{1}{\beta}}^\infty \frac{\beta e^{-\beta p}}{1 - e^{-\beta p}} e^{-e^{\beta(i-p)} B_{\boldsymbol{x},\beta}} dp \ .$$

To bound J, we make use of the following bound on $B_{\boldsymbol{x},\beta}$:

$$B_{\boldsymbol{x},\beta} = \sum_{k=0}^{D} x_k e^{-k\beta} \geq \sum_{k=0}^{\lceil \frac{D}{2} \rceil} e^{-k\beta} \geq \int_{-1}^{\frac{D}{2}} e^{-z\beta} dz = \frac{-1}{\beta}(e^{-\frac{\beta D}{2}} - e^{-\beta}) \geq \frac{1}{2\beta} \ .$$

This gives

$$J \leq \int_{i}^{\frac{1}{\beta}} \frac{\beta e^{-\beta p}}{1 - e^{-\beta p}} e^{-e^{\beta(i-p)} \frac{1}{2\beta}} dp \ .$$

Since $e^{\beta(i-p)} \geq e^{-1}$, we obtain,

$$J \leq \int_{1}^{\frac{1}{\beta}} \frac{\beta e^{-\beta p}}{1 - e^{-\beta p}} e^{-\frac{1}{2e\beta}} dp = \beta e^{-\frac{1}{2e\beta}} \int_{1}^{\frac{1}{\beta}} \frac{e^{-\beta p}}{1 - e^{-\beta p}} dp \ .$$

We can then use that $\int_a^b \frac{e^{-\beta p}}{1-e^{-\beta p}} = \frac{1}{\beta} \log \frac{(1-e^{\beta b})}{(1-e^{\beta a})} + a - b$ to evaluate $J \leq e^{-\frac{1}{2e\beta}} \log \frac{(1-e)}{(1-e^{\beta})}$. Since $e^{\beta} > 1 + \beta$, re-arranging yields $J \leq e^{-\frac{1}{2e\beta}} \log \frac{e-1}{\beta}$. Finally, since we can assume that $\frac{1}{\beta} \geq \log^c n$ for some sufficiently large c, we obtain,

$$J \leq e^{-\frac{\log^2 n}{2e}} \log \frac{e-1}{\beta} \leq n^{-2} \ .$$

We now turn our attention to $K = \int_{\frac{1}{\beta}}^{\infty} \frac{\beta e^{-\beta p}}{1-e^{-\beta p}} e^{-e^{\beta(i-p)} B_{\boldsymbol{x},\beta}} dp$. Since $1 - e^{-\beta p} \geq 1 - e^{-1} > \frac{1}{2}$, we get

$$K < \int_{\frac{1}{\beta}}^{\infty} 2\beta e^{-\beta p} e^{-e^{-\beta p} e^{\beta i} B_{\boldsymbol{x},\beta}} dp \ .$$

Using that $e^{-e^{-\beta p}} \leq 1 - \frac{1}{2} e^{-\beta p}$ (since $0 \leq e^{-\beta p} \leq 1$), we obtain,

$$K < \int_{\frac{1}{\beta}}^{\infty} 2\beta e^{-\beta p} (1 - \frac{1}{2} e^{-\beta p})^{e^{\beta i} B_{\boldsymbol{x},\beta}} dp \ .$$

Evaluating the integral, using

$$\int_a^{\infty} e^{-\beta p} (1 - \frac{1}{2} e^{-\beta p})^c = \frac{(e^{-a\beta} - 2)(1 - \frac{1}{2} e^{-a\beta})^c + 2}{\beta(1 + c)} \ ,$$

we obtain

$$K < 2 \frac{(e^{-1} - 2)(1 - \frac{1}{2} e^{-1})^{e^{\beta i} B_{\boldsymbol{x},\beta}} + 2}{1 + e^{\beta i} B_{\boldsymbol{x},\beta}} \leq \frac{4}{e^{\beta i} B_{\boldsymbol{x},\beta}} \ .$$

We can now combine our calculations to prove the lemma. Since $x_i = |A_i(v)|$, we have: $\mathbb{E}$[distance from v to its cluster center] is at most

$$\begin{aligned}\sum_{i=1}^{D} i \sum_{u \in A_i(v)} \mathcal{P}_{u,v} &\leq \sum_{i=1}^{D} i x_i (J+K) \\ &< \sum_{i=1}^{D} i x_i \left(n^{-2} + \frac{4}{e^{\beta i} B_{\boldsymbol{x},\beta}} \right) \\ &\leq n^{-2} \sum_{i=1}^{D} D x_i + \frac{4 \sum_{i=1}^{D} i x_i e^{-\beta i}}{B_{\boldsymbol{x},\beta}} \\ &\leq \frac{D}{n} + 4 S_{\boldsymbol{x},\beta} \leq 5 S_{\boldsymbol{x},\beta} \ . \end{aligned}$$

□

7.6.2 Simplifying the Bound Via Transformations

By Lemma 92, we must now bound the value of $S_{\boldsymbol{x},\beta} = \frac{\sum_{i=0}^{D} i x_i e^{-i\beta}}{\sum_{i=0}^{D} x_i e^{-i\beta}}$. To simplify our analysis, we will apply two transformations to $\boldsymbol{x}$ which will provide us with useful properties for bounding, while not decreasing any $S_{\boldsymbol{x},\beta}$ by more than a constant factor.

First transformation

The first transformation we apply will be to collate coefficients of $\boldsymbol{x}$ into indices which are just the powers of 2. That is, we sum the coefficients of $\boldsymbol{x}$ over regions of doubling size.

Let $f : \mathbb{R}^{D+1} \to \mathbb{R}^{D+1}$ be given by

$$f(\boldsymbol{x})_i = \begin{cases} \sum_{\ell=2i}^{4i-1} x_\ell & \text{if } i = 2^k \text{ for some } k \in \mathbb{N}_0, \\ 0 & \text{otherwise.} \end{cases}$$

We can bound $S_{\boldsymbol{x},\beta}$ in terms of $S_{f(\boldsymbol{x}),\beta}$.

Lemma 93. *For all $\boldsymbol{x} \in \mathbb{N}_0^D$, $S_{\boldsymbol{x},\beta} \leq 11 S_{f(\boldsymbol{x}),\beta}$.*

Proof. We start with the following auxiliary claim.

Claim 94. *Consider an expression of the form $\frac{\sum_{i=0}^{D} i w_i}{\sum_{i=0}^{D} w_i}$, where all w_i are non-negative. Let p be an integer with $p < \frac{\sum_{i=0}^{D} i w_i}{\sum_{i=0}^{D} w_i}$. For all $i < p$ let $0 \leq w'_i \leq w_i$, and for all $i \geq p$ let $w'_i \geq w_i$. Then $\frac{\sum_{i=0}^{D} i w'_i}{\sum_{i=0}^{D} w'_i} > p$.*

Intuitively, consider $\frac{\sum_{i=0}^{D} iw_i}{\sum_{i=0}^{D} w_i}$ as a weighted average of the i (with weights w_i). The claim then says that for any p which is less than the value of the average, increasing the weights for indices higher than p and reducing them for indices lower than p cannot reduce the weighted average below p.

Proof of Claim 94.

$$\begin{aligned}
\frac{\sum_{i=0}^{D} iw'_i}{\sum_{i=0}^{D} w'_i} &= \frac{\sum_{i=0}^{D} iw_i + \sum_{i=0}^{D} i(w'_i - w_i)}{\sum_{i=0}^{D} w'_i} \\
&= \frac{\frac{\sum_{i=0}^{D} iw_i}{\sum_{i=0}^{D} w_i} \cdot \sum_{i=0}^{D} w_i + \sum_{i=0}^{p-1} i(w'_i - w_i) + \sum_{i=p}^{D} i(w'_i - w_i)}{\sum_{i=0}^{D} w_i + \sum_{i=0}^{D}(w'_i - w_i)} \\
&> \frac{p \cdot \sum_{i=0}^{D} w_i + \sum_{i=0}^{p-1} p(w'_i - w_i) + \sum_{i=p}^{D} p(w'_i - w_i)}{\sum_{i=0}^{D} w_i + \sum_{i=0}^{D}(w'_i - w_i)} \\
&= \frac{p \cdot \left(\sum_{i=0}^{D} w_i + \sum_{i=0}^{D}(w'_i - w_i)\right)}{\sum_{i=0}^{D} w_i + \sum_{i=0}^{D}(w'_i - w_i)} = p \ . \qquad \square
\end{aligned}$$

We apply Claim 94 to analyze the effect of the transformation f, in particular to compare $S_{f(\boldsymbol{x}),\beta}$ with $S_{\boldsymbol{x},\beta}$. First we find an expression for $S_{\boldsymbol{x},\beta}$ in a form for which we can use the claim:

$$S_{\boldsymbol{x},\beta} = \frac{\sum_{i=0}^{D} ix_i e^{-i\beta}}{\sum_{i=0}^{D} x_i e^{-i\beta}} = \frac{\sum_{i=0}^{D} iw_i}{\sum_{i=0}^{D} iw_i} \ ,$$

where $w_i = x_i e^{-i\beta}$.

Next we do the same for $S_{f(\boldsymbol{x}),\beta}$:

$$S_{f(\boldsymbol{x}),\beta} = \frac{\sum_{k=0}^{\log D} 2^k \sum_{\ell=2^{k+1}}^{2^{k+2}-1} x_\ell e^{-2^k\beta}}{\sum_{k=0}^{\log D} \sum_{\ell=2^{k+1}}^{2^{k+2}-1} x_\ell e^{-2^k\beta}} = \frac{\sum_{\ell=2}^{D} 2^{\lfloor \log \ell - 1 \rfloor} x_\ell e^{-2^{\lfloor \log \ell - 1 \rfloor}\beta}}{\sum_{\ell=2}^{D} x_\ell e^{-2^{\lfloor \log \ell - 1 \rfloor}\beta}} \ .$$

We multiply both the numerator and denominator by a scaling factor to make the expression more comparable to $S_{\boldsymbol{x},\beta}$. Let $q := \lfloor \log S_{\boldsymbol{x},\beta} \rfloor$. Our scaling factor will be $e^{-2^{q-1}}$.

$$S_{f(\boldsymbol{x}),\beta} = \frac{\sum_{\ell=2}^{D} 2^{\lfloor \log \ell - 1 \rfloor} x_\ell e^{-2^{\lfloor \log \ell - 1 \rfloor}\beta}}{\sum_{\ell=2}^{D} x_\ell e^{-2^{\lfloor \log \ell - 1 \rfloor}\beta}} \geq \frac{\sum_{\ell=2}^{D} \frac{l}{4} x_\ell e^{(-2^{q-1} - 2^{\lfloor \log \ell - 1 \rfloor})\beta}}{\sum_{\ell=2}^{D} x_\ell e^{(-2^{q-1} - 2^{\lfloor \log \ell - 1 \rfloor})\beta}} = \frac{\sum_{i=0}^{D} iw'_i}{4\sum_{i=0}^{D} w'_i} \ ,$$

where $w'_i = \begin{cases} x_i e^{(-2^{q-1}-2^{\lfloor \log i-1 \rfloor})\beta} & \text{if } i \geq 2, \\ 0 & \text{otherwise.} \end{cases}$

We set $p = 3 \cdot 2^{q-2}$, and verify that we meet all of the conditions of the Claim 94:

Firstly we need that all w_i and w'_i are non-negative, which is obviously the case.

Secondly we need that $p < \frac{\sum_{i=0}^{D} i w_i}{\sum_{i=0}^{D} w_i}$, which is true since

$$p < 2^q \leq S_{\boldsymbol{x},\beta} = \frac{\sum_{i=0}^{D} i w_i}{\sum_{i=0}^{D} w_i} \ .$$

Thirdly we need $w'_i \leq w_i$ for all $i < p$ and $w'_i \geq w_i$ for all $i \geq p$. To show this, note that

$$w'_i \geq w_i \iff (-2^{q-1} - 2^{\lfloor \log i-1 \rfloor})\beta \geq -i\beta \iff 2^{q-1} + 2^{\lfloor \log i-1 \rfloor} \leq i \ .$$

When $i \leq 2^{q-1}$, clearly $2^{q-1} + 2^{\lfloor \log i-1 \rfloor} > i$, so $w'_i \leq w_i$.

When $2^{q-1} < i < p$, $2^{q-1} + 2^{\lfloor \log i-1 \rfloor} = 2^{q-1} + 2^{q-2} = p > i$, so $w'_i \leq w_i$.

When $p \leq i < 2^q$, $2^{q-1} + 2^{\lfloor \log i-1 \rfloor} = 2^{q-1} + 2^{q-2} = p \leq i$, so $w'_i \geq w_i$.

When $2^q \leq i$, $2^{q-1} + 2^{\lfloor \log i-1 \rfloor} \leq 2^{q-1} + 2^{\log i-1} \leq 2^{q-1} + \frac{i}{2} \leq i$, so $w'_i \geq w_i$.

Therefore we have all the necessary conditions to apply Claim 94, yielding $\frac{\sum_{i=0}^{D} i w'_i}{\sum_{i=0}^{D} w'_i} > p$. Then,

$$S_{f(\boldsymbol{x}),\beta} \geq \frac{\sum_{i=0}^{D} i w'_i}{4 \sum_{i=0}^{D} w'_i} > \frac{p}{4} \geq \frac{3q}{16} > \frac{3 S_{\boldsymbol{x},\beta}}{32} > \frac{S_{\boldsymbol{x},\beta}}{11} \ .$$

This completes the proof of Lemma 93. □

Second transformation

Having applied f to ensure that only power-of-2 coefficients of $\boldsymbol{x}$ are non-zero, we apply a second transformation to ensure that the coefficients are not "too decreasing"; in particular, we guarantee that each power-of-2 coefficient is at least half the previous one.

Let $g : \mathbb{R}^{D+1} \to \mathbb{R}^{D+1}$ be given by

$$g(\boldsymbol{x})_i = \begin{cases} \sum_{\ell \le i} \frac{\ell x_\ell}{i} & \text{if } i = 2^k \text{ for some } k \in \mathbb{N}_0, \\ 0 & \text{otherwise.} \end{cases}$$

This definition achieves our aim since when i is a power of 2,

$$2g(\boldsymbol{x})_{2i} = 2\sum_{\ell \le 2i} \frac{\ell x_\ell}{2i} = \sum_{\ell \le 2i} \frac{\ell x_\ell}{i} \ge \sum_{\ell \le i} \frac{\ell x_\ell}{i} = g(\boldsymbol{x})_{2i} \ .$$

Similarly to Lemma 93, we can bound $S_{\boldsymbol{x},\beta}$ in terms of $S_{g(\boldsymbol{x}),\beta}$.

Lemma 95. *For all $\boldsymbol{x} \in \mathbb{N}_0^D$ which have $x_i = 0$ for all $i \notin \{2^k : k \in \mathbb{N}_0\}$, $S_{\boldsymbol{x},\beta} \le 2S_{g(\boldsymbol{x}),\beta}$.*

Proof. We start by taking our $S_{g(\boldsymbol{x}),\beta}$ expression and substituting the sum index to account only for powers of two, since all other coefficients are 0:

$$S_{g(\boldsymbol{x}),\beta} = \frac{\sum_{i=0}^{D} i g(\boldsymbol{x})_i e^{-i\beta}}{\sum_{i=0}^{D} g(x)_i e^{-i\beta}} = \frac{\sum_{k=0}^{\log D} 2^k g(\boldsymbol{x})_{2^k} e^{-2^k\beta}}{\sum_{k=0}^{\log D} g(\boldsymbol{x})_{2^k} e^{-2^k\beta}} \ .$$

We now substitute in the definition of $g(\boldsymbol{x})$, and switch order of summation in the denominator, applying some straightforward bounds in the process.

$$S_{g(\boldsymbol{x}),\beta} \ge \frac{\sum_{k=0}^{\log D} 2^k x_{2^k} e^{-2^k\beta}}{\sum_{k=0}^{\log D} \sum_{\ell=0}^{k} \frac{2^\ell x_{2^\ell}}{2^k} e^{-2^k\beta}} = \frac{\sum_{k=0}^{\log D} 2^k x_{2^k} e^{-2^k\beta}}{\sum_{\ell=0}^{\log D} \sum_{k=\ell}^{\log D} \frac{2^\ell x_{2^\ell}}{2^k} e^{-2^\ell\beta}} \ .$$

We simplify the denominator by noting that $\frac{2^\ell}{2^k} \le 1$, reaching an expression which matches $S_{\boldsymbol{x},\beta}$:

$$S_{g(\boldsymbol{x}),\beta} \ge \frac{\sum_{k=0}^{\log D} 2^k x_{2^k} e^{-2^k\beta}}{2\sum_{\ell=0}^{\log D} x_{2^\ell} e^{-2^\ell\beta}} \ge \frac{S_{\boldsymbol{x},\beta}}{2} \ . \qquad \square$$

7.6.3 Bounding After Transformation

Now that we have shown in Lemmas 93 and 95 that the transformations f and g do not decrease $S_{\boldsymbol{x},\beta}$ by more than a constant factor, we show how they help to bound the value of $S_{\boldsymbol{x},\beta}$. Let $\boldsymbol{x}'$ be the vector obtained after applying the two transformations to $\boldsymbol{x}$, i.e., $\boldsymbol{x}' = g \circ f(\boldsymbol{x})$. We begin with the following lemma.

Lemma 96. *$\boldsymbol{x}'$ has the following properties:*

- $x'_i = 0$ *for all* $i \notin \{2^k : k \in \mathbb{N}_0\}$*;*
- $x'_1 \geq 2$*;*
- $||\boldsymbol{x}'||_1 = \sum_{i=0}^{D} x'_i \leq 2n$*;*
- $2x'_{2i} \geq x'_i$ *for all* i*.*

Proof. The first property is obvious due to transformation f. The second is true since $x'_1 \geq f(x)_1 = x_2 + x_3 \geq 2$. The third is the case since f does not increase L_1-norm and g at most doubles it, and the fourth follows from transformation g. □

Our argument will be based on examining the ratios between consecutive non-zero (i.e. power-of-two) coefficients in $\boldsymbol{x}'$. To that end, define $k_i = \log \frac{x'_{2^{i+1}}}{x'_{2^i}}$ for all $i \leq \log D$, and note that $k_i \geq \log \frac{1}{2} = -1$ for all i and $\sum_{i=0}^{\log D} k_i \leq \log n$ by Lemma 96.

We first show a condition on these k_i which guarantees that $S_{\boldsymbol{x}',\beta}$ (and hence $S_{\boldsymbol{x},\beta}$) is $O(\frac{\log n}{\beta \log D})$ for some particular value of β:

Lemma 97. *If for fixed* j *and for all* $m \geq 8$ *we have*

$$\sum_{\ell=j+\log \frac{\log n}{\log D}}^{j+\log \frac{\log n}{\log D}+m} k_\ell \leq 2^m \frac{\log n}{\log D}$$

then $S_{\boldsymbol{x}',2^{-j}} = O(\frac{2^j \log n}{\log D})$.

The intuition behind this lemma is that the cluster center of a node is likely within our desired radius of $O(\frac{2^j \log n}{\log D})$ unless the network expands very rapidly just outside that radius.

Proof. We first split $T_{\boldsymbol{x}',2^{-j}}$ (the numerator of $S_{\boldsymbol{x}',2^{-j}}$) into three parts, which we will bound separately:

$$T_{\boldsymbol{x}',2^{-j}} = \sum_{i=0}^{D} i x'_i e^{-i2^{-j}} = \sum_{i=0}^{\log D} 2^i x'_{2^i} e^{-2^{i-j}} = P + Q + R \ ,$$

where $P = \sum_{i=0}^{j+\log \frac{\log n}{\log D}+8} 2^i x'_{2^i} e^{-2^{i-j}}$, $Q = \sum_{i=j+\log \frac{\log n}{\log}+9}^{j+\log \log n} 2^i x'_{2^i} e^{-2^{i-j}}$, and $R = \sum_{i=\log \log n+1}^{\log D} 2^i x'_{2^i} e^{-2^{i-j}}$.

We now bound these parts. P is the largest, and we require that $P = O(\frac{2^j \log n}{\log D}) B_{\boldsymbol{x}',2^{-j}}$ (recall that $B_{\boldsymbol{x}',2^{-j}} = \sum_{i=0}^{D} x'_i e^{-i2^{-j}}$).

$$P = \sum_{i=0}^{j+\log \frac{\log n}{\log D}+8} 2^i x'_{2^i} e^{-2^{i-j}} \leq \sum_{i=0}^{j+\log \frac{\log n}{\log D}+8} 256 \frac{2^j \log n}{\log D} x'_{2^i} e^{-2^{i-j}}$$
$$\leq 256 \frac{2^j \log n}{\log D} \sum_{i=0}^{\log D} x'_{2^i} e^{-2^{i-j}} = 256 \frac{2^j \log n}{\log D} B_{\boldsymbol{x}',2^{-j}} \ .$$

Using the condition of Lemma 97, we can show that Q is also $O(\frac{2^j \log n}{\log D}) B_{\boldsymbol{x}',2^{-j}}$. Let $m \geq 9$. We begin by re-expressing $x'_{\frac{2^{j+m} \log n}{\log D}}$:

$$x'_{\frac{2^{j+m} \log n}{\log D}} = x'_{\frac{2^j \log n}{\log D}} \prod_{\ell=j+\log \frac{\log n}{\log D}}^{j+\log \frac{\log n}{\log D}+m-1} \frac{x'_{2^{\ell+1}}}{x'_{2^\ell}} = x'_{\frac{2^j \log n}{\log D}} 2^{\sum_{\ell=j+\log \frac{\log n}{\log D}}^{j+\log \frac{\log n}{\log D}+m-1} k_\ell} \ .$$

We can then apply the condition of the Lemma:

$$x'_{\frac{2^{j+m} \log n}{\log D}} \leq x'_{\frac{2^j \log n}{\log D}} 2^{2^{m-1} \frac{\log n}{\log D}} \ .$$

We make some re-arrangements to reach a form containing $B_{\boldsymbol{x}',2^{-j}}$:

$$x'_{\frac{2^{j+m} \log n}{\log D}} \leq e^{\frac{2^j \log n}{\log D} 2^{-j}} 2^{2^{m-1} \frac{\log n}{\log D}} x'_{\frac{2^j \log n}{\log D}} e^{-\frac{2^j \log n}{\log D} 2^{-j}}$$
$$\leq e^{\frac{\log n}{\log D}} 2^{2^{m-1} \frac{\log n}{\log D}} \sum_{i=0}^{D} x'_i e^{-i2^{-j}}$$
$$= 2^{(2^{m-1}+\log e) \frac{\log n}{\log D}} B_{\boldsymbol{x}',2^{-j}} \ .$$

We can use this to bound Q as follows:

$$Q = \sum_{i=j+\log \frac{\log n}{\log D}+9}^{j+\log \log n} 2^i x'_{2^i} e^{-2^{i-j}} = \frac{2^j \log n}{\log D} \sum_{m=9}^{\log \log n} 2^m x'_{\frac{2^{j+m} \log n}{\log D}} e^{-2^{m+\log \frac{\log n}{\log D}}}$$
$$\leq \frac{2^j \log n}{\log D} \sum_{m=9}^{\log \log n} 2^m \cdot 2^{(2^{m-1}+\log e) \frac{\log n}{\log D}} B_{\boldsymbol{x}',2^{-j}} \cdot e^{-2^{m+\log \frac{\log n}{\log D}}} \ .$$

Rearranging terms, we obtain,

$$Q = \frac{2^j \log n}{\log D} B_{\boldsymbol{x}',2^{-j}} \sum_{m=9}^{\log\log n} 2^{m+(2^{m-1}+\log e)\frac{\log n}{\log D} - 2^m \frac{\log n}{\log D}}$$
$$\leq \frac{2^j \log n}{\log D} B_{\boldsymbol{x}',2^{-j}} \sum_{m=9}^{\log\log n} 2^{-2^{m-2}\frac{\log n}{\log D}} \leq \frac{2^j \log n}{\log D} B_{\boldsymbol{x}',2^{-j}} \ .$$

R is always negligible, since the $e^{-2^{i-j}}$ term is very small for large i.

$$R = \sum_{i=j+\log\log n+1}^{\log D} 2^i x'_{2^i} e^{-2^{i-j}} \leq \sum_{i=j+\log\log n+1}^{\log D} D x'_{2^i} e^{-2\log n} \leq 2Dn^{1-2\log e} \leq 1 \ .$$

So,

$$S_{\boldsymbol{x}',2^{-j}} = \frac{P+Q+R}{B_{\boldsymbol{x}',2^{-j}}} \leq \frac{256\frac{2^j \log n}{\log D} B_{\boldsymbol{x}',2^{-j}} + \frac{2^j \log n}{\log D} B_{\boldsymbol{x}',2^{-j}} + 1}{B_{\boldsymbol{x}',2^{-j}}} \leq 258 \frac{2^j \log n}{\log D} \ . \square$$

Finally, we can show that there are many j for which the condition of Lemma 97 holds. The intuition here is that the condition only fails for a region in which the network is rapidly expanding, and since D and n are already fixed it cannot be rapidly expanding everywhere.

Lemma 98. *The number of integers j, $0.01\log D \leq j \leq 0.1\log D$, for which there is $i \geq 8$ satisfying $\sum_{\ell=j+\log\frac{\log n}{\log D}}^{j+\log\frac{\log n}{\log D}+i} k_\ell > 2^i \frac{\log n}{\log D}$ is upper bounded by $0.04\log D$.*

Proof. Consider the following process: take integers i with $0.01\log D \leq i \leq 0.1\log D$ in increasing order. If there is some $i' \geq i+8$ such that $\sum_{\ell=i}^{i'} k_\ell > 2^{i'-i}\frac{\log n}{\log D}$, then call all values between i and the largest such i' 'bad', and continue the process from $i'+1$. Let b denote the number of bad i. The average k_i over all bad i must be at least $\frac{2^8 \log n}{9\log D}$, and since all k_i are bounded below by -1 and sum to at most $\log n$, we have

$$\frac{2^8 \log n}{9\log D} b + (-1)(0.09\log D - b) \leq \log n \ ,$$

and so

$$b \leq \frac{\log n + 0.09\log D}{\frac{2^8\log n}{9\log D}+1} \leq \frac{1.09\log n}{\frac{2^8\log n}{9\log D}} \leq 0.04\log D \ .$$

For every j satisfying the condition of the lemma, $j + \log \frac{\log n}{\log D}$ must be bad, and so the number of such j is also at most $0.04 \log D$. □

We are now ready to prove our main result, Theorem 81.

Proof of Theorem 81. With probability at least $1 - \frac{0.04}{0.1-0.01} \geq 0.55$, for all $i \geq 8$ we have that

$$\sum_{\ell=j+\log \frac{\log n}{\log D}}^{j+\log \frac{\log n}{\log D}+i} k_\ell \leq 2^i \frac{\log n}{\log D} \ .$$

Then, $S_{\boldsymbol{x}',2^{-j}} = O(\frac{2^j \log n}{\log D})$ by Lemmas 97 and 98. Applying Lemmas 93 and 95, we get $S_{\boldsymbol{x},2^{-j}} = O(\frac{2^j \log n}{\log D})$. Finally, applying Lemma 92, we find that the expected distance from v to its cluster center is at most $O(\frac{2^j \log n}{\log D})$. This completes the proof of Theorem 81. □

7.7 Discussion and Open Problems

As we have seen throughout this thesis, the tasks of broadcasting and leader election in radio networks are longstanding, fundamental problems in distributed computing. Our contribution in this chapter is a new algorithm for these problems that improve running times for both to $O(D\frac{\log n}{\log D} + \log^{O(1)} n)$, and succeeds with high probability. When $D = \Omega(\log^c n)$ for a sufficiently large constant c, these running time bounds improve the fastest previous algorithms for broadcasting and leader election by factors $O(\log \log n)$ and $O(\log n \log \log n)$, respectively. More importantly, whenever n is polynomial in D, i.e. $n = D^{O(1)}$, the obtained running time is $O(D)$, which is asymptotically optimal since time D is required for any information to traverse the network.

There is no better lower bound than $\Omega(D + \log^2 n)$ for broadcasting or leader election when spontaneous transmissions are allowed, so the most immediate open question is to close that gap. While a tighter analysis of our method might trim the additive polylog(n) term significantly, it is difficult to see how the $\Omega(\log^2 n)$ term could be reached without a radically different approach. Similarly, the $\Theta(D\frac{\log n}{\log D})$ term seems to be a limit of the clustering approach, and reducing it to D would likely require significant changes. In fact, we would not be surprised if our upper bound $O(D\frac{\log n}{\log D})$ were tight for $D = \Omega(\log^c n)$ for a sufficiently large constant c.

An interesting question is whether spontaneous transmissions can help in *directed* networks, which would be very surprising, or for *deterministic* protocols.

Bibliography

[1] Y. Afek, N. Alon, Z. Bar-Joseph, A. Cornejo, B. Haeupler, and F. Kuhn. Beeping a maximal independent set. In *Proceedings of the 25th International Symposium on Distributed Computing (DISC)*, pages 32–50, 2011.

[2] 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.

[3] R. Bar-Yehuda, O. Goldreich, and A. Itai. Efficient emulation of single-hop radio network with collision detection on multi-hop radio network with no collision detection. *Distributed Computing*, 5(2):67–71, 1991.

[4] 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.

[5] I. Chlamtac and S. Kutten. On broadcasting in radio networks - problem analysis and protocol design. *IEEE Transactions on Communications*, 33(12):1240–1246, 1985.

[6] 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.

[7] 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.

[8] B. Chlebus and D. R. Kowalski. Almost optimal explicit selectors. In *Proceedings of the 15th International Symposium on Fundamentals of Computation Theory (FCT)*, pages 270–280, 2005.

[9] B. S. Chlebus, G. De Marco, and M. Talo. Naming a channel with beeps. *Fundamenta Informaticae*, 153(3):199–219, 2017.

[10] 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.

[11] 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.

[12] 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.

[13] 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.

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

[15] 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.

[16] A. Cornejo and F. Kuhn. Deploying wireless networks with beeps. In *Proceedings of the 24th International Symposium on Distributed Computing (DISC)*, pages 148–262, 2010.

[17] A. Czumaj and P. Davies. Communicating with beeps. In *Proceedings of the 19th International Conference on Principles of Distributed Systems (OPODIS)*, pages 1–16, 2015.

[18] A. Czumaj and P. Davies. Brief announcement: Optimal leader election in multi-hop radio networks. In *Proceedings of the 35th Annual ACM Symposium on Principles of Distributed Computing (PODC)*, pages 47–49, 2016.

[19] A. Czumaj and P. Davies. Exploiting spontaneous transmissions for broadcasting and leader election in radio networks. In *Proceedings of the 36th Annual ACM Symposium on Principles of Distributed Computing (PODC)*, pages 3–12, 2017.

[20] A. Czumaj and P. Davies. Brief announcement: Randomized blind radio networks. In *Proceedings of the 32nd International Symposium on Distributed Computing (DISC)*, 2018.

[21] A. Czumaj and P. Davies. Deterministic blind radio networks. In *Proceedings of the 32nd International Symposium on Distributed Computing (DISC)*, 2018.

[22] A. Czumaj and P. Davies. Deterministic communication in radio networks. *SIAM Journal on Computing*, 47(1):218–240, 2018.

[23] A. Czumaj and W. Rytter. Broadcasting algorithms in radio networks with unknown topology. *Journal of Algorithms*, 60(2):115–143, 2006.

[24] F. Dufoulon, J. Burman, and J. Beauquier. Brief announcement: Beeping a time-optimal leader election. In *Proceedings of the 37th Annual ACM Symposium on Principles of Distributed Computing (PODC)*, 2018.

[25] M. Elkin and G. Kortsarz. Polylogarithmic additive inapproximability of the radio broadcast problem. *SIAM Journal on Discrete Mathematics*, 19(4):881–899, 2005.

[26] K.-T. Förster, J. Seidel, and R. Wattenhofer. Deterministic leader election in multi-hop beeping networks. In *Proceedings of the 28th International Symposium on Distributed Computing (DISC)*, pages 212–226, 2014.

[27] 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.

[28] 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.

[29] M. Ghaffari, N. Lynch, and S. Sastry. Leader election using loneliness detection. In *Proceedings of the 25th International Symposium on Distributed Computing (DISC)*, pages 268–282, 2011.

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

[31] L. Gąsieniec, A. Pelc, and D. Peleg. The wakeup problem in synchronous broadcast systems. *SIAM Journal on Discrete Mathematics*, 14(2):207–222, 2001.

[32] L. Gąsieniec, D. Peleg, and Q. Xin. Faster communication in known topology radio networks. In *Proceedings of the 24th Annual ACM Symposium on Principles of Distributed Computing (PODC)*, pages 129–137, 2005.

[33] L Gąsieniec, T Radzik, and Q Xin. Faster deterministic gossiping in directed ad hoc radio networks. In *Algorithm Theory - SWAT 2004*, pages 397–407, 2004.

[34] B. Haeupler and D. Wajc. A faster distributed radio broadcast primitive. In *Proceedings of the 35th Annual ACM Symposium on Principles of Distributed Computing (PODC)*, pages 361–370, 2016.

[35] K. Hounkanli, A. Miller, and A. Pelc. Global synchronization and consensus using beeps in a fault-prone mac. In *Proceedings of the International Symposium on Algorithms and Experiments for Sensor Systems, Wireless Networks and Distributed Robotics (ALGOSENSORS)*, pages 16–28, 2016.

[36] K. Hounkanli and A. Pelc. Deterministic broadcasting and gossiping with beeps. CoRR abs/1508.06460, 2015.

[37] K. Hounkanli and A. Pelc. Asynchronous broadcasting with bivalent beeps. In *Proceedings of the 23rd International Colloquium on Structural Information and Communication Complexity (SIROCCO)*, pages 291–306, 2016.

[38] 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.

[39] T. Jurdziński and G. Stachowiak. Probabilistic algorithms for the wakeup problem in single-hop radio networks. *Theory of Computing Systems*, 38(3):347–367, 2005.

[40] M Khabbazian and D Kowalski. Time-efficient randomized multiple-message broadcast in radio networks. In *Proceedings of the 30th Annual ACM Symposium on Principles of Distributed Computing (PODC)*, pages 373–380, 2011.

[41] A Kowalski, D.and Pelc. Optimal deterministic broadcasting in known topology radio networks. *Distributed Computing*, 19(3):185–195, Jan 2007.

[42] 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.

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

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

[45] D. Kowalski and A. Pelc. Leader election in ad hoc radio networks: A keen ear helps. *Journal of Computer and System Sciences*, 79(7):1164–1180, 2013.

[46] 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.

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

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

[49] G. De Marco, M. Pelegrini, and G. Sburlati. Faster deterministic wakeup in multiple access channels. *Discrete Applied Mathematics*, 155(8):898–903, 2007.

[50] G. Miller, R. Peng, and S. Xu. Parallel graph decompositions using random shifts. In *Proceedings of the 25th Annual ACM Symposium on Parallelism in Algorithms and Architectures (SPAA)*, pages 196–203, New York, NY, 2013. ACM Press.

[51] K. Nakano and S. Olariu. Uniform leader election protocols for radio networks. *IEEE Transactions on Parallel and Distributed Systems*, 13(5):516–526, 2002.

[52] Calvin Newport. Radio network lower bounds made easy. In *Distributed Computing*, pages 258–272, 2014.

[53] 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.

[54] B. Walke. *Mobile radio networks*, volume 2. John Wiley & Sons, 1999.

[55] D. E. Willard. Log-logarithmic selection resolution protocols in a multiple access channel. *SIAM Journal on Computing*, 15(2):468–477, 1986.

[56] J. Yu, L. Jia, D. Yu, G.s Li, and X. Cheng. Minimum connected dominating set construction in wireless networks under the beeping model. In *2015 IEEE Conference on Computer Communications (INFOCOM)*, pages 972–980. IEEE, 2015.