# Exploiting Spontaneous Transmissions for Broadcasting and Leader Election in Radio Networks*

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

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

## ABSTRACT

We study two fundamental communication primitives: *broadcasting* and *leader election* in the classical model of multi-hop radio networks with unknown topology and without collision detection mechanisms. It has been known for almost 20 years that in undirected networks with $n$ nodes and diameter $D$, randomized broadcasting requires $\Omega(D \log \frac{n}{D} + \log^2 n)$ rounds in expectation, assuming that uninformed nodes are not allowed to communicate (until they are informed). Only very recently, Haeupler and Wajc (PODC'2016) showed that this bound can be slightly improved for the model with spontaneous transmissions, providing an $O(D\frac{\log n \log\log n}{\log D} + \log^{O(1)} n)$-time broadcasting algorithm. In this paper, we give a new and faster algorithm that completes broadcasting in $O(D\frac{\log n}{\log D} + \log^{O(1)} n)$ time, with high probability. This yields the first optimal $O(D)$-time broadcasting algorithm whenever $D$ is polynomial in $n$.

Furthermore, our approach can be applied to design a new leader election algorithm that matches the performance of our broadcasting algorithm. Previously, all fast randomized leader election algorithms have been using broadcasting as their subroutine and their complexity have been asymptotically strictly bigger than the complexity of broadcasting. In particular, the fastest previously known randomized leader election algorithm of Ghaffari and Haeupler (SODA'2013) requires $O(D \log \frac{n}{D} \min\{\log\log n, \log \frac{n}{D}\} + \log^{O(1)} n)$-time with high probability. Our new algorithm requires $O(D\frac{\log n}{\log D} + \log^{O(1)} n)$ time with high probability, and it achieves the optimal $O(D)$ time whenever $D$ is polynomial in $n$.

## CCS CONCEPTS

• **Theory of computation** → **Distributed algorithms**; • **Networks** → *Network algorithms*;

*Research partially supported by the Centre for Discrete Mathematics and its Applications (DIMAP) and by EPSRC award EP/N011163/1.


*PODC '17, July 25-27, 2017, Washington, DC, USA*

ACM ISBN 978-1-4503-4992-5/17/07…$15.00
http://dx.doi.org/10.1145/3087801.3087825

## KEYWORDS

Broadcasting; Leader Election; Radio Networks

## 1 INTRODUCTION

### 1.1 Model of communication networks

We consider the classical model of *ad-hoc radio networks* with *unknown structure*. A *radio network* is modeled by an *undirected* network $\mathfrak{N} = (V, E)$, where the set of nodes corresponds to the set of transmitter-receiver stations. An edge $\{v, u\} \in E$ means that node $v$ can send a message directly to node $u$ and vice versa. To make propagation of information feasible, we assume that $\mathfrak{N}$ is connected.

In accordance with the standard model of unknown (ad-hoc) radio networks (for more elaborate discussion about the model, see, e.g., [1, 3, 4, 6, 7, 11–13, 15, 17, 19]), we make the assumption that a node does not have any prior knowledge about the topology of the network, its in-degree and out-degree, or the set of its neighbors. We assume that the only knowledge of each node is the *size* of the network $n$ and the *diameter* of the network $D$. For our algorithms, knowledge of only a linear upper bound for these values will suffice.

Nodes operate in discrete, synchronous time steps. When we refer to the "running time" of an algorithm, we mean the number of time steps which elapse before completion (i.e., we are not concerned with the number of calculations nodes perform within time steps). In each time step a node can either *transmit* a message to all of its out-neighbors at once or can remain silent and *listen* to the messages from its in-neighbors. We do not make any restriction on the size of messages, though the algorithms we present can easily be made to operate under the condition of $O(\log n)$-bit transmissions.

A further important feature of the model considered in this paper is that it allows *spontaneous transmissions*, that is, any node can transmit if it so wishes. In some prior works (see, e.g., [4, 10, 17]), it has been assumed (typically for the broadcasting problem) that uninformed nodes are not allowed to communicate (until they are informed). While this assumption is natural for the broadcasting problem, it is meaningless for the leader election problem, and so, throughout the paper we will allow spontaneous transmissions.

The distinguishing feature of radio networks is the interfering behavior of transmissions. In the most standard radio networks model, the *model without collision detection* (see, e.g., [1, 3, 7, 19]), which is studied in this paper, if a node $v$ listens in a given round and precisely one of its in-neighbors transmits, then $v$ receives the message. In all other cases $v$ receives nothing; in particular, the lack of collision detection means that $v$ is unable to distinguish between zero of its in-neighbors transmitting and more than one.

The model without collision detection describes the most restrictive interfering behavior of transmissions; also considered in the literature is a less restrictive variant, the model with collision detection, where a node listening can distinguish between zero of its in-neighbors transmitting and more than one (cf. [12, 19]).

## 1.2 Key communications primitives: Broadcasting, leader election, and Compete

In this paper we consider two fundamental communications primitives: *broadcasting* and *leader election*. We present randomized algorithms that perform these tasks *with high probability* (i.e., $1 - n^{-c}$ for an arbitrary constant $c$), and analyze worst-case running time.

*Broadcasting* is one of the most fundamental problems in communication networks and has been extensively studied for many decades (see, e.g., [19] and the references therein). The premise of the broadcasting task is that one particular node, called the *source*, has a message which must become known to all other nodes. As such, broadcasting is one of the most basic means of global communication in a network.

*Leader Election* is another most fundamental problems in communication networks that aims to ensure that all nodes agree on such a 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 basic 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 source (for more examples, cf. [5, 11], and the references therein).

To design efficient algorithms for broadcasting and leader election, we will be studying an auxiliary problem that we call Compete. 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 $\mathfrak{R}$ 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 $\Omega(\log n)$) of candidate leaders, and then perform Compete using this set, with $ID$s as the messages to be propagated. Therefore, to design an efficient randomized broadcasting and leader election algorithms, it is sufficient to design a fast randomized algorithm for Compete (cf. Section 5).

## 1.3 Previous work

As a fundamental communications primitive, the task of *broadcasting* has been extensively studied for various network models, see, e.g., [19] and the references therein.

For the model studied in this paper, undirected radio networks with unknown structure and without collision detection, the first non-trivial major result was due to Bar-Yehuda et al. [3], who, in a seminal paper, designed an almost optimal randomized broadcasting algorithm achieving the running time of $O((D + \log n) \cdot \log n)$ with high probability. This bound was later improved by Czumaj and Rytter [10], and independently Kowalski and Pelc [16], who gave randomized broadcasting algorithms that complete the task in $O(D \log \frac{n}{D} + \log^2 n)$ time with high probability. Importantly, all these algorithms were assuming that nodes are not allowed to transmit spontaneously, i.e., they must wait to receive the source message before they can begin to participate. Indeed, for the model with no spontaneous transmissions allowed, it has been known that any randomized broadcasting algorithm requires $\Omega(D \log \frac{n}{D} + \log^2 n)$ time [1, 17]. Only very recently, Haeupler and Wajc [13] demonstrated that allowing spontaneous transmissions can lead to faster broadcasting algorithms, by designing a randomized algorithm that completes broadcasting in $O(D \frac{\log n \log\log n}{\log D} + \log^{O(1)} n)$ time, with high probability. This is the only algorithm (that we are aware of) that beats the lower bound of $\Omega(D \log \frac{n}{D} + \log^2 n)$ [1, 17] in the model with no spontaneous transmissions. Given that for the model that allows spontaneous transmissions any broadcasting algorithm requires $\Omega(D + \log^2)$ time (cf. [1, 19]), the algorithm due to Haeupler and Wajc [13] is *almost optimal* (up to an $O(\log\log n)$ factor) whenever $D$ is polynomial in $n$.

Broadcasting has been also studied in various related models, including directed networks, deterministic broadcasting protocols, models with collision detection, and models in which the entire network structure is known. For example, in the model with collision detection, an $O(D + \log^6 n)$-time randomized algorithm due to Ghaffari et al. [12] is the first to exploit collisions and surpass the algorithms for broadcasting without collision detection. For deterministic protocols, the best results are an $O(n \log D \log\log D)$-time algorithm in directed networks [9], and an $O(n \log D)$-time algorithm in undirected networks [14].

For more details about broadcasting in various model, see, e.g., [19] and the references therein.

The problem of *leader election* has also been extensively studied in the distributed computing community for several decades. For the model considered in this paper, it is known that a simple reduction (see, e.g., [2]), involving performing a network-wide binary search for the highest ID using broadcasting as a subroutine every step, requires $O(T_{BC} \log n)$ time. Here $T_{BC}$ is time taken to perform broadcasting (provided the broadcasting algorithm used can be extended to work from multiple sources). This yields leader election randomized algorithms taking time $O(D \log \frac{n}{D} \log n + \log^3 n)$ using the broadcasting algorithms of [10, 16], or $O(D \frac{\log^2 n \log\log n}{\log D} + \log^{O(1)} n)$ using the broadcasting algorithm of [13]. This approach has been improved only very recently by Ghaffari and Haeupler [11], who took a more complex approach to achieve an $O(D \log \frac{n}{D} + \log^3 n) \cdot \min\{\log\log n, \log \frac{n}{D}\}$ time algorithm based on growing clusters within the network. Notice that in the regime of large $D$ being polynomial in $n$, when $D \approx n^c$ for a constant $c$, $0 < c < 1$, the fastest leader election algorithm achieves the (high probability) running time of $O(D \log n \log\log n)$.

Leader election has also been studied in various related settings. For example, one can achieve $O(T_{BC})$ expected (rather than worst

case) running time [8], or time $O(T_{BC}\sqrt{\log n})$ with high probability even for directed networks [8], and deterministically time $O(n \log n \log D \log\log D)$ [9] or $O(n \log^{3/2} n\sqrt{\log\log n})$ [5].

### 1.4 New results

In this paper we extend the approach recently developed by Haeupler and Wajc [13] to design a fast randomized algorithm for Compete, running in time $O(D\frac{\log n}{\log D} + |S|D^{0.125} + \log^{O(1)} n)$, with high probability (Theorem 4.1). By applying this algorithm to the broadcasting problem (Theorem 5.1) and to the leader election problem (Theorem 5.2), we obtain randomized algorithms for both these problems running in time $O(D\frac{\log n}{\log D} + \log^{O(1)} n)$, with high probability. For $D = \Omega(\log^c n)$ 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 $D$ is polynomial in $n$ (i.e., $D = \Omega(n^\varepsilon)$, for some positive constant $\varepsilon$), this running time is $O(D)$, which is asymptotically optimal since time $D$ is required for any information to traverse the network.

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

Finally, even though the current lower bounds for the randomized broadcasting and leader election problems are $\Omega(D+\log^2 n)$, we would not be surprised if our upper bounds $O(D\frac{\log n}{\log D} + \log^{O(1)} n)$ were tight for $D = \Omega(\log^c n)$ for some sufficiently large constant $c$.

*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 [10, 16] should be used instead.

## 2 APPROACH

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, and others (see, e.g., [11, 13]). 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.

### 2.1 Clusterings, Partition, and schedulings

To implement this approach efficiently, we follow a similar line to that of Haeupler and Wajc [13] and rely on a clustering procedure of Miller et al. [18], adapted for the radio network model.

Lemma 2.1 (Lemma 3.1 of [13]). *Let $0 < \beta \leq 1$. Any network on $n$ nodes can be partitioned into clusters each with strong diameter $O(\frac{\log n}{\beta})$ with high probability, and every edge cut by this partition with probability $O(\beta)$. This algorithm can be implemented in the radio network setting in $O(\frac{\log^3 n}{\beta})$ rounds.*

The network being partitioned into clusters means that each node identifies one particular node as its *cluster center*, the subgraph of nodes identifying any particular node as their cluster center is connected, and any node which is a cluster center to anyone must be cluster center to itself. Here the term *"strong diameter"* refers to diameter using only edges within the relevant cluster.

The clustering provided by the application of Lemma 2.1 will be denoted by Partition($\beta$).

This framework will be used in our central result, Theorem 2.2, which states that upon applying Partition($\beta$) with $\beta$ 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 2.2. *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($\beta$) is $O(\frac{\log n}{\beta \log D})$.*

We will sketch the very technical proof of this result in Appendix, in Section A; for a complete proof of Theorem 2.2, we refer to the full version of this paper[1]. This result applies to the clustering method in any setting, not just radio networks, and hence may well be of independent interest. It improves over the result of [13] 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 [12] using the notion of *schedules*.

Lemma 2.3 (Lemma 2.1 of [13]). *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 $\ell$ 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 2.3. We note that, as shown in Lemma 4.2 of [13], 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.

### 2.2 Algorithm structure

The general approach then proceeds as follows: first there is a preprocessing phase, in which we partition the network using Partition($\beta$) from Lemma 2.1, and compute schedules within the clusters using Lemma 2.3. 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 $O(|p|\frac{\log n}{\log D}) = O(D\frac{\log n}{\log D})$.

[1] Available at https://arxiv.org/abs/1703.01859.

Of course, this omits many of the technical details, and we encounter several difficulties when trying to implement the approach. Firstly, Theorem 2.2 only bounds expected distance to cluster center with constant probability. However, by generating many different clusterings, with different random values of $\beta$, 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 $\beta$ must somehow be coordinated, which we solve by using an extra layer of "coarse" clusters, similarly to [13]. 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.

## 2.3 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 [13]. Here we summarize our main technical differences from that paper and other related works:

- It was known from [13] that when Partition($\beta$) 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 2.2, 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 more difficult.

# 3 ALGORITHM FOR COMPETE

Since our broadcasting and leader election protocols require the same asymptotic running time and use similar methods (cf. Section 5), we can combine their workings into a single generalized procedure Compete.

Compete takes as input a source set $S$, 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 4.1), 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 $|S| = O(D^{0.875})$.

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 1** Compete($S$)

1) Compute a *coarse clustering* using Partition($\beta$) with $\beta = \frac{1}{\sqrt{D}}$.
2) Compute a schedule within each coarse cluster.
3) Within each coarse cluster, for each integer $j$, $0.01 \log D \leq j \leq 0.1 \log D$, compute $D^{0.2}$ different *fine clusterings* using Partition($\beta$) with $\beta = 2^{-j}$.
4) Compute schedules within all fine clusterings.
5) Each coarse cluster center computes a $D^{0.99}$-length sequence of randomly chosen fine clusterings to use.
6) Transmit this sequence within each coarse cluster, using the coarse cluster schedules.
7) For each fine clustering in the sequence perform Intra-Cluster Propagation($O(\frac{\log n}{\beta \log D})$) (with the value of $\beta$ corresponding to the fine 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 clusterings*, i.e., sub-clusterings with smaller clusters. These are the clusterings we will use to propagate information through the network. The coarse clusters generate and transmit a random sequence of these fine clusterings, which tells their members in what order to use the fine clusterings for this propagation (this was the sole purpose of the coarse clustering). We show that, when applying Intra-Cluster Propagation($O(\frac{\log n}{\beta \log D})$) on a clustering with $\beta$ chosen at random, 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* (Algorithm 2), e.g., by performing the main process during even time-steps and the background process during odd time-steps.

**Algorithm 2** Compete($S$) - Background Process

1) Compute $D^{0.2}$ different *fine clusterings* using Partition($\beta$) 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

clusterings. This means that we do not have the shared randomness we use in the main process, so we cannot choose $\beta$ randomly (we instead fix $\beta = D^{-0.1}$) and we cannot use a random ordering of fine 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).

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 makes use of the computed clusters and schedule to propagate information. Specifically, the procedure facilitates communication between the cluster center and nodes within $\ell$ hops.

**Algorithm 3** Intra-Cluster Propagation($\ell$)

1) Broadcast the highest message known by the cluster center to all nodes within $\ell$ distance.
2) All such nodes which know a higher message participate in a broadcast towards the cluster center.
3) Broadcast the highest message known by the cluster center to all nodes within $\ell$ distance.

Here we apply Lemma 2.3: after computing schedules, it is possible to broadcast between the cluster center and nodes at distance at most $\ell$ in time $O(\ell + \log^{O(1)} n)$. That is, on an outward broadcast all nodes within distance $\ell$ 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 clusters at the same time, we will describe a background process (Algorithm 4) to deal with collisions between fine 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 4** Intra-Cluster Propagation Background Process

Repeat until main process is complete:
**for** $i = 1$ to $\log n$ **do**
  with probability $2^{-i}$ (coordinated in each cluster) 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 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 5), which ensures that the node will then hear its cluster's message (with constant probability).

The Decay protocol, first introduced by Bar-Yehuda et al. [3], is a fundamental transmission primitive employed by many randomized radio network communication algorithms.

**Algorithm 5** Decay at a node $v$

**for** $i = 1$ **to** $\log n$, in time step $i$ **do**:
  $v$ transmits its message with probability $2^{-i}$.

Lemma 3.1 ([3]). *After a single round of Decay, a node $v$ with at least one participating in-neighbor receives a message with constant probability.* □

## 4 ANALYSIS OF COMPETE ALGORITHM

In this section we prove the following guarantee on the behavior of Compete:

Theorem 4.1. *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 $O(D^{0.99} \log^{O(1)} n) = O(D)$ time, and upon its completion we have all the schedules required to perform Intra-Cluster Propagation. As in [13], 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 clusters.

Lemma 4.2. *For some constant $c$, upon applying Intra-Cluster Propagation($\ell$) 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 who border another fine cluster *risky*. We make use of a result of [13] (a corollary of Lemma 3.6 used during 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 clusters, and consider how long $v$ must wait to be informed if it has a neighbor in its own cluster who 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 [13], 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.

To show that there are not too many bad subpaths, we make use of the following result from [13]:

Lemma 4.3 (Corollary 3.8 of [13]). *After running Partition($\beta$) 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

$$1 - e^{-D^{-0.5}(2D^{0.11}+1)} \leq 1 - e^{-3D^{-0.39}} \leq 3D^{-0.39} \quad .$$

Taking the union bound over all nodes in a path, we find that any length-$D^{0.12}$ path is bad with probability upper bounded by $D^{0.12} \cdot 3D^{-0.39} \leq D^{-0.26}$.

Lemma 4.4. *All shortest paths $p$ between two vertices 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 [13], we first condition on the event that all exponentially distributed random variables $\delta_v$ used when computing the coarse clustering are at most $2D^{0.5} \log n$, which is the case with high probability (for details of how the clustering algorithm works see [13, 18]). 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 $\delta_v$. 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 order out of all messages of nodes in $S$. That is, by *message $j$* we mean the $j^{th}$ highest message in $S$.

Lemma 4.5 (***Bad subpaths***). *Let $p$ be any $(u, v)$-path 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 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 2.1 the first event occurs with high probability, by Corollary 3.7 of [13], we can make the probability of the second event an arbitrarily high constant by our choice of $c$, and by Lemma 4.2 and the 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 the 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 4.6 (***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 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 2.2, 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 [13], 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 4.2 the probability that $x$ and $w$ are valid, conditioned on the first event, is at least 0.98. Therefore each fine 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 $\beta_s^{-1}$ 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,

$$\mathbb{P}\left[\sum_{s\in S} x_s \leq \frac{c}{6}D^{0.12}\right] \leq e^{-\frac{2|S|^2(\frac{c}{6}D^{0.12})^2}{\sum_{s\in S}\beta_s^{-2}}} \leq e^{-\frac{2|S|(\frac{c}{6}D^{0.12})^2}{D^{0.1}}} \leq e^{-\log^2 n}.$$

By time $t'$, $w$ has advanced at least $\frac{\sum_{s\in S}}{c_2} \geq \frac{cD^{0.12}}{6c_2}$ 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 4.4–4.6 to show how to propagate messages along any shortest path between two nodes.

Lemma 4.7 (***All shortest paths***). *Let $u$ and $v$ be any nodes in $\Re$, $p$ be some 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 after $t + 2k(\frac{|p|\log n}{\log D} + (2i+b)D^{0.125})$ steps, $v$ knows a message at least as high as $i$ with high probability.*

Proof. We prove the lemma using double induction. Our 'outer' induction shall be on the value $i$.

**Base case:** $i = 1$. By Lemma 4.4, $p$ contains at most $D^{0.63}$ bad subpaths. Applying Lemmas 4.5 and 4.6, the time taken to inform $v$ of a message at least as high as 1 is at most $D^{0.63}\cdot O(D^{0.121}) + D^{0.88}\cdot O(D^{0.12}\frac{\log n}{\log D}) \leq kD\frac{\log n}{\log D}$.

**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 $|p|$.

**Base case:** $|p| \leq D^{0.12}$. $p$ 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 $t + k(\frac{(|p|+D^{0.11})\log n}{\log D} + (2(\ell - 1) + 1)D^{0.125})$. Then, by Lemma 4.6, $v$ knows an ID at least as high as $\ell$ by time-step

$$t + k\left(\frac{(|p| + D^{0.11})\log n}{\log D} + (2\ell - 1)D^{0.125}\right) + cD^{0.12}\frac{\log n}{\log D} \leq t + k\left(\frac{|p|\log n}{\log D} + 2\ell D^{0.125}\right).$$

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 $t + k(\frac{(|p|+D^{0.11})\log n}{\log D} + (2(\ell - 1) + 2)D^{0.125})$. Then, by Lemma 4.5, $v$ knows an ID at least as high as $i$ by time-step

$$t + k\left(\frac{(|p| + D^{0.11})\log n}{\log D} + 2\ell D^{0.125}\right) + cD^{0.121} \leq t + k\left(\frac{|p|\log n}{\log D} + (2\ell + 1)D^{0.125}\right).$$

**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 $\ell$ by time-step $t + k(\frac{(|p|-D^{0.12})\log n}{\log D} + (2i+b)D^{0.125})$. 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 $t + k((|p| + D^{0.11})\log n \log D + (2(\ell-1) + (b+1)))D^{0.125} \leq t + k(\frac{(|p|-D^{0.12})\log n}{\log D} + (2\ell + b))D^{0.125}$ Therefore, by Lemma 4.6, $v$ knows a message at least as high as $\ell$ by time-step

$$t + k\left(\frac{(|p| - D^{0.12})\log n}{\log D} + (2\ell + b)D^{0.125}\right) + cD^{0.12}\frac{\log n}{\log D} \leq t + k\left(\frac{|p|\log n}{\log D} + (2\ell + b)D^{0.125}\frac{\log n}{\log D}\right).$$

If the subpath is bad, then by inductive assumption (2), $u'$ knows an ID at least as high as $\ell$ by time-step $t + k(\frac{(|p|-D^{0.12})\log n}{\log D} + (2\ell + b - 1)D^{0.125}) \leq t + k(\frac{(|p|+D^{0.11})\log n}{\log D} + (2(\ell-1)+b)D^{0.125})$. 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 $t + k(\frac{(|p|+D^{0.11})\log n}{\log D} + (2(\ell-1)+b)D^{0.125})$ Therefore, by Lemma 4.5, $v$ knows a message at least as high as $\ell$ by time-step

$$t + k\left(\frac{(|p| + D^{0.11})\log n}{\log D} + (2(\ell - 1) + b)D^{0.125}\right) + cD^{0.121} \leq t + k\left(\frac{|p|\log n}{\log D} + (2\ell + b)D^{0.125}\frac{\log n}{\log D}\right).$$

This completes the proof by induction. □

We are now ready to prove Theorem 4.1:

Proof of Theorem 4.1. The precomputation phase takes at most $O(D + \log^{O(1)})$ time. Upon beginning the Intra-Cluster Propagation phase, one node $u$ knows the highest message. Therefore by Lemma 4.7, all nodes $v$ know this message within $2k(\frac{|dist(u,v)|\log n}{\log D} + (2|S|+b)D^{0.125}) = O(\frac{D\log n}{\log D} + |S|D^{0.125} + \log^{O(1)} n)$ time-steps, with high probability. □

## 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 5.1. *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 6** Leader Election

Nodes choose to become candidates in $C$ with probability $\frac{100\log n}{n}$.
Candidates randomly generate $\Theta(\log n)$-bit IDs.
Perform Compete($C$).

Theorem 5.2. *Algorithm 6 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. □

## 6 CONCLUSIONS

The tasks of broadcasting and leader election in radio networks are longstanding, fundamental problems in distributed computing. Our main contribution are new algorithms for these problems that improve running times for both to $O(D\frac{\log n}{\log D}+\log^{O(1)} n)$, with high probability. For $D = \Omega(\log^c n)$ 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 $D$ is polynomial in $n$ (i.e., $D = \Omega(n^\varepsilon)$, for some positive constant $\varepsilon$), 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 $\log^2 n$ could be reached without a radically different approach. Similarly, the $D\frac{\log n}{\log D}$ 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 bounds $O(D\frac{\log n}{\log D})$ were tight for $D = \Omega(\log^c n)$ for a sufficiently large constant $c$.

The main focus of this paper has been to study the impact of spontaneous transmissions for basic communication primitives in randomized algorithms undirected networks. An interesting question is whether spontaneous transmissions can help in *directed* networks, which would be very surprising, or for *deterministic* protocols.

## REFERENCES


[1] Noga Alon, Amotz Bar-Noy, Nathan Linial, and David Peleg. 1991. A Lower Bound for Radio Broadcast. *Journal of Computer and System Sciences* 43, 2 (October 1991), 290–298.

[2] Reuven Bar-Yehuda, Oded Goldreich, and Alon Itai. 1991. Efficient Emulation of Single-hop Radio Network with Collision on Multi-hop Radio Network with no Collision Detection. *Distributed Computing* 5, 1 (September 1991), 67–71.

[3] Reuven Bar-Yehuda, Oded Goldreich, and Alon Itai. 1992. 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 (August 1992), 104–126.

[4] Bogdan S. Chlebus, Leszek Gąsieniec, Alan Gibbons, Andrzej Pelc, and Wojciech Rytter. 2002. Deterministic Broadcasting in Unknown Radio Networks. *Distributed Computing* 15, 1 (January 2002), 27–38.

[5] Bogdan S. Chlebus, Dariusz R. Kowalski, and Andrzej Pelc. 2012. Electing a Leader in Multi-hop Radio Networks. In *Proceedings of the 16th International Conference on Principles of Distributed Systems (OPODIS) (Lecture Notes in Computer Science)*, Vol. 7702. Springer, Berlin, Heidelberg, 106–120.

[6] Marek Chrobak, Leszek Gąsieniec, and Wojciech Rytter. 2002. Fast Broadcasting and Gossiping in Radio Networks. *Journal of Algorithms* 43, 2 (May 2002), 177–189.

[7] Andrea E. F. Clementi, Angelo Monti, and Riccardo Silvestri. 2003. Distributed Broadcasting in Radio Networks of Unknown Topology. *Theoretical Computer Science* 302, 1-3 (April 2003), 337–364.

[8] Artur Czumaj and Peter Davies. 2016. Brief Announcement: Optimal Leader Election in Multi-Hop Radio Networks. In *Proceedings of the 35th Annual ACM Symposium on Principles of Distributed Computing (PODC)*. ACM Press, New York, NY, 47–49.

[9] Artur Czumaj and Peter Davies. 2016. Faster Deterministic Communication in Radio Networks. In *Proceedings of the 43rd Annual International Colloquium on Automata, Languages and Programming (ICALP)*. Leibniz International Proceedings in Informatics (LIPICS), Schloss Dagstuhl — Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany, 139:1–139:14.

[10] Artur Czumaj and Wojciech Rytter. 2003. Broadcasting Algorithms in Radio Networks with Unknown Topology. *Journal of Algorithms* 60, 2 (August 2003), 115–143.

[11] Mohsen Ghaffari and Bernhard Haeupler. 2013. Near Optimal Leader Election in Multi-hop Radio Networks. In *Proceedings of the 24th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*. SIAM, Philadelphia, PA, 748–766.

[12] Mohsen Ghaffari, Bernhard Haeupler, and Majid Khabbazian. 2015. Randomized Broadcast in Radio Networks with Collision Detection. *Distributed Computing* 28, 6 (December 2015), 407–422.

[13] Bernhard Haeupler and David Wajc. 2016. A Faster Distributed Radio Broadcast Primitive. In *Proceedings of the 35th Annual ACM Symposium on Principles of Distributed Computing (PODC)*. ACM Press, New York, NY, 361–370.

[14] Dariusz R. Kowalski. 2005. On Selection Problem in Radio Networks. In *Proceedings of the 24th Annual ACM Symposium on Principles of Distributed Computing (PODC)*. ACM Press, New York, NY, 158–166.

[15] Dariusz R. Kowalski and Andrzej Pelc. 2004. Faster Deterministic Broadcasting in Ad Hoc Radio Networks. *SIAM Journal on Discrete Mathematics* 18, 2 (2004), 332–346.

[16] Dariusz R. Kowalski and Andrzej Pelc. 2005. Broadcasting in Undirected ad hoc Radio Networks. *Distributed Computing* 18, 1 (July 2005), 43–57.

[17] Eyal Kushilevitz and Yishay Mansour. 1998. An $\Omega(D\log(N/D))$ Lower Bound for Broadcast in Radio Networks. *SIAM Journal on Computing* 27, 3 (1998), 702–712.

[18] Gary L. Miller, Richard Peng, and Shen Chen Xu. 2013. Parallel Graph Decompositions Using Random Shifts. In *Proceedings of the 25th Annual ACM Symposium on Parallelism in Algorithms and Architectures (SPAA)*. ACM Press, New York, NY, 196–203.

[19] David Peleg. 2007. Time-efficient Broadcasting in Radio Networks: A Review. In *Proceedings of the 4th International Conference on Distributed Computing and Internet Technology (ICDCIT) (Lecture Notes in Computer Science)*, Vol. 4882. Springer, Berlin, Heidelberg, 1–18.


## A APPENDIX: CLUSTERING PROPERTY (PROOF OF THEOREM 2.2)

In this section we provide main ideas needed to prove a key property of the clustering method in our algorithm Partition($\beta$), Theorem 2.2. Partition($\beta$) is based on a method first introduced by Miller et al. [18]. The concept is as follows: each node $v$ independently generates an exponentially distributed random variable $\delta_v$, that is, a variable taking values in $\mathbb{R}_{\geq 0}$ with $\mathbb{P}[\delta_v \leq y] = 1 - e^{-\beta y}$. Then, each node chooses its cluster center $u$ to be the node maximizing $\delta_u - dist(u, v)$. It can be seen by the triangle inequality that a node which is cluster center to any node is also cluster center to itself. For details of how to implement this in the radio network setting, see [13].

The rest of this section is now devoted to a sketch of the proof of Theorem 2.2; for a complete and very technical proof of Theorem 2.2, we refer to the full version of this paper.

## A.1 Bounding expected distance by $O(S_{x,\beta})$

Our first step in proving Theorem 2.2 is to obtain a bound for distance to 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($\beta$).

Lemma A.1. *For any fixed node $v$ and value $\beta$ with $D^{-0.01} \le \beta \le D^{-0.1}$, the expected distance from $v$ to its cluster center upon applying Partition($\beta$) 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:

$$\begin{aligned}
&\mathbb{E}[\text{distance from } v \text{ to its cluster center}] \\
&= \sum_{i=0}^{D} i \cdot \mathbb{P}\left[v\text{'s cluster center is distance } i \text{ away}\right] \\
&= \sum_{i=0}^{D} i \cdot \left( \sum_{u \in A_i(v)} \mathbb{P}\left[u \text{ is } v\text{'s cluster center}\right] \right) .
\end{aligned}$$

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 $\mathbb{P}[u \text{ is } v\text{'s cluster center}]$. We note that,

$$\mathcal{P}_{u,v} = \int_i^{\infty} \beta e^{-\beta p} \mathbb{P}\left[u \text{ is } v\text{'s cluster center} | \delta_u = p\right] dp$$

by conditioning on the value of $\delta_u$ 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 $\mathcal{P}_{u,v} = 0$).

Having conditioned on $\delta_u$, we can evaluate the probability that $u$ is $v$'s cluster center based on the random variables generated by other nodes, since the probabilities that each other node 'beats' $u$ are now independent, $\mathcal{P}_{u,v}$ is equal to:

$$\int_i^{\infty} \beta e^{-\beta p} \prod_{w \neq u} \mathbb{P}\left[\delta_w - dist(v,w) < \delta_u - dist(v,u) | \delta_u = p\right] dp \ .$$

We can simplify by grouping the nodes $w$ based on distance from $v$, though we must be careful to include a $\frac{1}{\mathbb{P}[\delta_u < p]}$ term to cancel out $u$'s contribution to the resulting product:

$$\mathcal{P}_{u,v} = \int_i^{\infty} \frac{\beta e^{-\beta p}}{\mathbb{P}\left[\delta_u < p\right]} \prod_{k=0}^{D} \prod_{w \in A_k(v)} \mathbb{P}\left[\delta_w - k < p - i\right] dp \ .$$

Plugging in the cumulative distribution function 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 $1 - y \le e^{-y}$ 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 $x_k$:

$$\begin{aligned}
\mathcal{P}_{u,v} &\le \int_i^{\infty} \frac{\beta e^{-\beta p}}{1 - e^{-\beta p}} \prod_{k=0}^{D} \prod_{w \in A_k(v)} e^{-e^{-\beta(p-i+k)}} dp \\
&= \int_i^{\infty} \frac{\beta e^{-\beta p}}{1 - e^{-\beta p}} \prod_{k=0}^{D} e^{-e^{-\beta(p-i+k)} x_k} dp \ .
\end{aligned}$$

We can also remove the remaining product by taking it as a sum into the exponent, and re-arranging some terms yields:

$$\begin{aligned}
\mathcal{P}_{u,v} &\le \int_i^{\infty} \frac{\beta e^{-\beta p}}{1 - e^{-\beta p}} e^{-e^{\beta(i-p)} \sum_{k=0}^{D} x_k e^{-\beta k}} dp \\
&= \int_i^{\infty} \frac{\beta e^{-\beta p}}{1 - e^{-\beta p}} e^{-e^{\beta(i-p)} B_{\boldsymbol{x},\beta}} dp \ ,
\end{aligned}$$

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:

$$\mathcal{P}_{u,v} \le J + K \ ,$$

where,

$$\begin{aligned}
J &= \int_i^{\frac{1}{\beta}} \frac{\beta e^{-\beta p}}{1 - e^{-\beta p}} e^{-e^{\beta(i-p)} B_{\boldsymbol{x},\beta}} dp \ , \\
K &= \int_{\frac{1}{\beta}}^{\infty} \frac{\beta e^{-\beta p}}{1 - e^{-\beta p}} e^{-e^{\beta(i-p)} B_{\boldsymbol{x},\beta}} dp \ .
\end{aligned}$$

To bound $J$, we make use of the following bound on $B_{\boldsymbol{x},\beta}$:

$$\begin{aligned}
B_{\boldsymbol{x},\beta} &= \sum_{k=0}^{D} x_k e^{-k\beta} \ge \sum_{k=0}^{\lceil \frac{D}{2} \rceil} e^{-k\beta} \ge \int_{-1}^{\frac{D}{2}} e^{-z\beta} dz \\
&= \frac{-1}{\beta}\left(e^{-\frac{\beta D}{2}} - e^{-\beta}\right) \ge \frac{1}{2\beta} \ .
\end{aligned}$$

This gives

$$J \le \int_i^{\frac{1}{\beta}} \frac{\beta e^{-\beta p}}{1 - e^{-\beta p}} e^{-e^{\beta(i-p)} \frac{1}{2\beta}} dp \ .$$

Since $e^{\beta(i-p)} \ge e^{-1}$, we obtain,

$$J \le \int_1^{\frac{1}{\beta}} \frac{\beta e^{-\beta p}}{1 - e^{-\beta p}} e^{-\frac{1}{2e\beta}} dp = \beta e^{-\frac{1}{2e\beta}} \int_1^{\frac{1}{\beta}} \frac{e^{-\beta p}}{1 - e^{-\beta p}} dp \ .$$

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 $J \le e^{-\frac{1}{2e\beta}} \log \frac{(1-e)}{(1-e^{\beta})}$. Since $e^{\beta} > 1 + \beta$, re-arranging yields $J \le e^{-\frac{1}{2e\beta}} \log \frac{e-1}{\beta}$. Finally, since we can assume that $\frac{1}{\beta} \ge \log^c n$ for some sufficiently large $c$, we obtain,

$$J \le e^{-\frac{\log^2 n}{2e}} \log \frac{e-1}{\beta} \le n^{-2} \ .$$

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 $1 - e^{-\beta p} \ge 1 - e^{-1} > \frac{1}{2}$, 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 $e^{-e^{-\beta p}} \le 1 - \frac{1}{2}e^{-\beta p}$ (since $0 \le e^{-\beta p} \le 1$), 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,

$$K < 2\frac{(e^{-1} - 2)(1 - \frac{1}{2}e^{-1})^{e^{\beta i}B_{\boldsymbol{x},\beta}} + 2}{1 + e^{\beta i}B_{\boldsymbol{x},\beta}} \le \frac{4}{e^{\beta i}B_{\boldsymbol{x},\beta}} \ .$$

We can now combine our calculations to prove the lemma. Since $x_i = |A_i(v)|$, we have,

$$\begin{aligned}
&\mathbb{E}[\text{distance from } v \text{ to its cluster center}] \\
&= \sum_{i=0}^{D} i \sum_{u \in A_i(v)} \mathcal{P}_{u,v} \le \sum_{i=0}^{D} i x_i (J + K) \\
&< \sum_{i=0}^{D} i x_i \left( n^{-2} + \frac{4}{e^{\beta i}B_{\boldsymbol{x},\beta}} \right) \le n^{-2} \sum_{i=0}^{D} D x_i + \frac{4\sum_{i=0}^{D} i x_i e^{-\beta i}}{B_{\boldsymbol{x},\beta}} \\
&\le \frac{D}{n} + 4S_{\boldsymbol{x},\beta} \le 5S_{\boldsymbol{x},\beta} \ . \qquad \square
\end{aligned}$$

## A.2 Simplifying the form of $x$ to bound $S_{x,\beta}$

By Lemma A.1, 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 increasing any $S_{\boldsymbol{x},\beta}$ by more than a constant factor.

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 A.2. *For all $\boldsymbol{x} \in \mathbb{N}_0^D$, $S_{\boldsymbol{x},\beta} \le 11 S_{f(\boldsymbol{x}),\beta}$.*

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 non-zero 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 A.2, we can bound $S_{\boldsymbol{x},\beta}$ in terms of $S_{g(\boldsymbol{x}),\beta}$.

Lemma A.3. *For all $\boldsymbol{x} \in \mathbb{N}_0^D$ which have $x_i = 0$ for all $i \notin \{2^k : k \in \mathbb{N}_0\}$, $S_{\boldsymbol{x},\beta} \le 2S_{g(\boldsymbol{x}),\beta}$.*

## A.3 Bounding $S_{x,\beta}$ for simplified $x$

Now that we have shown in Lemmas A.2 and A.3 that the transformations $f$ and $g$ do not increase $S_{\boldsymbol{x},\beta}$ 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 A.4. *$\boldsymbol{x'}$ has the following properties:*

- *$x'_i = 0$ for all $i \notin \{2^k : k \in \mathbb{N}_0\}$;*
- *$x'_1 \ge 2$;*
- *$||\boldsymbol{x'}||_1 = \sum_{i=0}^{D} x'_i \le 2n$, ;*
- *$2x'_{2i} \ge x'_i$ for all $i$, due to transformation $g$.*

Our argument will be based on examining the ratios between consecutive non-zero coefficients in $\boldsymbol{x'}$. To that end, define $k_i = \log \frac{x'_{2^{i+1}}}{x'_{2^i}}$ for all $0.01 \log D \le i \le 0.1 \log D$, and note that $k_i \ge \log \frac{1}{2} = -1$ for all $i$ and $\sum_{i=0}^{\log D} k_i \le \log n$ by Lemma A.4.

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 $\beta$:

Lemma A.5. *If for fixed $j$ and for all $m \ge 8$ we have*

$$\sum_{\ell=j+\log \frac{\log n}{\log D}}^{j+\log \frac{\log n}{\log D}+m} k_\ell \le 2^m \frac{\log n}{\log D}$$

*then $S_{\boldsymbol{x}',2^{-j}} = O(\frac{2^j \log n}{\log D})$.*

Finally, we can show that there are many $j$ for which the condition of Lemma A.5 holds.

Lemma A.6. *The number of integers $j$, $0.01 \log D \le j \le 0.1 \log D$, for which there is $i \ge 8$ 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 $0.04 \log D$.*

We are now ready to prove our main result, Theorem 2.2.

Proof of Theorem 2.2. With probability at least $1 - \frac{0.04}{0.1 - 0.01} \ge 0.55$, for all $i \ge 8$ we have that

$$\sum_{\ell=j+\log \frac{\log n}{\log D}}^{j+\log \frac{\log n}{\log D}+i} k_\ell \le 2^i \frac{\log n}{\log D} \ .$$

Then, $S_{\boldsymbol{x}',2^{-j}} = O(\frac{2^j \log n}{\log D})$ by Lemmas A.5 and A.6. Applying Lemmas A.2 and A.3, we get $S_{\boldsymbol{x},2^{-j}} = O(\frac{2^j \log n}{\log D})$. Finally, applying Lemma A.1, we find that the expected distance from $v$ to its cluster center is at most $O(\frac{2^j \log n}{\log D})$. $\square$