The Energy Complexity of BFS in Radio Networks

Yi-Jun Chang
ETH Zürich
yi-jun.chang@eth-its.ethz.ch

Varsha Dani
University of New Mexico
varshadani@gmail.com

Thomas P. Hayes*
University of New Mexico
hayest@gmail.com

Seth Pettie†
University of Michigan
seth@pettie.net

ABSTRACT

We consider a model of *energy complexity* in Radio Networks in which transmitting or listening on the channel costs one unit of energy and computation is free. This simplified model captures key aspects of battery-powered sensors: that battery-life is most influenced by transceiver usage, and that at low transmission powers, the actual cost of transmitting and listening are very similar.

The energy complexity of tasks in single-hop (clique) networks are well understood [6, 9, 20, 32]. Recent work of Chang et al. [8] considered energy complexity in *multi-hop* networks and showed that Broadcast admits an *energy-efficient* protocol, by which we mean each of the n nodes in the network spends $O(\text{polylog}(n))$ energy. This work left open the strange possibility that *all* natural problems in multi-hop networks might admit such an energy-efficient solution.

In this paper we prove that the landscape of energy complexity is rich enough to support a multitude of problem complexities. Whereas Broadcast can be solved by an energy-efficient protocol, exact computation of Diameter cannot, requiring $\Omega(n)$ energy. Our main result is that BreadthFirstSearch has sub-polynomial energy complexity at most $2^{O(\sqrt{\log n \log\log n})} = n^{o(1)}$; whether it admits an efficient $O(\text{polylog}(n))$-energy protocol is an open problem.

Our main algorithm involves recursively solving a generalized BFS problem on a "cluster graph" introduced by Miller, Peng, and Xu [31]. In this application, we make crucial use of a close relationship between distances in this cluster graph, and distances in the original network. This relationship is new and may be of independent interest.

We also consider the problem of approximating the network Diameter. From our main result, it is immediate that Diameter can be 2-approximated using $n^{o(1)}$ energy per node. We observe that, for all $\epsilon > 0$, approximating Diameter to within a $(2-\epsilon)$ factor requires $\Omega(n)$ energy per node. However, this lower bound is only due to graphs of very small diameter; for large-diameter graphs, we prove that the diameter can be nearly 3/2-approximated using $O(n^{1/2+o(1)})$ energy per node.

CCS CONCEPTS

• **Theory of computation → Distributed algorithms**.

KEYWORDS

distributed computing, graph algorithms, energy-aware computing, radio networks, sensor networks

ACM Reference Format:
Yi-Jun Chang, Varsha Dani, Thomas P. Hayes, and Seth Pettie. 2020. The Energy Complexity of BFS in Radio Networks. In *ACM Symposium on Principles of Distributed Computing (PODC '20), August 3–7, 2020, Virtual Event, Italy.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3382734.3405713

1 INTRODUCTION

Consider a network of n tiny sensors scattered throughout a National Park. We'd like the sensors to organize themselves, so that in the event of a forest fire, say, information about it can be *efficiently* broadcast to the entire network.

In this extremely low power setting, sensors would need to spend most of their time with their transceiver units shut off to conserve power. In a steady state, we might expect that we have a good *labelling* of the nodes, and each node with label i wakes up at times of the form $jP + i$, where j runs through every positive integer, and P, the polling period, is also a positive integer. Each node wakes up just long enough to receive a message and forward it on any neighbors with label $i + 1$. In this way, at the expense of adding P to the latency, the nodes are able to reduce their power consumption by a factor of P, compared to the always-on scenario.

Once P has been optimized, which should be a function of the available power, the next issue is how to find a good labelling efficiently. In this paper we focus mainly on the problem of computing *BFS labelings*: a given source s has label zero, and all other devices label themselves by the distance (in hops) to s. Such a labeling gives a 2-approximation to the diameter, and via up-casts and down-casts, allows for time- and energy-efficient dissemination of a message from any origin. Thus, the problem of finding a BFS labelling is a very natural question in this context.

1.1 The Model

We work within the classic *Radio Network* model [10], but in contrast to most prior work in this model, we treat *energy* (defined below) as the primary measure of complexity and *time* to be important, but secondary.

*Supported by NSF CAREER award CCF-1150281.
†Supported by NSF grants CCF-1514383, CCF-1637546, and CCF-1815316.

PODC '20, August 3–7, 2020, Virtual Event, Italy

ACM ISBN 978-1-4503-7582-5/20/08…$15.00
https://doi.org/10.1145/3382734.3405713

There are $|V|$ devices associated with the nodes of an *unknown* undirected graph $G = (V, E)$. *Time* is partitioned into discrete steps. All devices agree on time zero,[1] and agree on some upper bound $n \geq |V|$. In each timestep, each device performs some computation and chooses to either idle, listen to the channel, or transmit a message. If a device v chooses to listen, and *exactly* one device $u \in N(v)$ transmits a message m_u, then v receives m_u. In all other cases, v receives no feedback from the environment.[2] Devices can locally generate unbiased random bits; there is no shared randomness. Let RN[b] denote this Radio Network model, where b is the maximum number of bits per message. All of our algorithms work in RN[$O(\log n)$] and all our lower bounds apply even to RN[∞].

Cost Measures. An algorithm runs in *time* t if all devices halt and return their output by timestep t. Typically the algorithm is randomized, with some probability of failure, but t is a function of n or other given parameters, not a random variable. The *energy cost* of $v \in V$ is the number of timesteps for which v is listening or transmitting. (This is motivated by the fact that the *sleep mode* of tiny devices is so efficient that it is reasonable to approximate its energy-cost by *zero*, and that transceiver usage is often the most expensive part of a computation. Moreover, at low transmission powers, transmitting and listening are comparable; see, e.g., [34, Fig. 2] and [5, Table 1].) The energy cost of the *algorithm* is the maximum energy cost of any device.

Energy Complexity. Most prior work on energy complexity has focused on *single-hop* (clique) networks, typically under the assumption that $|V| = n$ is *unknown*, and that some type of collision-detection is available.[3] Because of the high degree of symmetry, there are only so many interesting problems in single-hop networks. Nakano and Olariu [32] proved that the Initialization problem (assign devices distinct IDs in $\{1, \ldots, |V| = n\}$) can be solved with $O(\log \log n)$ energy. Bender et al. [6] showed that with collision-detection, all n devices holding messages can transmit all of them using $O(\log(\log^* n))$ energy. Chang et al. [9] proved that $\Theta(\log(\log^* n))$ is optimal, and more generally, settled the complexity of LeaderElection and ApproximateCounting (estimating "n") in all the collision-detection models, with and without randomization. It was proved that collision-detection gives two exponential advantages in energy complexity. With randomization, LeaderElection/ApproximateCounting takes $\Theta(\log^* n)$ energy (without CD) or $\Theta(\log(\log^* n))$ energy (with CD), and deterministically, they take $\Theta(\log N)$ energy (without CD [20]) and $\Theta(\log \log N)$ energy (with CD), where devices initially have IDs in $[N]$. See also [18–22]. Three-way tradeoffs between time, energy, and error probability were studied by Chang et al. [9] and Kardas et al. [24].

[1] Synchronizing devices in an energy-efficient manner is an interesting open problem. In some situations it makes sense to assume the devices begin in a synchronized state, e.g., if the sensors are simultaneously turned on and dropped from an airplane on the aforementioned National Park.

[2] Here $N(v) = \{u \mid \{u, v\} \in E(G)\}$ is the neighborhood of v. A more powerful model allows for *collision detection*, i.e., differentiation between zero and two or more transmitters in $N(v)$. Since collision detection only gives a polylog(n) advantage in any complexity measure (Local-Broadcast in Section 2 allows each vertex to differentiate between zero and two or more transmitters in polylog(n) rounds w.h.p.) and we are insensitive to such factors, we assume the weakest model, without collision detection.

[3] Sender-side CD enables devices to detect if another device is transmitting; receiver-side CD lets receivers detect if at least two devices are transmitting.

Very recently Chang et al. [8] extended the single-hop notion of energy complexity to *multi-hop* networks (G is *not* a clique), and proved nearly sharp upper and lower bounds on Broadcast, both in RN[$O(\log n)$] and the same model when listeners have collision detection. Without CD the energy complexity of Broadcast is between $\Omega(\log^2 n)$ and $O(\log^3 n)$; with CD it is between $\Omega(\log n)$ and $O\left(\frac{\log n \log\log n}{\log\log\log n}\right)$.

Other Energy Models. Other notions of energy complexity have been studied in radio networks. For example, when distances between devices are very large, transmitting is significantly more expensive than listening, and it makes sense to design algorithms that minimize the worst-case number of transmissions per device. Gasnieniec et al. [13], Klonowski and Pajak [26], and Berenbrink et al. [7] studied broadcast and gossiping problems under this cost model. Klonowski and Sulkowska [27] defined a distributed model in which devices are scattered randomly at points in $[n^{1/d}]^d$ and can choose their transmission power dynamically. Several works have looked at energy complexity against an adversarial *jammer*, where the energy cost is sometimes a function of the adversary's energy budget. See, e.g., [15, 23, 25, 29].

Time Complexity. Most prior work in the RN model has studied the time complexity of basic primitives such as LeaderElection, Broadcast, BFS, etc. We review a few results most relevant to our work. Bar-Yehuda et al.'s [3] *decay* algorithm solves BFS in $O(D \log^2 n)$ time and Broadcast in $O(D \log n + \log^2 n)$ time. Here D is the diameter of the network. Since $\Omega(D)$ is an obvious lower bound, the question is which log-factors are necessary. Alon et al. [2] proved that the additive $\log^2 n$ term is necessary in a strong sense: even with full knowledge of the graph topology, Broadcast needs $\Omega(\log^2 n)$ time even when $D = O(1)$. Kushilevitz and Mansour [28] proved that if devices are forbidden from transmitting before hearing the message, then $\Omega(D \log(n/D))$ time in necessary. Czumaj and Davies [12] (improving [16]) gave a Broadcast algorithm running in $O(D \log_D n + \text{polylog}(n))$ time, which is optimal when $D > n^{\epsilon}$. These Broadcast algorithms *do not* solve BFS. Improving the classic $O(D \log^2 n)$ decay algorithm for BFS, Ghaffari and Haeupler [14] solve BFS in $O(D \log(n) \log\log(n) + \text{polylog}(n))$ time.

New Results. It is useful to coarsely classify energy-efficiency bounds as either *feasible* or *infeasible*. We consider polylog(n) energy to be feasible and polynomial energy $n^{\Omega(1)}$ to be infeasible.[4] It is not immediately obvious that there are *any* natural, infeasible problems, especially if we are considering the full power of RN[∞], where message congestion is not an issue. In this paper we demonstrate that the energy landscape is rich, and that even coarsely classifying the energy complexity of simple problems is technically challenging and demands the development of new algorithm design techniques. Our results are as follows

- We develop a recursive BreadthFirstSearch algorithm in RN[$O(\log n)$] with "intermediate" energy-complexity $2^{O(\sqrt{\log n \log\log n})} = n^{o(1)}$. The algorithm involves simulating itself on a clustered version of the input graph. Due to the

[4] These definitions seem to be robust to certain modeling assumptions, e.g., whether collision detection is available.

nature of the RN model, this simulation is not free, but incurs a polylogarithmic increase in energy, which restricts the profitable depth of recursion to be at most $\sqrt{\log n / \log\log n}$.
- We give examples of some "hard" problems in energy-complexity, even when the model is RN[∞]. The problem of deciding whether $\mathsf{diam}(G)$ is 1 or at least 2 takes $\Omega(n)$ energy; in this case the hard graph G is dense. We adapt the construction of [1] (designed for the CONGEST model) to show that even on *sparse* graphs, with arboricity $O(\log n)$, deciding whether $\mathsf{diam}(G)$ is 2 or at least 3 takes $\tilde{\Omega}(n)$ energy.
- To complement the hardness results, we show that Diameter can be nearly 3/2-approximated[5] in RN[$O(\log n)$] with $O(n^{1/2+o(1)})$ energy, by adapting [17, 35] and using our new BreadthFirstSearch routine.

The existence of a subpolynomial-energy BreadthFirstSearch algorithm is somewhat surprising for information-theoretic reasons. Observe that the number of edges in $E(G)$ that are collectively discovered by all devices is at most the number of messages successfully received, which itself is at most the aggregate energy cost. Thus, if the per-device energy cost is $n^{o(1)}$, we can never hope to know about more than $n^{1+o(1)}$ edges in $E(G)$ — a negligible fraction of the input on dense graphs! On the other hand, it is possible to efficiently verify the *non-existence* of many non-edges. Given a candidate BFS-labeling, for example, it is straightforward to *verify* its correctness with polylog(n) energy.

Organization. In Section 2 we review the Miller-Peng-Xu [31] clustering algorithm and prove that it preserves distances better than previously known. In Section 3 we define some communications primitives and prove that they can be executed on the cluster graph (as if it were an RN[$O(\log n)$] network) at the cost of a polylogarithmic factor increase in energy usage. In Section 4 we design and analyze a recursive BFS algorithm, which uses $2^{O(\sqrt{\log n \log\log n})}$ energy. In Section 5 we consider the energy cost of approximately computing the network's Diameter.

2 CLUSTER PARTITIONING

Miller, Peng, and Xu [31] introduced a remarkably simple algorithm for partitioning a given graph into vertex-disjoint clusters with certain desirable properties. In this section we prove that the MPX clustering approximately preserves relative *distances* from the original graph significantly better than previously known.

Given a graph $G = (V, E)$, and a parameter β, each vertex $v \in V$ independently samples a random variable $\delta_v \sim \text{Exponential}(\beta)$ from the exponential distribution with mean $1/\beta$. Assign each v to the "cluster" centered at $u \in V$ that minimizes $\mathsf{dist}_G(v, u) - \delta_u$. Equivalently, we may think of a cluster forming at each vertex u at time $-\delta_u$, and spreading through the graph at a uniform rate of one edge per time unit. Each vertex v is absorbed into the first cluster to reach it, if this happens prior to time $-\delta_v$, when it would start growing its own cluster. Refer to Figure 1. Throughout the paper, *we only choose β such that $1/\beta$ is an integer.*

Miller et al. [31] were primarily interested in this construction because the algorithm parallelizes well, the clusters have diameter $O(\log(n)/\beta)$ w.h.p., and a $O(\beta)$-fraction of the edges are "cut," having their endpoints in distinct clusters. Haeupler and Wajc [16] observed that this algorithm can be efficiently implemented in the Radio Network model [10, 11], with only minor modifications.

[5] I.e., it returns a value in the range $\left[\left\lfloor \frac{2}{3}\mathsf{diam}(G)\right\rfloor, \mathsf{diam}(G)\right]$.

2.1 The Cluster Graph as a Distance Proxy

Define $\mathsf{Cl}(u)$ to be the cluster containing u. The cluster graph, $\mathsf{cluster}(G, \beta) = G^* = (V^*, E^*)$ is defined by

$$V^* = \{\mathsf{Cl}(u) \mid u \in V(G)\}$$
$$\text{and } E^* = \{(\mathsf{Cl}(u), \mathsf{Cl}(v)) \mid (u, v) \in E(G), \mathsf{Cl}(u) \neq \mathsf{Cl}(v)\}.$$

To prove that distances in G^* are a good proxy for distances in G, we make use of the following lemma, which is a slight variant of lemmas by Miller, Peng, Vladu, and Xu [30, Lemma 2.2] and Haeupler and Wajc [16, Corollary 3.8]. We include a proof for completeness.

Define $\mathsf{Ball}_G(v, \ell) = \{u \in V \mid \mathsf{dist}_G(u, v) \leq \ell\}$ to be the ball of radius ℓ around v.

Lemma 2.1. *Let $G^* = \mathsf{cluster}(G, \beta)$ be the cluster graph for G. For every positive integer j and $\ell > 0$, the probability that the number of G^*-clusters intersecting $\mathsf{Ball}_G(v, \ell)$ is more than j is at most*

$$(1 - \exp(-2\ell\beta))^j.$$

Proof. Condition on the time t that the $(j+1)$st signal would reach vertex v, as well as on the identities $v_1, \ldots, v_j$ of the vertices whose signals reach v before time t. Due to the memoryless property of the exponential distribution, each of these arrival times are independently distributed as $\min\{t, \mathsf{dist}(v_i, v)\} - X \leq t - X$, where $X \sim \text{Exponential}(\beta)$.

Now, if $\max_{1 \leq i \leq j} X_i > 2\ell$, then $\mathsf{Ball}_G(v, \ell)$ cannot intersect any clusters except those centered at $v_1, \ldots, v_j$, because they do not reach $\mathsf{Ball}_G(v, \ell)$ until times $\geq t - \ell$, whereas the first signal reached v before time $t - 2\ell$, and has therefore already flooded all of $\mathsf{Ball}_G(v, \ell)$ before time $t - \ell$. Thus,

$$\begin{aligned}&\mathbf{P}\left(\mathsf{Ball}_G(v, \ell) \text{ intersects more than } j \text{ clusters}\right)\\ &\leq \mathbf{P}\left(\forall i \in [1, j], X_i \leq 2\ell\right)\\ &= (1 - \exp(-2\ell\beta))^j.\end{aligned}$$

□

A natural way to show that G^* approximately preserves distances in G is to consider the fraction of edges in a shortest path that are "cut" by the partition, which corresponds to applying Lemma 2.1 with $\ell = 1/2$ and $j = 1$.[6] This was the approach taken in [8], but it only guarantees that the fraction of edges cut concentrates around its expectation ($O(\beta)$) for paths of length $\tilde{\Omega}(\text{poly}(\beta^{-1}))$. In Lemmas 2.2 and 2.3 we use Lemma 2.1 in a different way to bound the ratio of distances in G to those in G^*, which works even for relatively short distances. Lemma 2.2 applies to all distances (and suffices for our BFS application in Section 4) whereas Lemma 2.3 applies to distances $\Omega(\beta^{-1}\log^2 n)$.

[6] One imagines a vertex v_e in the middle of an edge e; e is cut iff $\mathsf{Ball}_G(v_e, 1/2)$ intersects two clusters, which must cover distinct endpoints of e.

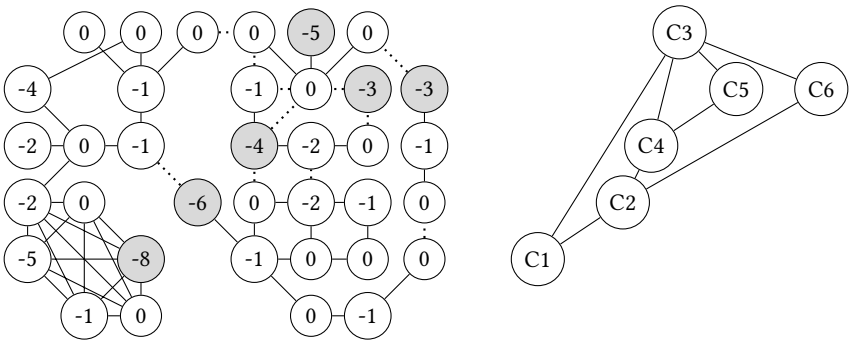

Figure 1: Constructing a cluster graph. At left, the original graph; with the (rounded) start time, $-\delta_v$, marked on each vertex. The cluster centers have been darkened, and the dotted lines indicate edges that cross a cluster boundary. At right, the corresponding cluster graph. Note that the distances in the cluster graph are broadly proportional to the original distances, but can vary significantly.

Lemma 2.2. *Let $G^* = \mathsf{cluster}(G, \beta)$ be a clustering of G. There exists a constant C such that for every pair $u, v \in V(G)$,*

$$\mathbf{P}\left(\mathrm{dist}_{G^*}(\mathsf{Cl}(u), \mathsf{Cl}(v)) \in \left[\left\lfloor \frac{\mathrm{dist}_G(u,v)\cdot\beta}{8\log(n)} \right\rfloor, \lceil \mathrm{dist}_G(u,v)\cdot\beta \rceil \cdot C\log(n)\right]\right) \geq 1 - \frac{1}{n^3}.$$

More generally, let $P = (u, \ldots, v)$ be any length-d path connecting u and v. With probability $1 - \frac{1}{n^3}$, there exists a path P^* in G^* connecting $\mathsf{Cl}(u)$ and $\mathsf{Cl}(v)$ with length at most $d \cdot C\beta\log(n)$, where each cluster in P^* intersects P.

Proof. First observe that the probability of any δ_v-value being outside $[0, 4\log(n)/\beta)$ is $\ll n^{-4}$ and hence all clusters have radius less than $4\log(n)/\beta$ with probability $\ll n^{-3}$. This gives the lower bound on $\mathrm{dist}_{G^*}(u, v)$.

For the upper bound, define ℓ to be the integer $1/\beta$. Fix any length-d path P from u to v (e.g., a shortest path, with $d = \mathrm{dist}_G(u, v)$), and cover its vertices with $\left\lceil \frac{d}{2\ell+1} \right\rceil$ paths of length 2ℓ. Applying Lemma 2.1 to the center vertex u' of one of these subpaths, we conclude that the number of clusters that intersect $\mathsf{Ball}_G(u', \ell)$, (which includes the entire subpath) is more than j with probability

$$(1 - \exp(-2\beta\ell))^j = (1 - \exp(-2))^j, \tag{1}$$

Choosing j to be the appropriate multiple of $\log(n)$, we can make this probability $\ll n^{-4}$. Taking a union bound over the $\approx \beta d/2 < n$ subpaths, the probability that any subpath intersects more than $C\log(n)$ clusters is $\ll n^{-3}$. This concludes the proof. □

Lemma 2.2 suffices to achieve our main result, BFS labeling in $2^{O(\sqrt{\log n \log\log n})}$ energy, but the exponent can be improved by a constant factor by using Lemma 2.3 whenever applicable. We include the proof of Lemma 2.3 since it may be of independent interest.

Lemma 2.3. *Let $G^* = \mathsf{cluster}(G, \beta)$ be a clustering of G. There exists a constant C such that for every pair $u, v \in V(G)$*

$$\mathbf{P}\left(\mathrm{dist}_{G^*}(\mathsf{Cl}(u), \mathsf{Cl}(v)) \in \left[\frac{\mathrm{dist}_G(u,v)\cdot\beta}{8\log(n)},\ \mathrm{dist}_G(u,v)\cdot C\beta\right]\right) \geq 1 - \frac{1}{n^3}.$$

Proof. We condition on the event that all cluster radii are at most $4\log(n)/\beta$, which fails to hold with probability $\ll n^{-3}$. As before, the lower bound on $\mathrm{dist}_{G^*}(u, v)$ follows from this event. Furthermore, this implies that sufficiently distant segments on the shortest u-v path are essentially independent.

As before, cover the vertices of the shortest u-v path with length-2ℓ subpaths, $\ell = 1/\beta$, and color the subpaths with $4\log(n) + 1$ colors such that any two subpaths of the same color are at distance at least $8\log(n)/\beta$. Each color-class contains $\Omega(\log n)$ subpaths. By Lemma 2.1 and (1), the number of clusters intersecting subpaths of a particular color class is stochastically dominated by the sum of $\Omega(\log n)$ geometrically distributed random variables with constant expectation $\frac{1}{1-(1-\exp(-2))} = \exp(2)$. By a Chernoff bound, the probability that this sum deviates from its expectation by more than a constant factor is $1/\mathrm{poly}(n)$. Hence, for sufficiently large C (controlling the number of summands and the tolerable deviation) the probability that any color-class hits too many distinct clusters is $\ll n^{-3}$. □

Remark 2.1. *Lemma 2.3 cannot be improved by more than constant factors. It is easy to construct families of graphs for which both the upper and lower bounds are tight, with high probability, depending on which vertex pairs are chosen.*

2.2 Distributed Implementation

The definition of $\mathsf{cluster}(G, \beta)$ immediately lends itself to a distributed implementation in radio networks, as was noted in [16]. For completeness we show how it can be reduced to calls to Local-Broadcast.

Local-Broadcast: We are given two disjoint vertex sets $\mathcal{S}$ and $\mathcal{R}$, where each vertex $u \in \mathcal{S}$ holds a message m_u. An

Local-Broadcast algorithm guarantees that for every $v \in \mathcal{R}$ with $N(v) \cap \mathcal{S} \neq \emptyset$, with probability $1 - f$, v receives some message m_u from *at least one* vertex $u \in N(v) \cap \mathcal{S}$. We only apply this routine with $f = 1/\text{poly}(n)$.

Lemma 2.4. Local-Broadcast *can be implemented in* $O(\log \Delta \log f^{-1})$ *time and energy, where* $\Delta \leq n-1$ *is an upper bound on the maximum degree. Senders use* $O(\log f^{-1})$ *energy; receivers that hear a message use* $O(\log \Delta)$ *energy in expectation; receivers that hear no message use* $O(\log \Delta \log f^{-1})$ *energy.*

Proof. This lemma follows from a small modification to the *Decay* algorithm [4], which is known to be optimal in terms of *time*; see Newport [33]. For the sake of completeness, we provide a proof here. Each sender $u \in \mathcal{S}$ repeats the following $O(\log f^{-1})$ times. Randomly pick an $X_u \in [1, \log \Delta]$ such that $\mathbf{P}(X_u = t) \geq 2^{-t}$ and transmit m_u at time step X_u. The energy of any sender is clearly $O(\log f^{-1})$ with probability 1. For a receiver $v \in \mathcal{R}$, if the number of senders in $N(v)$ is in the range $[2^{t-1}, 2^t]$, v will receive *some* message with constant probability in the tth timestep of every iteration. Receivers with no adjacent sender will never detect this, and spend $\Theta(\log \Delta \log f^{-1})$ energy. □

We show that cluster(G, β) can be computed, w.h.p., using $4\log(n)/\beta$ Local-Broadcasts in the communication network $G = (V, E)$. Every vertex u will learn its cluster-identifier ID(Cl(u))) and get a label $\mathcal{L}(v)$ such that $\mathcal{L}(v) = 0$ iff v is a cluster center and $\mathcal{L}(v) = i$ iff there is a $u \in N(v)$ with $\mathcal{L}(u) = i - 1$ such that $\text{Cl}(u) = \text{Cl}(v)$. If $\mathcal{L}(v) = i$, we say that v is at *layer* i.

The graph cluster(G, β) is constructed as follows. Every vertex v picks a value $\delta_v \sim$ Exponential(β) and sets its start time to be $\text{start}_v \leftarrow \lceil \frac{4\log(n)}{\beta} - \delta_v \rceil$. With probability at least $1 - 1/n^3$, all start times are positive. For $i = 1$ to $4\log(n)/\beta$, do the following. At the beginning of the ith iteration, if v is not yet in any cluster and $\text{start}_v = i$, then v becomes a cluster center and sets $\mathcal{L}(v) = 0$. During the ith iteration, we execute Local-Broadcast with $\mathcal{S}$ being the set of all clustered vertices and $\mathcal{R}$ the set of all as-yet unclustered vertices. The message of $u \in \mathcal{S}$ contains ID(Cl(u)) and $\mathcal{L}(u)$. Any vertex $v \in \mathcal{R}$ receiving a message from $u \in \mathcal{S}$ joins u's cluster and sets $\mathcal{L}(v) = \mathcal{L}(u) + 1$. Lemma 2.5 follows immediately from the above construction.

Lemma 2.5. *The cluster graph* cluster(G, β) *can be constructed using* $4\log(n)/\beta$ Local-Broadcast*s with probability* $1 - 1/n^3$. *This takes* $O(\log^3(n)/\beta)$ *time and* $O(\log^3(n)/\beta)$ *energy per vertex.*

3 COMMUNICATION PRIMITIVES FOR THE CLUSTER GRAPH

Our BFS algorithm forms a cluster graph G^* and computes BFS recursively on numerous subgraphs of G^*. In order for this type of recursion to work, we need to argue that algorithms on the (abstract) G^* can be simulated, with some time and energy cost, on the underlying G. We focus on algorithms that are composed *exclusively* of calls to Local-Broadcast (as our BFS algorithm is), but the method can be used to simulate arbitrary radio network algorithms.

We use the primitives Down-cast and Up-cast to allow cluster centers to disseminate information to their constituents and gather information from some constituent.

Down-cast: There is a set $\mathcal{U}$ of vertices such that each $u \in \mathcal{U}$ is a cluster center, and the goal is to let each $u \in \mathcal{U}$ broadcast a message m_u to all members of Cl(u).

Up-cast: There is a set $\mathcal{U}$ of vertices such that each $u \in \mathcal{U}$ wants to deliver a message m_u to the center of Cl(u). Any cluster center v with at least one $u \in \mathcal{U} \cap \text{Cl}(v)$ must receive *any* message from one such vertex.

Lemma 3.1. Up-cast *and* Down-cast *can be implemented with* $O\left(\frac{\log^3 n}{\beta \log(1/\beta)}\right)$ *calls to* Local-Broadcast *on* G*, in which each vertex participates in* $O(\log n)$ Local-Broadcast*s. I.e., the total time and energy per vertex are* $O\left(\frac{\log^5 n}{\beta \log(1/\beta)}\right)$ *and* $O(\log^3 n)$*, respectively.*

Proof. Consider the following two quantities:

$C = O(\log_{(1/\beta)} n)$. By Lemma 2.1, C is an upper bound on the number of clusters intersecting $N(v) \cup \{v\}$, with high probability. Intuitively, C represents the *contention* at v.

$\mathcal{D} = 4\log(n)/\beta$ is the maximum radius of any cluster, i.e., the maximum $\mathcal{L}$-value is at most $\mathcal{D}$.

If there were only *one* cluster, then doing an Up-cast or Down-cast would be easily reducible to $O(\log(n)/\beta)$ Local-Broadcasts. In order to minimize interference between neighboring clusters, we modify, slightly, the clustering algorithm so that all constituents of a cluster have shared randomness. When a new cluster center v is formed, it generates a subset $S_{\text{Cl}(v)} \subset [\ell]$, $\ell = \Theta(C \log n)$, by including each index independently with probability $1/C$. It disseminates $S_{\text{Cl}(v)}$ to all members of Cl(v) along with ID(Cl(v)). It is straightforward to show that with probability $1 - 1/\text{poly}(n)$, for every v,

$$\text{There exists } j \in [\ell] : j \in S_{\text{Cl}(v)} \text{ and for all } u \in N(v), j \notin S_{\text{Cl}(u)} \tag{2}$$

Down-cast is implemented in $\mathcal{D}$ stages, each stage consisting of ℓ steps. In step j of stage i, we execute Local-Broadcast with $\mathcal{S}$ consisting of every v with a message to send such that $\mathcal{L}(v) = i - 1$ and $j \in S_{\text{Cl}(v)}$, and with $\mathcal{R}$ consisting of every u with $\mathcal{L}(u) = i$ and $j \in S_{\text{Cl}(u)}$. By (2), during stage i, every layer-i vertex in every participating cluster receives the cluster center's message with high probability. An Up-cast is performed in an analagous fashion.

Each Up-cast/Down-cast performs $\ell\mathcal{D} = \Theta(C\mathcal{D}\log n) = O(\frac{\log^3 n}{\beta \log(1/\beta)})$ executions of Local-Broadcast on G, for a total of $O(\frac{\log^5 n}{\beta \log(1/\beta)})$ time. Each vertex v participates in $O(|S_{\text{Cl}(v)}|)$ Local-Broadcasts, which is $O(\log n)$ w.h.p., for a total of $O(\log^3 n)$ energy. □

Lemma 3.2. *A call to* Local-Broadcast *on the cluster graph* $G^* = \text{cluster}(G, \beta)$ *can be simulated with* $O\left(\frac{\log^3 n}{\beta \log(1/\beta)}\right)$ *calls to* Local-Broadcast *on* G*; each vertex in* $V(G)$ *participates in* $O(\log n)$ Local-Broadcast*s.*

Proof. Let $\mathcal{S}$ and $\mathcal{R}$ be the sets of sending and receiving clusters in G^*. All members of C know that C is in $\mathcal{S}$ or $\mathcal{R}$. The Local-Broadcast algorithm has three steps.

(1) Begin by doing a Down-cast in each $C \in \mathcal{S}$. Each member of C learns the message m_C.
(2) Perform one Local-Broadcast on G, with sender set $\bigcup_{C\in\mathcal{S}} C$ and receiver set $\bigcup_{C'\in\mathcal{R}} C'$. At this point, w.h.p., every $\mathcal{R}$-cluster adjacent to an $\mathcal{S}$-cluster has at least one constituent that has received a message.
(3) Finally, do one Up-cast on every cluster $C \in \mathcal{R}$ to let the cluster center of C learn one message from a constituent of C, if any.

The algorithm clearly satisfies the requirement of Local-Broadcast on cluster(G, β). The number of calls to Local-Broadcast on G is $O(C\mathcal{D}\log n) = O\left(\frac{\log^3 n}{\beta\log(1/\beta)}\right)$ and each vertex participates in $O(\log n)$ of them. □

4 BFS WITH SUB-POLYNOMIAL ENERGY

4.1 Technical Overview

Suppose every vertex in the graph could cheaply compute its distance from the source up to an *additive* $\pm\rho$ error. Given this knowledge, we could trivially solve exact BFS in $\tilde{O}(D)$ time and $\tilde{O}(\rho)$ energy per vertex, simply by letting vertices sleep through steps that they need not participate in. In particular, we would advance the *BFS wavefront* one layer at a time using calls to Local-Broadcast, except that each vertex u would sleep through the first $\widetilde{\mathrm{dist}}_G(s,u)-\rho$ calls to Local-Broadcast, where $\widetilde{\mathrm{dist}}_G$ is the approximate distance. It would be guaranteed to fix $\mathrm{dist}_G(s,u)$ (and halt) in the next 2ρ calls to Local-Broadcast.

Lemmas 2.2 and 2.3 suggest a method of obtaining approximate distances. If we computed the cluster graph $G^* = \mathrm{cluster}(G,\beta)$ and then computed *exact* distances on G^*, Lemmas 2.2 and 2.3 allow us to approximate all distances from the source, up to an additive error of $\tilde{O}(\beta^{-1})$ (for small distances) and multiplicative error of w^2 (for larger distances), where $w = \Theta(\log n)$ is a sufficiently large multiple of $\log n$. Note that, from the perspective of energy efficiency, the main advantage to computing distances in G^* rather than G is that G^* has a smaller diameter $w\beta\cdot\mathrm{diam}(G)$.

Our algorithm computes distances up to D by advancing the *BFS wavefront* in $\lceil\beta D\rceil$ stages, extending the radius β^{-1} per stage. The *ith wavefront* W_i is defined to be the vertex set

$$W_i = \{u \in V(G) \mid \mathrm{dist}_G(S,u) = i\beta^{-1}\},$$

where S is the set of sources. (Recall that β^{-1} is an integer.) To implement the ith stage correctly it suffices to activate a vertex set X_i that includes all the affected vertices, in particular:

$$X_i \supset \{u \in V(G) \mid \mathrm{dist}_G(S,u) \in [i\beta^{-1},(i+1)\beta^{-1}]\} \quad \text{(w.h.p.)}$$

In order for each vertex u to decide whether it should join X_i or sleep through the ith stage, u maintains lower and upper bounds on its distance to the ith wavefront, or more accurately, the distance from its cluster $\mathsf{Cl}(u)$ to W_i in G.

Invariant 4.1. *Before the ith stage begins, each vertex u knows $L_i(\mathsf{Cl}(u))$ and $U_i(\mathsf{Cl}(u))$ such that*

$$\mathrm{dist}_G(W_i,\mathsf{Cl}(u)) = \mathrm{dist}_G(S,\mathsf{Cl}(u)) - i\beta^{-1} \in [L_i(\mathsf{Cl}(u)), U_i(\mathsf{Cl}(u))].$$

Clearly, if some cluster C satisfies Invariant 4.1 at stage $i-1$ with the interval $[L_{i-1}(C), U_{i-1}(C)]$, it also satisfies Invariant 4.1 at stage i with $L_i(C) = L_{i-1}(C)-\beta^{-1}$ and $U_i(C) = U_{i-1}(C)-\beta^{-1}$ since the $(i-1)$th stage advances the wavefront by exactly β^{-1}. In the algorithm these are called *Automatic Updates*; they can be done locally, without expending any energy. In order to keep the interval $[L_i(C), U_i(C)]$ relatively narrow (and hence useful for keeping vertices in C asleep), we occasionally refresh it with a *Special Update*. Let $W_i^* \subseteq V(G^*)$ be the clusters in G^* that intersect the wavefront W_i. We call BFS on a subgraph G_i^* of G^* from the source-set W_i^*, up to a radius of $Z[i]$. The only clusters that participate in this recursive call are those that are likely to be relevant, i.e., those C for which $L_i(C) \le Z[i]\cdot\beta^{-1}$. (The $Z[i]$ sequence will be defined shortly.) After this recursive call completes we update $[L_i(C), U_i(C)]$ for all participating C by applying Lemmas 2.2 and 2.3 to the (exact) distance $\mathrm{dist}_{G_i^*}(W_i^*, C)$ obtained in the cluster graph.

Specification. Our Recursive-BFS procedure (see Figure 2) takes four parameters: G, the graph, $S \subset V(G)$, the set of sources, $A \subseteq V(G)$, the set of *active* vertices (which is a superset of S), and D, the depth of the search. When we make a call to Recursive-BFS, every vertex can locally calculate D and whether it is in S or A.[7] G^* denotes the cluster graph returned by cluster(G,β), where β is a parameter fixed throughout the computation. We compute G^* once, just before the first recursive call to Recursive-BFS$(G,\cdot,\cdot,\cdot)$; subsequent calls to Recursive-BFS on G with different (S,A,D) parameters can use the same G^*. It is important to remember that G can be either the actual radio network (RN) or a *virtual* RN on which we can simulate RN algorithms, with a certain overhead in terms of time and energy. At the termination of Recursive-BFS(G,S,A,D), every vertex $u \in A$ returns $\mathrm{dist}_A(S,u)$ if it is at most D, and ∞ otherwise. Vertices in $V(G)\backslash A$ expend no energy.

Correctness. If one believes that the algorithm (Figure 2) faithfully implements the high level description given so far, its correctness is immediate. Every time we set $[L_i(C), U_i(C)]$ the interval is correct with probability $1 - 1/\mathrm{poly}(n)$, either because $[L_{i-1}(C), U_{i-1}(C)]$ is correct (an Automatic Update), or because they are set according to Lemmas 2.2 and 2.3, which hold with probability $1 - 1/\mathrm{poly}(n)$ (Special Update). If $L_i(C)$ is correct for all C, then X_i will include all vertices necessary to compute the $(i+1)$th wavefront, and the ith stage will succeed, up to the $1/\mathrm{poly}(n)$ error probability inherent in calls to Local-Broadcast. The main question is whether the procedure is *efficient*.

Efficiency. We will argue that for a very specific $Z[\cdot]$ sequence, which guides the Special Update steps, the following claims hold:

Claim 1. *Each vertex is included in the set X_i for $\tilde{O}(1)$ values of i.*

Claim 2. *For each vertex u, $\mathsf{Cl}(u)$ is included in G_i^* for $\tilde{O}(1)$ values of i.*

Our algorithms (cluster and Recursive-BFS) are based solely on calls to Local-Broadcast. Define $\mathsf{En}(D)$ to be the number of calls to Local-Broadcast that one vertex participates in when computing BFS to distance D. If Claims 1 and 2 hold, then

$$\mathsf{En}(D) \le \tilde{O}(1)\cdot\mathsf{En}(\tilde{O}(\beta D)) + \tilde{O}(\beta^{-1}) \tag{3}$$

[7] The purpose of the A parameter is to refrain from computing useless information. E.g., when we compute the distance from the clusters W_i^* intersecting the ith wavefront, we are only interested in distances to clusters intersecting as-yet unvisited vertices (those intersecting A), not settled vertices "behind" the wavefront.

Recursive-BFS(G, S, A, D)

[Initialize Distance Estimates]

1. Call Recursive-BFS(G^*, S^*, A^*, D^*) where $D^* = w\beta D$. For each cluster C in A^*,
$$L_0(C) \leftarrow \text{dist}_{A^*}(S^*, C) \cdot \frac{1}{\beta w}, \qquad U_0(C) \leftarrow \max\left\{w\beta^{-1}, w^2 \cdot L_0(C)\right\}.$$
2. $A = A \backslash \{u \mid L_0(\text{Cl}(u)) = \infty\}$. *(Deactivate vertices at distance greater than D, w.h.p.)*
3. For i from 0 to $\lceil \beta D \rceil - 1$

[Iteratively Advance BFS Wavefront β^{-1} Steps]

4. Define $X_i = \{u \in A \mid L_i(\text{Cl}(u)) \le \beta^{-1}\}$.
5. Advance BFS wavefront from W_i to W_{i+1} using β^{-1} calls to Local-Broadcast. Only vertices in X_i participate in this step.
6. $A \leftarrow A \backslash \{u \mid \text{dist}_G(S, u) < (i+1)\beta^{-1}\}$. *(Deactivate settled vertices.)*

[Estimate Distances to $(i+1)$th Wavefront W_{i+1}]

7. Define G^*_{i+1} to be the subgraph of G^* induced by
$$\text{Y} = \{C \in A^* \mid L_i(C) \le (Z[i+1] + 1) \cdot \beta^{-1}\}.$$
Vertices in Y-clusters participate in a *Special Update*. Call Recursive-BFS$(G^*, W^*_{i+1}, \text{Y}, Z[i+1])$. For each cluster C with $\text{dist}_{G^*_{i+1}}(W^*_{i+1}, C) = x$, set
$$L_{i+1}(C) \leftarrow \min\{Z[i+1] \cdot \beta^{-1} + 1,\ x \cdot \beta^{-1}/w\},$$
$$U_{i+1}(C) \leftarrow \min\{U_i(C) - \beta^{-1},\ \max\{x, 1\} \cdot \beta^{-1} w\}.$$
8. Active vertices that did not participate in the Special Update perform an *Automatic Update*. For each $C \in A^* \backslash \text{Y}$,
$$L_{i+1}(C) \leftarrow L_i(C) - \beta^{-1}$$
$$U_{i+1}(C) \leftarrow U_i(C) - \beta^{-1}$$

Figure 2

The $\tilde{O}(\beta^{-1})$ term accounts for the cost of computing G^* (Lemma 2.5) and the $\tilde{O}(1)$ times a vertex is included in X_i (Claim 1), each of which involves β^{-1} Local-Broadcasts on G. Every recursive call to Recursive-BFS$(G^*, \cdot, \cdot, D')$ has $D' = \tilde{O}(\beta D)$ and by Claim 2 each vertex participates in $\tilde{O}(1)$ such recursive calls. Moreover, according to Lemma 3.2, the *energy* overhead for simulating one call to Local-Broadcast on G^* is $\tilde{O}(1)$ calls to Local-Broadcast on G. This justifies the first term of (3). The time and energy of our algorithm is analyzed in Theorem 4.6. As a foreshadowing of the analysis, if D_0 is the distance threshold of the top-level call to Recursive-BFS, we will set set $\beta = 2^{-\sqrt{\log D_0 \log\log n}}$ and apply (3) to recursion depth $\sqrt{\log D_0 / \log\log n}$.

The Z-Sequence. The least obvious part of the Recursive-BFS algorithm is the Z-sequence, which guides how Special Updates are performed. Recall that $w = \Theta(\log n)$ is a sufficiently large multiple of $\log n$; if we are computing BFS to distance D in G, then we need never compute BFS beyond distance $D^* \ge w\beta D$ in G^*. The Z-sequence is defined as follows.

$$Y[i] = \max_{j \ge 0}\{2^j \text{ such that } 2^j | i\} \qquad (i \ge 1)$$
$$\text{I.e., } Y = (1, 2, 1, 4, 1, 2, 1, 8, 1, 2, 1, 4, 1, 2, 1, 16,$$
$$1, 2, 1, 4, 1, 2, 1, 8, 1, 2, 1, 4, 1, 2, 1, 32, \ldots)$$
$$Z[0] = D^*$$
$$Z[i] = \min\{D^*,\ \alpha \cdot Y[i]\}, \quad \text{where } \alpha = 4 \qquad (i \ge 1)$$
$$D^* = \min_{j \ge 0}\{\alpha 2^j \text{ such that } \alpha 2^j \ge w\beta D\}$$

In other words, Z is derived by multiplying Y by $\alpha = 4$, truncating large elements at D^*, and beginning the sequence at 0, with $Z[0] = D^*$. (Here $Z[0]$ corresponds to the distance threshold D^* used in Step 1 of Recursive-BFS to estimate distances to the 0th wavefront $W_0 = S$.)

Figure 3 gives an example, from the perspective of a single cluster, of how the distance estimate evolve over time.

Organization of Section 4. In Section 4.2 we prove a number of lemmas that relate to the correctness and efficiency of Recursive-BFS, including proofs of Claims 1 and 2. In Section 4.3 we analyze the overall time and energy-efficiency of the BFS algorithm.

4.2 Auxiliary Lemmas

Lemma 4.1 justifies how distance estimates are updated in Steps 1, 7, and 8 of Recursive-BFS in order to preserve Invariant 4.1, with high probability.

Lemma 4.1. *Let W_i be the ith wavefront; let* Y *include all clusters C such that* $\text{dist}_G(W_i, C) \in [i\beta^{-1}, (i + Z')\beta^{-1}]$*; and let G^*_i be the subgraph of G^* induced by* Y*. If* $\text{Cl}(u) \in \text{Y}$ *and* $\text{dist}_G(S, u) \ge i\beta^{-1}$*, then w.h.p.,*

$$\text{dist}_G(W_i, u) \in \left[\min\left\{\frac{Z'}{\beta} + 1,\ \text{dist}_{G^*_i}(W^*_i, \text{Cl}(u)) \cdot \frac{1}{w\beta}\right\},\right.$$
$$\left.\max\left\{1,\ \text{dist}_{G^*_i}(W^*_i, \text{Cl}(u))\right\} \cdot \frac{w}{\beta}\right]$$

Proof. If $d = \text{dist}_G(W_i, u) \ge Z'\beta^{-1} + 1$ then the lower bound is already correct, so suppose that $d \le Z'\beta^{-1}$. Let P be any length-d

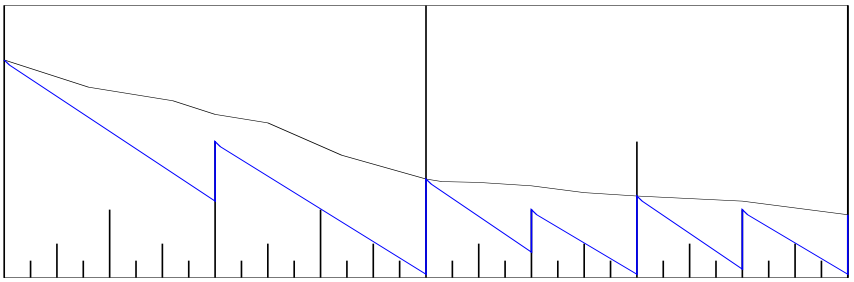

Figure 3: Part of the time evolution of the distance of a fixed cluster, C, from the frontier in the cluster graph G^*. The x-axis is time spent moving the wavefront across the underlying graph G. Every vertical tick mark is a time at which this is suspended, so that, recursively, BFS can be done on the cluster graph G^*, starting from the current wavefront. The height of each such tick mark indicates the depth to which this search is to be done. The y-axis is the distance of C to the wave front, *in* G^*. The top curve shows the irregular, but monotonic, decrease of this distance over time. The bottom curve, in blue, shows the high-probability lower bound on this distance, from the perspective of the cluster in question. Note that every time the top curve intersects a tick mark, the cluster must participate in the BFS on the cluster graph, or this BFS will fail. Every time the bottom curve intersects a tick mark, the cluster will wake up in order to participate in the BFS, because it thinks it *may* be needed. Note that, by design, the lower curve often passes just above the tick marks without actually intersecting them. The reader should bear in mind that these two curves chart the actual/likely distance of C to the wavefront in G; the algorithm maintains the related interval $[L_i(C), U_i(C)]$, which bounds distances from C to the wavefront *in* G.

path from u to W_i in G. Lemma 2.2 implies that w.h.p., there is a path P^* in G_i^* from W_i^* to $\mathsf{Cl}(u)$ with length at most $O(\beta d \log n) < w\beta d$, and so $\mathrm{dist}_{G_i^*}(W_i^*, \mathsf{Cl}(u)) \le w\beta d$, as required.

This upper bound follows from the cluster diameter upper bound $K = 8\log(n)/\beta \le w/(2\beta) - 1$. Thus, if $\mathrm{dist}_{G_i^*}(W_i^*, \mathsf{Cl}(u)) = d'$ then $\mathrm{dist}_G(W_i, u) \le (d'+1)\cdot(K+1) \le \max\{d'+1\}\cdot w\beta^{-1}$. □

Lemma 4.1 shows that Step 1 of Recursive-BFS initializes $L_0(\cdot), U_0(\cdot)$ to satisfy Invariant 4.1, w.h.p. Here $\mathrm{Y} = A^*$ is the set of all active clusters; if $\mathrm{dist}_A(S,u) \in [0,D]$ (the relevant range), then Lemma 4.1 guarantees that $\mathrm{dist}_A(S,u) \in [L_0(\mathsf{Cl}(u)), U_0(\mathsf{Cl}(u))]$ after Step 1. The estimates set in Step 8 of Recursive-BFS are trivially correct; Lemma 4.1 also guarantees that the lower and upper bounds fixed in Step 7 are correct.

We use several properties of the Z sequence, listed in Lemma 4.2.

Lemma 4.2. *Fix an index i.*

(1) *For any number $b \ge \alpha$, define $j > i$ to be the smallest index such that $Z[j] \ge b$. Then*

$$j - i \le b/\alpha.$$

Suppose the number b additionally satisfies that $b \le Z[i]$ and $b \in \{\alpha, 2\alpha, 4\alpha, 8\alpha, \ldots D^*\}$. Then we have $Z[i] = b$ and $j - i = Z[j]/\alpha$.

(2) *Define $j > i$ to be the smallest index such that $Z[j] > Z[i]$ or $Z[j] = D^*$. Then we have $j - i = Z[i]/\alpha$; moreover, all indices $k \in \{i+1, \ldots, j-1\}$ satisfy that $Z[k] \le Z[i]/2$.*

Proof. Parts 1 and 2 follow from the fact that in the Y-sequence, the values at least 2^ℓ appear periodically with period 2^ℓ. Thus, the values at least $\alpha 2^\ell$ in the Z-sequence also appear periodically with period 2^ℓ. □

We are now prepared to prove Claim 1.

Proof of Claim 1. It follows from Invariant 4.1 that X_i, as defined in Step 4 of Recursive-BFS, includes all active vertices within distance β^{-1} of the ith wavefront W_i. It remains to show no u is included in X_i for more than $\mathrm{poly}(\log n)$ indices i.

Suppose that $u \in X_i$ for $i > 0$. It follows that $L_i(\mathsf{Cl}(u)) \le \beta^{-1}$ and that in the previous stage, $L_{i-1}(\mathsf{Cl}(u)) \le 2\beta^{-1}$. Since $Z[i] \ge \alpha = 4$, it must have been that $\mathsf{Cl}(u)$ was included in Y and participated in the Special Update (Step 7 of Recursive-BFS) before stage i. If $\mathrm{dist}_{G_i^*}(W_i^*, \mathsf{Cl}(u)) = x$ and after the Special Update, $L_i(\mathsf{Cl}(u)) \le \beta^{-1}$, it must be that $x \le w$, and hence $U_i(\mathsf{Cl}(u)) \le w^2\beta^{-1}$. Thus, u may participate in at most w^2 more stages (joining $X_i, X_{i+1}, \ldots, X_{i+w^2}$) before its distance is settled and it is deactivated, in Step 6 of Recursive-BFS. □

Before proving Claim 2 we begin with three auxiliary lemmas, Lemmas 4.3, 4.4, and 4.5. We defer the proofs of these three lemmas to the full version of the paper.

Lemma 4.3. *Recall $\alpha = 4$. Suppose cluster C is included in G_i^* and G_j^*, but not in $G_{i'}^*$ for any $i' \in \{i+1, \ldots, j-1\}$. Then we have*

$$\frac{L_i(C)}{8\alpha} \le \frac{j-i}{\beta} \le \max\left\{\frac{1}{\beta}, \frac{L_i(C)}{\alpha}\right\}.$$

Lemma 4.4. *Suppose C appears in G_i^* and G_j^* but not in $G_{i'}^*$ for any $i' \in \{i+1, \ldots, j-1\}$. Suppose that when $L_i(C)$ is set during a Special Update (Step 7 of* Recursive-BFS*), we have $L_i(C) = (Z[i]/\beta) + 1$. It must be that $Z[j] > Z[i]$ or $Z[j] = D^*$.*

In the Recursive-BFS algorithm, the upper bound estimates $U_i(C)$ are all monotonically decreasing with i, due to the way Special and Automatic Updates are performed in Steps 7 and 8. On the other hand, the lower bound estimates $L_i(C)$ are only monotonically decreasing during Automatic Updates and may oscillate many times over the execution of the algorithm. (See Figure 3 for a depiction of how this happens.) Since $U_\cdot(\cdot)$-values offer a more stable way to measure progress, we need to connect them with the $L_\cdot(\cdot)$-values, which directly influence the composition of X_i and G_i^*.

Lemma 4.5. *If $[L_i(C), U_i(C)]$ is set during a Special Update step, then*

$$U_i(C) \le \max\{2w^2 \cdot L_i(C),\ 2w^2 \cdot \beta^{-1}\}$$

We are now in a position to prove Claim 2, that each vertex participates in G_i^* for at most $\tilde{O}(1)$ indices i.

Proof of Claim 2. Suppose that C participates in a Special Update that sets $[L_i(C), U_i(C)]$ with $U_i(C) \ge 2w^2 \cdot \beta^{-1}$ and that the next interval to be set by a Special Update is $[L_j(C), U_j(C)]$. Then

$$(j - i) \ge \frac{\beta \cdot L_i(C)}{8\alpha} \ge \frac{\beta \cdot U_i(C)}{16\alpha w^2}. \tag{4}$$

The first inequality of (4) follows from Lemma 4.3 and the second inequality from Lemma 4.5. Since $U_*(C)$ is decremented by at least β^{-1} in each stage, (4) implies that

$$U_j(C) \le U_i(C) - (j - i) \cdot \beta^{-1} \le U_i(C)\left(1 - \frac{1}{16\alpha w^2}\right).$$

In other words, C participates in at most $\log_{1+\Theta(1/w^2)} D = \Theta(w^2 \log D) = O(\log^3 n)$ Special Updates until some stage i in which $U_i(C) < 2w^2 \cdot \beta^{-1}$, after which C participates in at most $O(w^2)$ Special Updates all constituents of C settle their distance from the source and are deactivated. □

4.3 Time and Energy Complexity of BFS

The remainder of this section constitutes a proof of Theorem 4.6.

Theorem 4.6. *Let $G = (V, E)$ be a radio network, $s \in V$ be a distinguished source vertex, and $D = \max_u \mathrm{dist}_G(s, u)$. A Breadth First Search labeling can be computed in $\tilde{O}(D) \cdot 2^{O(\sqrt{\log D \log\log n})}$ time and $\tilde{O}(1) \cdot 2^{O(\sqrt{\log D \log\log n})}$ energy, with high probability.*

The main problem is to compute BFS up to some threshold distance D_0. Once we have a solution to this problem, we can obtain bounds in terms of the (unknown) D parameter by testing every $D_0 = 2^k$ that is a power of 2, stopping at the first value that labels all of $V(G)$. We use a call to Local-Broadcast as a unit of measurement of both time and energy, i.e., calling Local-Broadcast takes one unit of time, and every *participating* vertex expends one unit of energy. (By Lemma 2.4 actual time and energy are at most a $O(\log^2 n)$ factor larger.)

The algorithm we apply is a slightly modified Recursive-BFS, where all cluster graphs in all recursive invocations are constructed with $\beta = 2^{-\sqrt{\log D_0 \log\log n}}$. We only apply Recursive-BFS to recursion depth $L = \sqrt{\log D_0 / \log\log n}$, at which point we revert to the trivial BFS algorithm that settles all distances up to D' using D' time and energy, by calling Local-Broadcast D' times.

Define $\mathsf{En}_r(D')$ to be the number of calls to Local-Broadcast that a vertex participates in when computing BFS to distance D', and when the recursion depth is $r \in [0, L]$. Thus, we have

$$\mathsf{En}_r(D') = \begin{cases} \tilde{O}(1) \cdot \mathsf{En}_{r+1}(\tilde{O}(\beta D')) + \tilde{O}(\beta^{-1}) & \text{if } r < L \\ D' & \text{if } r = L \end{cases}$$

By Lemma 2.5 the cost to create the cluster graph G^* is $\tilde{O}(\beta^{-1})$. By Claim 1 each vertex appears in X_i for $\tilde{O}(1)$ stages i, and for each, participates in β^{-1} calls to Local-Broadcast. These costs are covered by the $\tilde{O}(\beta^{-1})$ term. All calls to Recursive-BFS on G^* involve computing BFS to some distance at most $D^* = w\beta D' = \tilde{O}(\beta D')$. By Claim 2, every vertex participates in $\tilde{O}(1)$ such recursive calls. Moreover, by Lemma 3.2, every cluster C (vertex in G^*) that participates in a call to Local-Broadcast on G^* can be simulated such that constituent vertices of C participate in $\tilde{O}(1)$ calls to Local-Broadcast on G. The costs of recursive calls are represented by the $\tilde{O}(1) \cdot \mathsf{En}_{r+1}(\tilde{O}(\beta D'))$ term.

When the recursion depth r reaches L, the *maximum* value of D' is therefore at most

$$D_L = D_0 \cdot (\tilde{O}(\beta))^L = (\tilde{O}(1))^L = 2^{O(\sqrt{\log D_0 \log\log n})},$$

since $\beta^L = D_0^{-1}$. Thus, the energy cost of the top-level recursive call is at most

$$\mathsf{En}_0(D_0) = (\tilde{O}(1))^L \cdot (D_L + \tilde{O}(\beta^{-1})) = \tilde{O}(1) \cdot 2^{O(\sqrt{\log D_0 \log\log n})}.$$

We can set up a similar recursive expression for the time of this algorithm.

$$\mathsf{Time}_r(D') \le \begin{cases} O(D') + \tilde{O}(\beta^{-1}) \cdot \sum_{i=0}^{\lceil \beta D' \rceil - 1} \mathsf{Time}_{r+1}(Z[i]) & \text{if } r < L \\ D' & \text{if } r = L \end{cases}$$

The $r = L$ case is the time of the trivial algorithm, so we focus on justifying the expression for $r < L$. The time to advance the BFS wavefront over all $\lceil \beta D' \rceil$ stages of Step 5 is $O(D')$. We treat Step 1 as the Special Update for $i = 0$ with $Z[0] = D^*$. In general, the Special Update for stage i takes $\mathsf{Time}_{r+1}(Z[i])$ time with respect to G^*, and each unit of time (i.e., a call to Local-Broadcast) is simulated in G in time linear in the maximum cluster diameter, namely $\tilde{O}(\beta^{-1})$. By Lemma 4.2, each value $b \in B = \{\alpha, 2\alpha, 4\alpha, \ldots, D^*\}$ appears less than $(\beta D'/b)$ times in $Z[0], \ldots, Z[\lceil \beta D' \rceil - 1]$, hence we can rewrite the sum as $\sum_{b \in B} (\beta D'/b) \cdot \mathsf{Time}_{r+1}(b)$. Assuming inductively that $\mathsf{Time}_{r+1}(b)$ is $b \cdot (\tilde{O}(1))^{L-(r+1)}$, which holds when $r + 1 = L$, we have

$$\begin{aligned} \mathsf{Time}_r(D') &\le O(D') + \tilde{O}(\beta^{-1}) \cdot \sum_{b \in B} (\beta D'/b) \cdot \mathsf{Time}_{r+1}(b) \\ &= O(D') + \tilde{O}(1) \cdot \sum_{b \in B} (D'/b) \cdot b \cdot (\tilde{O}(1))^{L-(r+1)} \\ &= D' \cdot (\tilde{O}(1))^{L-r} \end{aligned}$$

Hence $\mathsf{Time}_0(D_0) = D_0 \cdot (\tilde{O}(1))^L = \tilde{O}(D_0) \cdot 2^{O(\sqrt{\log D_0 \log\log n})}$.

5 HARDNESS OF DIAMETER APPROXIMATION

In this section, we show that certain approximations of diameter cannot be computed in $o(n)$ energy, even allowing messages of unlimited size. Our lower bounds also hold in the setting where the network supports *collision detection*, i.e., in each time slot t, each listener v is able to distinguish between the following two cases: (i) at least two vertices in $N(v)$ transmit at time t (noise), or (ii) no vertex in $N(v)$ transmits at time t (silence). All proofs in this section are omitted due to the page constraint.

Theorem 5.1. *The energy complexity of computing a $(2 - \epsilon)$-approximation of diameter is $\Omega(n)$, even on the class of unit-disc graphs.*

Theorem 5.1 is proved by showing that it takes $\Omega(n)$ energy to distinguish between (i) an n-vertex complete graph K_n (which has

diameter 1), or (ii) an n-vertex complete graph minus one edge $K_n - e$ (which has diameter 2).

Theorem 5.2. *The energy complexity of computing an $(3/2-\epsilon)$-approximation of diameter is $\Omega(n/\log^2 n)$, even on graphs of $O(\log n)$-arboricity or $O(\log n)$ treewidth.*

The proof of Theorem 5.2 follows the framework of [1], which shows that computing diameter takes $\Omega(n/\log^2 n)$ time in the CONGEST model, or more generally $\Omega\left(\frac{n}{B\log n}\right)$ time in the message-passing model with B-bit message size constraint. Note that a time lower bound in CONGEST does not, in general, imply any lower bound in RN$[\infty]$, which *has no message size constraint*. The main challenge for proving Theorem 5.2 is that we allow messages of unbounded length.

5.1 Upper Bounds

The approximation ratios in Theorems 5.1 and 5.2 cannot be improved. Observe that BFS already gives a 2-approximation of diameter, as $D' = \max_{u\in V(G)}\{\mathsf{dist}_G(s,u)\} \in [\mathsf{diam}(G)/2, \mathsf{diam}(G)]$, and we know that a BFS can be computed in $n^{o(1)}$ energy.

Theorem 5.3. *There is an algorithm that computes a 2-approximation of diameter in $n^{1+o(1)}$ time and $n^{o(1)}$ energy.*

If we allow an energy budget of $n^{\frac{1}{2}+o(1)}$ then it is possible to achieve a *nearly* 3/2-approximation by applying the algorithm of [17, 35], which computes a D' such that $\lfloor 2\mathsf{diam}(G)/3\rfloor \le D' \le \mathsf{diam}(G)$. More precisely, if we write $\mathsf{diam}(G) = 3h + z$, where h is a non-negative integer, and $z \in \{0,1,2\}$, then $D' \in [2h+z, \mathsf{diam}(G)]$ for the case $z = 0, 1$, and $D' \in [2h+1, \mathsf{diam}(G)]$ for the case $z = 2$. Note that this does not contradict the $\Omega(n)$ energy lower bound for distinguishing between $\mathsf{diam}(G) = 1$ and $\mathsf{diam}(G) = 2$ in Theorem 5.1, nor does it contradict Theorem 5.2.

Theorem 5.4. *There is an algorithm that computes an approximation D' such that $\lfloor 2\mathsf{diam}(G)/3\rfloor \le D' \le \mathsf{diam}(G)$ in $n^{3/2+o(1)}$ time and $n^{1/2+o(1)}$ energy.*

REFERENCES

[1] A. Abboud, K. Censor-Hillel, and S. Khoury. 2016. Near-Linear Lower Bounds for Distributed Distance Computations, Even in Sparse Networks. In *Distributed Computing (DISC)*, Cyril Gavoille and David Ilcinkas (Eds.). Springer Berlin Heidelberg, 29–42.

[2] N. Alon, A. Bar-Noy, N. Linial, and D. Peleg. 1991. A lower bound for radio broadcast. *J. Comput. System Sci.* 43, 2 (1991), 290–298.

[3] R. Bar-Yehuda, O. Goldreich, and A. Itai. 1991. Efficient emulation of single-hop radio network with collision detection on multi-hop radio network with no collision detection. *Distributed Computing* 5, 2 (1991), 67–71.

[4] R. Bar-Yehuda, O. Goldreich, and A. Itai. 1992. On the time-complexity of broadcast in multi-hop radio networks: An exponential gap between determinism and randomization. *J. Comput. System Sci.* 45, 1 (1992), 104–126.

[5] M. Barnes, C. Conway, J. Mathews, and D. K. Arvind. 2010. ENS: An Energy Harvesting Wireless Sensor Network Platform. In *Proceedings of the 5th International Conference on Systems and Networks Communications (ICSNC)*. 83–87.

[6] M. Bender, T. Kopelowitz, S. Pettie, and M. Young. 2018. Contention resolution with constant throughput and log-logstar channel accesses. *SIAM J. Comput.* 47 (2018), 1735–1754. Issue 5.

[7] P. Berenbrink, C. Cooper, and Z. Hu. 2009. Energy efficient randomised communication in unknown AdHoc networks. *Theoretical Computer Science* 410, 27 (2009), 2549 – 2561.

[8] Y.-J. Chang, V. Dani, T. P. Hayes, Q. He, W. Li, and S. Pettie. 2018. The Energy Complexity of Broadcast. In *Proceedings of the 2018 ACM Symposium on Principles of Distributed Computing (PODC)*. 95–104.

[9] Y.-J. Chang, T. Kopelowitz, S. Pettie, R. Wang, and W. Zhan. 2017. Exponential separations in the energy complexity of leader election. In *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing (STOC)*. 771–783.

[10] I. Chlamtac and S. Kutten. 1985. On broadcasting in radio networks-problem analysis and protocol design. *IEEE Transactions on Communications* 33, 12 (1985), 1240–1246.

[11] I. Chlamtac and S. Kutten. 1987. Tree-Based Broadcasting in Multihop Radio Networks. *IEEE Trans. Computers* 36, 10 (1987), 1209–1223.

[12] A. Czumaj and P. Davies. 2017. Exploiting Spontaneous Transmissions for Broadcasting and Leader Election in Radio Networks. In *Proceedings of the 2017 ACM Symposium on Principles of Distributed Computing (PODC)*. 3–12.

[13] L. Gasieniec, E. Kantor, D. R. Kowalski, D. Peleg, and C. Su. 2007. Energy and Time Efficient Broadcasting in Known Topology Radio Networks. In *Proceedings 21st International Symposium on Distributed Computing (DISC)*. 253–267.

[14] M. Ghaffari and B. Haeupler. 2016. Near-Optimal BFS-Tree Construction in Radio Networks. *IEEE Communications Letters* 20, 6 (2016), 1172–1174.

[15] S. Gilbert, V. King, S. Pettie, E. Porat, J. Saia, and M. Young. 2014. (Near) optimal resource-competitive broadcast with jamming. In *Proceedings of the 26th ACM Symposium on Parallelism in Algorithms and Architectures (SPAA)*. 257–266.

[16] B. Haeupler and D. Wajc. 2016. A faster distributed radio broadcast primitive. In *Proceedings 35th ACM Symposium on Principles of Distributed Computing (PODC)*. ACM, 361–370.

[17] S. Holzer, D. Peleg, L. Roditty, and R. Wattenhofer. 2014. Brief announcement: Distributed 3/2-approximation of the diameter. In *Proc. 28th International Symposium on Distributed Computing (DISC 2014)*. Springer, 562–564.

[18] T. Jurdzinski, M. Kutylowski, and J. Zatopianski. 2002. Efficient algorithms for leader election in radio networks. In *Proceedings of the 21st Annual ACM Symposium on Principles of Distributed Computing (PODC)*. 51–57.

[19] T. Jurdzinski, M. Kutylowski, and J. Zatopianski. 2002. Energy-Efficient Size Approximation of Radio Networks with No Collision Detection. In *Proceedings of the 8th Annual International Conference on Computing and Combinatorics (COCOON)*. 279–289.

[20] T. Jurdzinski, M. Kutylowski, and J. Zatopianski. 2002. Weak Communication in Radio Networks. In *Proceedings of the 8th International European Conference on Parallel Computing (Euro-Par)*. 965–972.

[21] T. Jurdzinski, M. Kutylowski, and J. Zatopianski. 2003. Weak communication in single-hop radio networks: adjusting algorithms to industrial standards. *Concurrency and Computation: Practice and Experience* 15, 11–12 (2003), 1117–1131.

[22] T. Jurdzinski and G. Stachowiak. 2002. Probabilistic Algorithms for the Wakeup Problem in Single-Hop Radio Networks. In *Proceedings of the 13th International Symposium on Algorithms and Computation (ISAAC)*. 535–549.

[23] J. Kabarowski, M. Kutylowski, and W. Rutkowski. 2006. Adversary Immune Size Approximation of Single-Hop Radio Networks. In *Proceedings Third International Conference on Theory and Applications of Models of Computation (TAMC)*. 148–158.

[24] M. Kardas, M. Klonowski, and D. Pajak. 2013. Energy-Efficient Leader Election Protocols for Single-Hop Radio Networks. In *Proceedings 42nd International Conference on Parallel Processing (ICPP)*. 399–408.

[25] V. King, S. Pettie, J. Saia, and M. Young. 2018. A Resource-competitive Jamming Defense. *Distributed Computing* 31 (2018), 419–439. Issue 6.

[26] M. Klonowski and D. Pajak. 2018. Brief Announcement: Broadcast in Radio Networks, Time vs. Energy Tradeoffs. In *Proceedings 37th ACM Symposium on Principles of Distributed Computing (PODC)*. 115–117. https://doi.org/10.1145/3212734.3212786

[27] Marek Klonowski and Malgorzata Sulkowska. 2016. Energy-optimal algorithms for computing aggregative functions in random networks. *Discrete Mathematics & Theoretical Computer Science* 17, 3 (2016), 285–306.

[28] E. Kushilevitz and Y. Mansour. 1998. An $\Omega(D\log(N/D))$ Lower Bound for Broadcast in Radio Networks. *SIAM J. Comput.* 27, 3 (1998), 702–712.

[29] M. Kutylowski and W. Rutkowski. 2003. Adversary Immune Leader Election in ad hoc Radio Networks. In *Proceedings 11th Annual European Symposium on Algorithms (ESA)*. 397–408. https://doi.org/10.1007/978-3-540-39658-1_37

[30] G. L. Miller, R. Peng, A. Vladu, and S. C. Xu. 2015. Improved Parallel Algorithms for Spanners and Hopsets. In *Proceedings of the 27th ACM on Symposium on Parallelism in Algorithms and Architectures (SPAA)*. 192–201.

[31] G. L. Miller, R. Peng, and S. C. Xu. 2013. Parallel graph decompositions using random shifts. In *Proceedings of the 25th Annual ACM Symposium on Parallelism in Algorithms and Architectures (SPAA)*. 196–203.

[32] K. Nakano and S. Olariu. 2000. Energy-Efficient Initialization Protocols for Single-Hop Radio Networks with No Collision Detection. *IEEE Trans. Parallel Distrib. Syst.* 11, 8 (2000), 851–863.

[33] C. Newport. 2014. Radio Network Lower Bounds Made Easy. In *Proceedings of the 28th International Symposium on Distributed Computing (DISC)*. 258–272.

[34] J. Polastre, R. Szewczyk, and D. Culler. 2005. Telos: enabling ultra-low power wireless research. In *Proceedings of the 4th International Symposium on Information Processing in Sensor Networks (IPSN)*. 364–369.

[35] L. Roditty and V. Vassilevska Williams. 2013. Fast approximation algorithms for the diameter and radius of sparse graphs. In *Proceedings 45th ACM Symposium on Theory of Computing (STOC)*. 515–524.