Randomized Broadcast in Radio Networks with Collision Detection

Mohsen Ghaffari
ghaffari@mit.edu
MIT

Bernhard Haeupler
haeupler@mit.edu
MIT

Majid Khabbazian
mkhabbazian@ualberta.ca
University of Alberta

ABSTRACT

We present a randomized distributed algorithm that in radio networks with collision detection broadcasts a single message in $O(D + \log^6 n)$ rounds, with high probability[1]. This time complexity is most interesting because of its optimal additive dependence on the network diameter D. It improves over the currently best known $O(D \log \frac{n}{D} + \log^2 n)$ algorithms, due to Czumaj and Rytter [FOCS 2003], and Kowalski and Pelc [PODC 2003]. These algorithms where designed for the model without collision detection and are optimal in that model. However, as explicitly stated by Peleg in his 2007 survey on broadcast in radio networks, it had remained an open question whether the bound can be improved with collision detection.

We also study distributed algorithms for broadcasting k messages from a single source to all nodes. This problem is a natural and important generalization of the single-message broadcast problem, but is in fact considerably more challenging and less understood. We show the following results: If the network topology is known to all nodes, then a k-message broadcast can be performed in $O(D+k \log n+\log^2 n)$ rounds, with high probability. If the topology is not known, but collision detection is available, then a k-message broadcast can be performed in $O(D + k \log n + \log^6 n)$ rounds, with high probability. The first bound is optimal and the second is optimal modulo the additive $O(\log^6 n)$ term.

Categories and Subject Descriptors

F.2.2 [**Analysis of Algorithms and Problem Complexity**]: Non-numerical Algorithms and Problems—*Computations on Discrete Structures*; G.2.2 [**Discrete Mathematics**]: Graph Theory—*Network Problems*

PODC'13, July 22–24, 2013, Montréal, Québec, Canada.

[1]We use the phrase "high probability" to indicate a probability at least $1 - \frac{1}{n^c}$, for a constant $c \geq 1$, and where n is the network size.

General Terms

Algorithms, Theory

Keywords

Wireless networks, Radio Networks, Broadcast, Collision Detection, Random Linear Network Coding

1. INTRODUCTION

The classical information dissemination problem in radio networks is the problem of broadcasting a single message to all nodes of the network (single-message broadcast). This problem and its generalizations have received extensive attention.

A characteristic of radio networks is that multiple messages that arrive at a node simultaneously interfere (collide) with one another and none of them is received successfully. Regarding whether nodes can distinguish such a collision from complete silence, the model is usually divided into two categories of with and without collision detection. Throughout studies of problems in radio networks, it has been observed that many problems can be solved faster in the model with collision detection [20]. Despite this trend, it had remained unclear whether this is also the case for broadcast or not [19]. We show that single-message broadcast can be indeed solved faster, in simply diameter plus poly-logarithmic time, if collision detection is available[1].

Broadcasting k messages from one node to all nodes is a natural and important generalization of the single-message broadcast problem. Usually, this generalization involves new and significantly different challenges, mainly because the dissemination of different messages can interfere with each other. We show how to overcome these challenges and obtain an (almost) optimal k-message broadcast algorithm.

1.1 Model and Problem Statements

We work in the *radio network model with collision detection* [4]: a synchronous network $G = (V, E)$ where in each round, each node either transmits a packet with B bits or listens. As a standard assumption, to ensure that each packet can contain a constant number of ids, we assume that $B = \Omega(\log n)$. Each node v receives a packet from its

*The research in this paper was supported by AFOSR award No. FA9550-13-1-0042, NSF grant Nos. CCF-AF-0937274, CNS-1035199, 0939370-CCF, CCF-1217506, and NSF-PURDUE-STC award 0939370-CCF.

[1]Even though our work is theoretical, we remark that most practical radio networks can detect collisions.

neighbors only if it listens in that round and exactly one of its neighbors is transmitting. If two or more neighbors of v transmit, then v only detects the collision, which is modeled as v receiving a special symbol $\top$ indicating a collision. We explain that some of our results hold even in the model *without collision detection*, where if two or more neighbors of v transmit, then v does not receive anything.

The single-message broadcast problem is defined as follows: A single *source* node has a single message of length at most $\Theta(B)$ bits and the goal is to deliver this message to all nodes in the network. The k-message single-source broadcast problem is defined similarly, with the difference that the source has k messages which need to be delivered to all other nodes. We focus on randomized solutions to these problems where we require that the message(s) are delivered to all nodes with high probability. In the unknown topology setting (which is our default setting), we assume[2] that nodes know a polynomial upper bound on n and a constant factor upper bound on diameter D. In the known topology setting, similar to [7], we assume that nodes know the whole graph.

1.2 Our Results

Our main results are as follows:

Theorem 1.1. *In radio networks with unknown topology and with collision detection, there is a randomized distributed algorithm that broadcasts a single message in $O(D + \log^6 n)$ rounds, with high probability.*

Theorem 1.2. *In radio networks with known topology (even without collision detection), there is a randomized distributed algorithm that broadcasts k messages in $O(D + k \log n + \log^2 n)$ rounds, with high probability.*

Theorem 1.3. *In radio networks with unknown topology and with collision detection, there is a randomized distributed algorithm that broadcasts k messages in $O(D + k \log n + \log^6 n)$ rounds, with high probability.*

About Theorem 1.1, we remark that prior to this work, the best known solution for single-message broadcast was the $O(D \log n/D + \log^2 n)$ algorithms presented independently by Czumaj and Rytter [6], and Kowalski and Pelc [15], for the model without collision detection. In that model, these bounds are optimal [1,17]. As Peleg points out in [19], prior to this work, it was unclear whether these upper bounds can be improved in the model with collision detection. Theorem 1.1 answers this question by showing that a better upper bound is indeed achievable. We remark that the bound of Theorem 1.1 is within an additive poly-log of the $\Omega(D + \log^2 n)$ lower bound, that follows from the $\Omega(\log^2 n)$ lower bound of [1] and the obvious lower bound of $\Omega(D)$.

About Theorems 1.2 and 1.3, we remark that these two results use random linear network coding (RLNC). Moreover, we note that even in the strong model of centralized algorithms with full topology knowledge, with collision detection, and with network coding, k-message broadcast has a lower bound of $\Omega(D + k \log n + \log^2 n)$ rounds. This lower bound follows from the $\Omega(k \log n)$ throughput-based lower bound of [10] for a k-message broadcast, the $\Omega(\log^2 n)$ lower bound of [1] for a single message broadcast, and the trivial $\Omega(D)$ lower bound. Thus, the complexity of Theorem 1.2 is optimal and the complexity of Theorem 1.3 is optimal modulo the additive $O(\log^6 n)$ term.

When looking at the issue from a practical angle, Theorem 1.1 and Theorem 1.3 have an interesting message: they show that one can replace the (expensive and not-completely-reasonable) assumption of all nodes knowing the full topology of the network, with (the considerably more reasonable and usually-available) collision detection, and still perform single or multiple broadcast tasks almost in the same time.

To achieve the above three results, we present three new technical elements, which each can be interesting on their own:

(A) The first element is a distributed construction of a Gathering-Spanning-Tree (GST) with round complexity of $O(D \log^4 n)$. GSTs were first introduced by [7] to obtain broadcast algorithms with an additive $O(D)$ diameter dependence in the known topology setting [7, 8, 18]. The only known construction of GST prior to this work was the centralized algorithm of Gasieniec et al. [7], which has step-complexity of $O(n^2)$ operations and requires the full knowledge of the graph. We use our new GST construction to prove Theorem 1.1. For this we first decompose the graph appropriately, then we construct a GST for every part in parallel and lastly we use this setup to broadcast the (single) message efficiently.

(B) The second element is a novel transmission schedule atop GST for solving multiple message broadcast problems. We contend this schedule to be the right generalization of [7] for multiple messages. Such a generalization was also attempted in [18] but its correctness was disproved [21].

(C) The third element is *backwards analysis*, an new way to analyze the progress of messages during a multi-message radio network broadcast. Backward analysis shows that a message spreads quickly even when other messages that are spread at the same time cause collisions. A priori it is not clear that information dissemination remains efficient in the presence of these collisions, which only arise in the mutli-message setting. Insights from the backwards analysis were crucial in the design of our multi-message transmission schedule and also enable us to apply the projection analysis of Haeupler [11] for analyzing random linear network coding to proof Theorem 1.2 and Theorem 1.3.

1.3 Related Work

Designing distributed broadcast algorithms for radio networks has received extensive attention, starting with the pioneering work of Bar-Yehuda, Goldreich and Itai (BGI) [2]. Here, we present a brief review of the results that directly relate to this paper.

Single-Message Broadcast: Peleg [19] provides a comprehensive survey of the known results about single-message broadcast. BGI [2] present the Decay protocol which broadcasts a single message in $O(D \log n + \log^2 n)$ rounds. The best known distributed algorithms for single-message broadcast in for the setting where the topology is unknown are the

[2]It is easy to see that the latter assumption can be removed without any change is our time-bounds, by finding a 2-approximation of D in time $O(D)$, using the beep waves tool of [9].

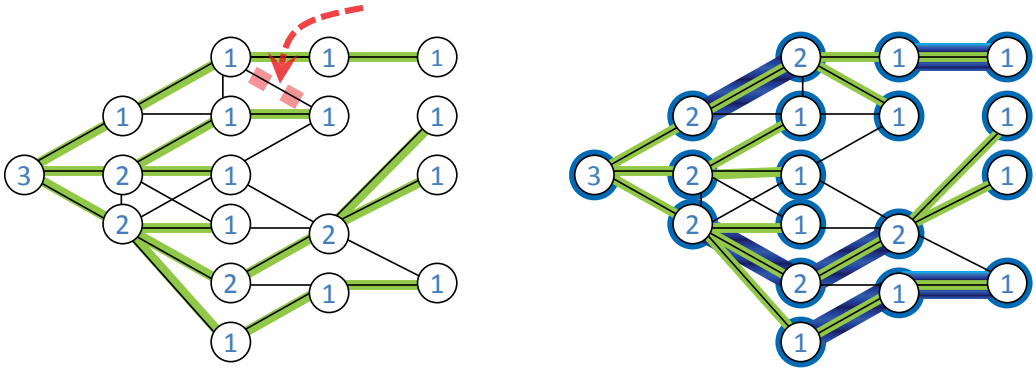

Figure 1: Gathering Spanning Tree

$O(D \log \frac{n}{D} + \log^2 n)$ algorithms presented independently by Czumaj and Rytter [6], and Kowalski and Pelc [15]. These algorithms can be viewed as clever optimizations of the Decay protocol [2]. Moreover, similar to the Decay protocol, these two algorithms are presented for the model without collision detection and are optimal in that model [1, 17]. Prior to this work, no better algorithm was known for the model with collision detection. If the topology of the network is known, then the algorithm of Gasieniec, Peleg and Xin [7] achieves the optimal $O(D + \log^2 n)$ time complexity. Kowlaski and Pelc [16] gave an explicit deterministic broadcast protocol which achieves the same time complexity.

Multi-Message Broadcast: The complexity of multi-message broadcast (with bounded packet size) is less understood. In the model without collision detection, the following results are known. The earliest work on multi-message broadcast problem is by BarYehuda et al. [3], which broadcasts k messages in $O((n + (k + D)\log n)\log \Delta)$ rounds, where Δ is the maximum node degree. Chlebus et al. [5] present a deterministic algorithm that broadcasts k messages in $O(k \log^3 n + n \log^4 n)$ rounds. The best known algorithm for multi-message broadcast is by Khabbazian and Kowalski that broadcast k messages using network coding in $O(k \log \Delta + (D + \log n)\log n \log \Delta)$ rounds [14]. Again, prior to this work, no better algorithm was known for the model with collision detection. Ghaffari et al. [10] showed a lower bound of $\Omega(k \log n)$ rounds.

2. SINGLE-MESSAGE BROADCAST

We first recall the definition of a Gathering-Spanning-Tree (GST) [7], in Section 2.1. Then, in Section 2.2, we present a distributed algorithm with time complexity $O(D \log^4 n)$ for constructing a GST, in radio networks with unknown topology (even without collision detection). In Section 2.3, we then show that this algorithm can be used to broadcast a single message in $O(D + \log^6 n)$ rounds, in radio network with unknown topology but with collision detection.

2.1 Gathering Spanning Trees (GST)

Ranked BFS: Consider a BFS tree $\mathcal{T}$ in graph G, rooted at source node s. Also, suppose that in this tree, we have assigned to each node v a level number $\ell(v)$, which is equal to the distance of v from s. We *rank* the nodes of $\mathcal{T}$ using the following inductive *ranking rule*: Each leaf of $\mathcal{T}$ gets rank 1. Then, consider node v and suppose that all children of v in $\mathcal{T}$ are already ranked. Let r be the maximum rank of these children. If v has exactly one child with rank r, then node v gets rank r. If v has two or more children with rank r, then v gets rank $r + 1$. As shown in [7], one can easily see that in each *ranked BFS*, the largest rank is at most $\lceil \log_2 n \rceil$.

Gathering Spanning Tree (GST) [7]: A ranked BFS-tree $\mathcal{T}$ is called a GST of graph G if and only if the following *collision-freeness property* is satisfied:

> In graph G, any node of rank r on level l of $\mathcal{T}$ is adjacent to *at most one* node of rank r at level $l-1$ of $\mathcal{T}$. In other words, if there are two nodes u_1 and u_2 with rank r on level l of $\mathcal{T}$, and their parents in $\mathcal{T}$ are respectively v_1 and $v_2 \neq v_1$ (on level $l-1$ of $\mathcal{T}$), and v_1 and v_2 have rank r as well, then there is no edge between v_1 and u_2 or between v_2 and u_1.

Fast Stretches in a GST: In a GST $\mathcal{T}$, for each path in $\mathcal{T}$ from a node v to a node u that is a descendant of v in $\mathcal{T}$, we call this path a *fast stretch* if all the nodes on the path have the same rank. Note that a fast stretch might be just a single node.

Distributed GST: In a distributed construction of a GST, each node u must learn the following four items[3]: (1) its level $\ell(u)$, (2) its own rank $r(u)$, (3) the id of its parent v, and (4) the rank of its parent $r(v)$.

Figure 1 presents an example of a GST. The black edges present the graph G and the thicker green edges present a rank labeled BFS tree $\mathcal{T}$ of G. On the left side, we see a rank-labeled BFS tree, but this tree is not a GST because of the violation of the collision-freeness property indicated by the red dashed arrow. On the right side, we see another rank-labeled BFS of the same graph G, which is a GST. In this GST, the green edges that are coated with wide blue lines indicate the fast stretches. Each node that is not incident

[3]From (2) and (4), any node u can easily infers whether it is the first node in a fast stretch and whether its parent is in that stretch as well.

on any of these blue-coated edges forms a trivial fast-stretch made of just a single node.

Broadcast Atop GST: In [7] Gasieniec et al. presented an algorithm to broadcast a single message in $O(D+\log^2 n)$ rounds, atop a GST. A high-level explanation is as follows: with a careful timing, the message can be sent through the fast stretches without any collision. That is, we can (almost simultaneously) send the message through different stretches such that in each fast stretch, the message gets broadcast from the start of the stretch to the end of the stretch in a time asymptotically equivalent to the length of the stretch. On the other hand, since the largest rank in the tree $\mathcal{T}$ is at most $\lceil \log_2 n \rceil$ and because on each path from the source to any node v, the ranks are non-increasing, we get that the path from the source to each node v is made of at most $\lceil \log_2 n \rceil$ distinct fast stretches. By using the *decay protocol*[4] [2] on each of the (at most) $\lceil \log_2 n \rceil$ connections between the fast stretches, we get a broadcast algorithm with time complexity $O(D + \log^2 n)$. We refer the reader to [7] for the details of this broadcast algorithm. We remark that we will use [7] simply as a black-box that broadcasts a single-message in time $O(D+\log^2 n)$ on top of the GSTs we construct.

2.2 Distributed GST Construction

In this subsection, we present the following result:

Theorem 2.1. *In the radio networks (even without collision detection), there exists a distributed GST construction algorithm with time complexity $O(D \log^4 n)$ rounds.*

We show a GST construction with round-complexity of $O(D \log^5 n)$ in Sections 2.2.1 to 2.2.3. We later improve this to $O(D \log^4 n)$ rounds, in Section 2.2.4.

2.2.1 Black-Box Tools

Before starting the construction, we first present two black-box tools which we use in our construction.

***Decay Protocol* [2]:** Rounds are divided into phases of $\log n$ rounds, and in the i^{th} round of each phase, each node v transmits with probability 2^{-i} (if it has a message for transmission).

Lemma 2.2. ***(Bar-Yehuda et al. [2])*** *For each node v, if at least one neighbor of v has a message for transmission, then in each phase of decay, node v receives at least one message with probability at least $\frac{1}{8}$. Moreover, in $\Theta(\log n)$ such phases, v receives at least one message, with high probability.*

***Recruiting Protocol*:** This tool can be abstracted by the guarantees that it provides, which we present in Lemma 2.3.

Lemma 2.3. *Consider a bipartite graph $\mathcal{H}$ where nodes on one side are called* red *and nodes on the other side are called* blue. *The recruiting protocol achieves the following three properties, w.h.p., in $\Theta(\log^3 n)$ rounds:* ***(a)*** *for each blue node u, we assign an adjacent red node of v to u. In this case, we say u is* recruited *by v (then called parent of u),* ***(b)*** *each red node v knows whether it recruited zero, one, or at least two blue nodes,* ***(c)*** *each recruited blue node u knows whether its parent v recruited zero, one, or at least two blue nodes.*

[4]Decay is a standard technique for coping with collisions in radio networks. We present a short recap on it in Section 2.2.1.

Next, we present the *recruiting protocol*. We defer the proof of Lemma 2.3 to the full version.

Recruiting Protocol: The protocol consists of $\Theta(\log^2 n)$ recruiting iterations, each having $2 + \Theta(\log n)$ rounds as follows:

- In the first round of the j^{th} recruiting iteration, each red node transmits its id with probability $2^{-\lceil \frac{j}{\Theta(\log n)} \rceil}$.
- Then, we run a phase of Decay protocol, consisting of $\Theta(\log n)$ rounds, from the side of blue node. In this phase, each not-recruited blue node u that received a message of a red node v tries to transmit $u.id$ and $v.id$ (together in one packet).
- After that, the red nodes repeat the exact transmissions of the first round of this iteration, with new contents as follows: (1) if in the previous Decay phase, a red node v received its own id from exactly one blue node u, then v broadcasts $v.id$, (2) if the red node v received its own id from two or more blue nodes, then v broadcasts a special message Σ. (3) Otherwise, v transmits an empty message.
- Next, if a blue node u received its own id or the special message Σ in the last round, then we say u is recruited by red node v, where v is the red node such that u received $v.id$ in the first round. Note that each red node v knows whether it recruited zero, one or at least two blue nodes.

2.2.2 GST Construction Outline

We first construct a BFS-tree of G and assign to each node v a level $\ell(v)$ that is equal to the distance of v from the source. This can be done in $O(D \log^2 n)$ rounds, as follows: Rounds are divided into D epochs each consisting of $\Theta(\log n)$ phases of decay (thus, each epoch has $\Theta(\log^2 n)$ rounds). In each epoch, a node v participates in the decays iff it is the source or it has received a message by the end of the last epoch. During these rounds, each node relays the first message it received. The epoch in which a node v receives a message for the first time determines the BFS level $\ell(v)$ of node v.

Now that we have a BFS-tree, we build the GST on top of this BFS layering, level by level, and from the largest level towards the source. For this, the problem boils down to the following scenario: Consider level l of layering and assume that the GST is already built for levels $j \geq l$. Consider the bipartite graph H induced on the nodes of level $l-1$ and level l, ignoring the (possible) edges inside each level. The core of the problem is to design an algorithm to construct the part of GST between levels $l-1$ and l, i.e., the part that is H.

Let us call the nodes on level $l-1$ *red nodes*, and the nodes on level l *blue nodes*. To construct the part of GST that is in H, we assign a red *parent* v to each blue node u, from amongst the red neighbors of u in H. In this case, v is known as u's *parent* and u is a *child* of v. This assignment,

along with the rankings of blue nodes, leads to a ranking for the red nodes. More precisely, let v be a red node and let i be the maximum rank of blue node children of v in the assignment. Node v gets rank i if it has only one child with rank i, and v gets rank $i+1$ if it has more than one child with rank i.

To have a GST, these assignments should be *collision-free*. That is, if there exist blue nodes u_1 and u_2 and their respective parents v_1 and v_2, all four with rank i, then H must have no edge between v_1 and u_2, or between v_2 and u_1. We refer to the problem of finding such an assignment as the *Bipartite Assignment Problem*.

More precisely, in the *Bipartite Assignment Problem*, we should achieve the following 6 properties: (1) For each blue node u, we should assign a red neighbor v as its parent, (2) we should rank the red nodes as follows: for each red node v, suppose i is the maximum rank of the children of v. Then, v should get rank i if v has exactly one blue child of rank i, and v should receive rank of $i+1$ if v has two or more blue children of rank i, (3) the assignment should be *collision-free*, (4) each red node must know its rank and (5) each blue node u should know the id of its parent and (6) each blue node u should know the rank of its parent.

The *Bipartite Assignment Problem* is the core of the GST construction and once we have a solution for it, repeating the solution level by level from the largest level to source constructs a GST. In the next subsection, we explain how to solve this problem in $O(\log^5 n)$ rounds.

2.2.3 The Bipartite Assignment Algortihm

Consider bipartite graph H as explained. We solve the bipartite assignment problem (defined above) in H in a rank by rank basis, starting with the largest possible rank $\lceil \log n \rceil$ (of blue nodes), and going down in ranks until reaching rank 1. We spend $\Theta(\log^4 n)$ rounds on each rank. Let us consider the case of a bipartite assignment for blue nodes of rank i in graph H, assuming that ranks greater than i are already solved.

We first identify the red neighbors of the blue nodes with rank i. This is done by using $\Theta(\log n)$ phases of Decay where blue nodes of rank i transmit. This identifies the desired red nodes as every such red node receives at least one message with high probability and no other red node receives any message. From now on, throughout the procedure for rank i, only these red nodes are active. Now the algorithm is divided into $\Theta(\log n)$ epochs. Each epoch consists of three stages as follows:

Stage I: Call a blue node u of rank i a *loner* if u has exactly one active red neighbor. We first detect the loner blue nodes. For this, in one round, each active red node transmits a message. Only loner blue nodes receive a message and each other blue node receives a collision. We then use $\Theta(\log n)$ phases of Decay, where each blue loner tries transmitting. This with high probability informs all red nodes that are connected to at least one loner blue node. We call these red nodes *loner-parents*.

Stage II: This stage is divided into three parts, and each red node is active in only one of the parts. Loner-parents, which we identified in the stage I, are active only in part 1. Each other active red node randomly and uniformly decides to be either *brisk* or *lazy*, which respectively mean it is active in part 2 or in part 3. These parts are as follows:

- **Part 1.** Loner-parents use a *recruiting protocol*. During this recruiting protocol, each blue neighbor of each red loner-parent get recruited with high probability. These assignments are *permanent*. All the blue nodes that are recruited become inactive for the rest of the assignment problem.
- **Part 2.** Brisk red nodes run a Recruiting protocol. Then, each blue node that is not the only recruited child of its parent considers its parent as its *permanent* GST parent and becomes inactive permanently (for the GST construction). The other recruited blue nodes become inactive only for the remainder of this epoch, but these assignments are *temporary* and the related nodes restart in the next epoch, ignoring their temporary assignments.
- **Part 3.** We repeat the procedure of part 2, but this time with lazy red nodes and with the active blue nodes that did not get recruited in parts 1 or 2.

Stage III: Let us say that a red node is *marked* if it was a loner-parent or if it recruited zero or strictly more than one blue nodes in parts 2 or 3. Each marked red node becomes inactive after this epoch. Thus, the only red nodes that remain active after this epoch are those that do not have any loner neighbor and recruited exactly one child in part 2 or 3 of the stage II. Each marked red node knows whether it recruited zero, one, or at least two children (in stage II). We use this knowledge to rank these marked red nodes giving them rank of i if they recruited exactly one blue child and rank of $i+1$ if they recruited more than one blue child. Blue children of marked red nodes also know that their parents of marked and they can also compute the rank of their parents (refer to property (c) of Lemma 2.3).

Before inactivating the marked red nodes, we do one simple thing: marked red nodes run $\Theta(\log n)$ phases of Decay sending their id and rank. Each blue node of any rank strictly lower than i that receives a red node id considers the first red node that it heard from as its permanent GST parent, records the id and rank of that red parent, and then, becomes inactive for the rest of the assignment problem.

After running the bipartite assignment algorithm for all the ranks, if a red node v has no child, then v is a leaf and in the GST, v gets rank 1.

Figure 2 shows an example of assignments during an epoch (the first epoch). The green arrows in the leftmost part indicate the loner blue nodes at the start of the epoch. The loner parent red nodes are indicated by a number 1 next to them, meaning they are active in part 1. Brisk and lazy red nodes are respectively indicated by numbers 2 and 3, next to them. The smaller nodes present the (temporarily or permanently) deactivated nodes. The green dashed lines show the permanent assignments and the (thicker) orange dashed lines show the temporary assignments. After the end of epoch, nodes with temporary assignment are re-activated.

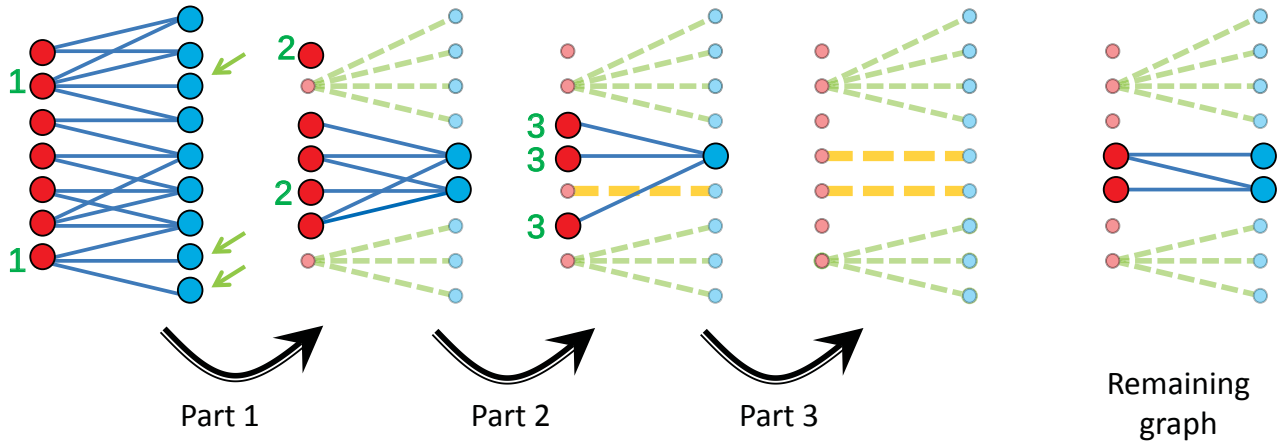

Figure 2: Parts 1, 2, and 3 of the stage II of the first epoch of the assignment algorithm, and the graph remaining after the first epoch

The graph remaining after the first epoch is presented on the right side of the Figure 2, by solid blue lines.

Analysis: In Lemma 2.4, we prove that in each of the $\Theta(\log n)$ epochs except the first one, we reduce the size of the assignment problem for rank i by at least a constant factor, with at least a positive constant probability. Here, by size of the assignment problem, we mean the number of the active red nodes with a blue neighbor of rank i. A standard Chernoff bound then shows that in $\Theta(\log n)$ epochs, each blue node of rank i has a parent. It is clear that the parents are ranked according to the ranking rules of GST and nodes know their own rank, the id of their parents, and the rank of their parents. We show in Lemma 2.5 that with high probability, the assignment is collision-free.

Lemma 2.4. *In each epoch $j' \leq 2$, with a probability at least $1/7$, the number of remaining active red nodes for the next epoch goes down with a factor at least $8/7$.*

Proof. Consider epoch $j' \geq 2$ and let η be the number of active red nodes at the start of this epoch. We show that the expected number of red nodes that remain active at the end of this epoch is at most $\frac{3\eta}{4}$. This is enough for the proof because with this, and by Markov's inequality, we get that with probability at least $1/7$, the number of active remaining red nodes at the end of this epoch is at most $\frac{7\eta}{8}$.

Each red node remains active after epoch j' only if it gets a temporary assignment, i.e., if it is not a loner-parent and it recruits exactly one child during parts 2 and 3 of Stage II. Thus, the expected number of red nodes that remain active is at most equal to the expected of number of brisk red nodes (those that act in part 2) plus the number of blue nodes that are active in part 3. The expected number of brisk red nodes is at most $\frac{\eta}{2}$. To complete the proof, we show that the expected number of blue nodes that remain active for part 3 (after the assignments of part 2) is at most $\frac{\eta}{4}$.

After each epoch, the only red nodes that remain active are those that have a temporary assignment, i.e., those that each have recruited exactly one child and that child is not a loner. Moreover, the only active remaining blue nodes are those blue nodes temporarily matched to the remaining red nodes. Thus, after each epoch, the number of remaining active red nodes and the number of remaining active blue nodes are equal. From this, we can conclude that since $j' \geq 2$, at the start of epoch j', the number of active blue nodes is at most η.

Using Lemma 2.3, we infer that in part 1 of stage II, each blue neighbor of a loner-parent is w.h.p. recruited by a red loner-parent. Thus, in particular, each loner is recruited with high probability. Hence, at the start of part 2 of stage II, each remaining active blue node has at least 2 red node neighbors. Since each non-loner-parent red node is active in part 2 of stage II with probability $1/2$, and because in part 2 of stage II each active blue node that has an active red node neighbor gets recruited with high probability (by Lemma 2.3), each blue node remains active after part 2 of stage II with probability at most $1/4$. We know that because of the previous paragraph, the number of active remaining blue nodes at the start of part 2 of stage II is at most η. Hence, the expected number of blue nodes remaining active after part 2 is at most $\frac{\eta}{4}$. This completes the proof of the lemma. □

Lemma 2.5. *With high probability, the bipartite assignment algorithm creates a collision-free assignment.*

Proof. We show that if there exist blue nodes u_1 and u_2 ($u_1 \neq u_2$) and their respective red parents v_1 and v_2 ($v_1 \neq v_2$), all four with rank i, then with high probability, H must not have any edge between u_2 and v_1, or between u_1 and v_2. For the sake of contradiction, and without loss of generality, suppose that there is an edge between u_2 and v_1. Since v_2 and u_2 have rank i, blue node u_2 must have been a loner when v_2 recruited it. Thus, v_2 recruited u_2 after v_1 became inactive. Hence, in the epoch that v_1 recruited u_1, u_2 was active. Therefore, using Lemma 2.3 we get that in the part 1 of the epoch in which v_1 recruited u_1, u_2 must have been w.h.p. recruited by either v_1 or some other loner-parent. Since $v_2 \neq v_1$ recruited u_2, we get that v_2 must have been that other loner parent. This means that at that time, v_2 had a loner child ($\neq u_2$) and thus, v_2 has recruited more than one child of rank i. This means that v_2 must have had rank $i+1$ which contradicts with the assumption that v_2 has rank i. □

2.2.4 Pipelining the GST Construction

Note that in the above algorithm, and in assignment problem between levels $l-1$ and l, once we are done with the assignment problem of ranks i and $i-1$, nodes of level $l-1$ that receive rank i are already determined, i.e., no other node in level $l-1$ will receive rank i. Thus, we can solve the two problems of rank $i-2$ assignment between levels $l-1$ and l and rank i assignments between levels $l-2$ and $l-1$, essentially simultaneously, by interleaving them in even and odd rounds. Using the same idea, it is easy to see that one can pipe-line the assignment problems of different ranks between different levels. Then, the assignment problem between levels $l-1$ and l starts after $\Theta((D-l)\log^4 n)$ rounds. Thus, the assignment problem of largest possible rank between levels 0 and 1 starts after $\Theta(D\log^4 n)$ rounds. The largest rank is at most $\lceil \log n \rceil$. Since each rank takes $\Theta(\log^4 n)$ rounds, the whole GST construction problem finishes after $\Theta(D\log^4 n)$ rounds.

2.3 Unknown Topology Single-Message Broadcast in $O(D+\log^6 n)$ Rounds

Theorem 1.1. *(restated) In radio networks with unknown topology and with collision detection, there is a randomized distributed algorithm that broadcasts a single message in $O(D+\log^6 n)$ rounds, with high probability.*

Proof. We first use a wave of collisions to get a BFS layering in time D. That is, the source transmits in all rounds $[1, D]$, and each node v transmits in all rounds $[r, D]$ where r is such that v receives a message or a collision in round $r-1$. For each node v, the round $r-1$ in which v receives the first message or collision determines distance of v from the source.

Having this BFS layering, we decompose the graph into $O(\log^4 n)$ rings, each consisting of $D' = D/\log^4 n$ consecutive layers of the BFS layering.

Then, we compute a gathering spanning tree for each of the rings in $O(D'\log^4 n) = O(D)$ rounds. Note that computation of a GST for each ring only depends on D' which is the number of BFS layers that the ring contains, and that given the BFS-layering, the computation of the GSTs of all rings is performed in parallel.

Having these GSTs, broadcasting the message inside each ring takes $O(D' + \log^2 n)$ rounds, using [7]. Finally, we use $O(\log^2 n)$ rounds of decay protocol [2] to propagate the message from the outer boundary of one ring to the inner boundary of the next ring. Since there are $O(\log^4 n)$ rings, the whole broadcast takes $\big(O(D' + \log^2 n) + O(\log^2 n)\big) \cdot O(\log^4 n) = O(D + \log^6 n)$ rounds. □

3. MUTLI-MESSAGE BROADCAST

While broadcasting one message in the known topology setting is well understood, having a tight bound $\Theta(D + \log^2 n)$ [7], achieving the optimal broadcast time for multiple messages is non-trivial even for networks with known topology. We show the following:

Theorem 1.2. *(restated) In radio network with known topology (even without collision detection), there is a randomized distributed algorithm that broadcasts k messages in $O(D + k\log n + \log^2 n)$ rounds, with high probability.*

We remark that this bound is optimal, given the $\Omega(k \log n)$ lower bound of [10] for k-message broadcast, the $\Omega(\log^2 n)$ lower bound of [1] for single message broadcast, and the trivial $\Omega(D)$ lower bound.

Furthermore, it is easy to combine the known topology algorithm of Theorem 1.2 with the ideas of the proof of Theorem 1.1 and the standard technique of grouping messages and pipe-lining the groups, to prove Theorem 1.3. We defer the details to the full version.

3.1 Challenges in Broadcasting Multiple Messages

Given the known transmission schedules for broadcasting a single message in optimal $O(D + \log^2 n)$ time on top of a GST, it is intriguing to try to use the same transmission schedule to solve the multi-message broadcast problem. However, since we cannot disjoin the spreading process of different messages this approach faces two challenges:

Firstly, when a node v has already learned multiple messages and is triggered by the schedule to transmit, v needs to decide which message to forward. Choosing one message over the others can slow down the progress of those other messages.

Fortunately, random linear network coding (RLNC) [13] provides a general technique for making such decisions: Instead of deciding on one specific message whenever a node is triggered to send it, node v transmits a random linear combination of all packets it has received. It has been shown that this is the universal optimal strategy, that is, this succeeds with high probability as soon as it was possible (in hindsight) to send k messages to each of the receivers [12]. We think that network coding might be in fact necessary for obtaining the optimal throughput performance that we achieve. Our multi message broadcast utilizes RLNC and uses recent advances in analyzing RLNC performance [11] for the proofs. Even though RLNC and its analysis need to be carefully tailored to broadcast in radio networks, this gives us a good plan to remedy the first issue.

The second issue is subtle but turns out to be more problematic: When proving progress of messages all known single-message schedules and their analysis (e.g., those of [7]) rely crucially on nodes that do not have the (single) message to remain silent and cause no collisions. In a multi-message setting it becomes a necessity that we make progress for a message while allowing other nodes that do not have this message to transmit to make progress on other messages.

Trying to understand this problem prompted us to define the property *multi-message viable (MMV)*: We say a transmission schedule broadcasts one message in a MMV way if it broadcasts one message while nodes that do not have the message but are scheduled to transmit are allowed to send noise. Intuitively, this captures the viewpoint where we focus on one message and the transmissions of other messages are regarded as noise, possibly harming the progress of the message in consideration. We later see that a property very close to MMV is exactly what is enough to prove that a schedule works well with RLNC.

Unfortunately proving that a schedule is MMV is not straightforward and it is a priori not clear whether the already existing schedules are MMV. The easiest example to see this is the simple Decay algorithm of [2]: in Decay, if a node is scheduled to transmit but it does not have the message, then this node remain silent. Decay broadcasts

a single message in $O(D \log n + \log^2 n)$ rounds [2]. This follows almost directly from a simple progress lemma which shows that in $O(\log n)$ rounds of Decay a node gets informed (receives the single message) with constant probability if at least one of its neighbors is informed. However, if the nodes that do not have the message are allowed to send noise when the schedule prompts them to transmit, then the key progress lemma of [2] does not hold anymore. Surprisingly, even though this lemma breaks, it is still true that one message is spread quickly in this case (when uninformed nodes are noising), meaning that Decay broadcasts in time $O(D \log n + \log^2 n)$ rounds in a MMV way:

Lemma 3.1. *Consider the transmission schedule of Decay: for each round r, for each node v at distance l_v from source, if $r \equiv l_v + 1 \mod 3$, then v is prompted to transmit with probability $2^{-((r-l_v-1)/3 \mod \lceil \log n \rceil)}$. Also, suppose that each node that is prompted to transmit but does not have the message sends "noise". Then, all nodes receive the message by round $O(D \log n + \log^2 n)$, with high probability.*

To prove this lemma, we need to go away from the analysis approach in [2] that chooses a shortest path from source s to node v and shows that the broadcast message makes fast progresses along this path when moving forwards in time. Instead we use what we call *backwards analysis*: in a nutshell, we move backwards in time and find a sequence of collision-free transmissions from s to v, where hops of this sequence are unraveled backwards (from v to s). Meanwhile unraveling this sequence, each of these transmission can be the broadcast message or just "noise", depending on whether the sender has received the broadcast message or not. Once we reach s, it means the transmissions in the sequence indeed where the broadcast message. We defer the details of the proof to the proof of Lemma 3.1 in the full version.

Unfortunately, in contrast to the simple Decay schedule, the schedule of [7] appears to be not MMV. In Section 3.2, we present a new transmission schedule for GSTs that is MMV. In the proof of Theorem 1.2, we again use our backwards analysis to show that this new schedule is in fact MMV, while also proving that our multi-message algorithm which combines RLNC with this schedule broadcasts k messages in optimal time $O(D + k \log n + \log^2 n)$.

3.2 A Multi-Message Transmission Schedule Atop GST

In this section, we present our transmission schedule for GSTs. Later we use this schedule along with random linear network coding to achieve our optimal multi-message algorithm.

Suppose we have a GST T for graph G. For each node u, let l_u be the distance of u from source s in graph G (that is, the BFS level of u). Also, let r_u be the rank of u in GST T. We first construct a virtual directed graph G', from graph G, as follows: we add a directed edge from every node u with rank r that is the first node of a fast stretch to every descendant of u in T that has rank r (thus, to all nodes in that fast stretch). We call this a *fast* edge. We use the notation d_u to denote the length of the shortest (directed) path from s to u in G', and we call this *virtual-distance*. Given graph G, GST T, and the respective virtual graph G' (and the related virtual-distances), our schedule is defined as follows:

Multi-Message Viable GST Schedule: In round t, each node u at BFS-level l of G with rank r in GST T and virtual-distance d in the virtual graph G' does as follows: (a) if $t \equiv 2(l+3r) \pmod{6\lceil \log_2 n \rceil}$, then u transmits; (b) if $t \equiv 1+2d \pmod 6$), then u transmits with probability $2^{-((t-1-2d)/6 \mod \lceil \log_2 n \rceil)}$; otherwise, u listens.

Note that the case (a) only happens in even rounds and case (b) happens only in odd rounds. As in [7], we call the transmissions triggered by case (a) *fast transmissions* and the transmissions triggered by case (b) *slow transmissions*.

We remark that this schedule uses fast transmissions exactly as in [7, 18] to pipeline the messages along the fast stretches of GST. We see in Lemma 3.3 that these fast transmissions are collision-free. The crucial difference with the schedule in [7,18] lies in defining the slow transmissions with respect to the virtual-distance in graph G' (instead of levels in G). This change results in slow transmissions not trying to push messages away from the source themselves, but instead trying to push messages towards entry points of the fast stretches (even if this leads to the message going back towards the source). While this modification seems minor, it is crucial for allowing the *backwards analysis* technique to show that the new schedule is efficient and MMV.

Lemma 3.2. *In virtual graph G', for each node u, we have $d_u \leq 2\lceil \log_2 n \rceil$.*

Lemma 3.3. *There are no collisions between any two fast transmissions.*

Proposition 3.4. *If node u with level l is the beginning of a fast stretch in GST $\mathcal{T}$ and u sends a message at time t in a fast transmission round, then any node v with level $l' > l$ on the same fast stretch receives this message by time $t' = t + 2(l' - l)$.*

Lemma 3.5. *For any node u with virtual-distance d_u, if there is at least one node v connected to u in G with virtual-distance $d_v = d_u - 1$, then during each interval of $6\lceil \log_2 n \rceil$ rounds, with probability at least $\frac{1}{8}$, node u receives a message from one node with virtual-distance $d_u - 1$.*

3.3 Optimal Multi-Message Broadcast Algorithm

We achieve our optimal multi-message broadcast algorithm by combining random linear network coding (RLNC) with the Multi-Message GST Schedule that we presented in Section 3.2. We first recall on the exact working of RLNC [12, 13] and then present our multi-message broadcast algorithm.

In RLNC, the k messages are regarded as bit-vectors $\vec{m}_1$, ..., $\vec{m}_k \in \mathbb{F}_2^l$ over $\mathbb{F}_2$, the finite field of order two. Each network coded packet p consists of a linear combination of messages, that is, the vector $\sum_{i=1}^{k} \alpha_i \vec{m}_i \in \mathbb{F}_2^l$. We remark that the standard implementation of RLNC requires that the coefficient vector $\vec{\alpha} = (\alpha_1, \ldots, \alpha_k) \in \mathbb{F}_2^k$ is transmitted with each message. Doing this would increase the packet size to k bits which could be too large. We note that in the known topology setting there is no need for actually including the coefficient vectors in the packets because using the topology knowledge, all nodes can compute the coefficients offline in a consistent manner. In the unknown topology scenario, using generations, that is, dividing messages into groups of size $\log n$ and then doing network coding only inside each

group keeps the coefficient overhead to $O(\log n)$ bits. We defer the details for this to the full version.

Because of linearity, a node that has a number of these packets can create a packet of this form for any coefficient combination that is spanned by the coefficient vectors of the packets that it has received by that time. Also, if a node has a set of k packets with linearly independent coefficient vectors, then this node can reconstruct all the k messages using Gaussian elimination. In RLNC, every node u stores all its received packets to maintain the subspace that is spanned by them. Whenever u decides to generate a *new coded packet*, it chooses a random coefficient vector from this subspace by taking a random linear combination of the packets stored. Once the subspace spanned by the coefficient vectors in packets received by u is the full space $\mathbb{F}_2^k$, then u decodes and reconstructs all the messages.

Multi-Message Broadcast Algorithm: Whenever in MMV schedule of Section 3.2, a node u is prompted to transmit, u transmits a packet determined as follows: (a) if this is a slow transmission, or if this is a fast transmission and u is the first node on a fast stretch, then u transmits a new coded packet, that is, a packet that is created using network coding by combining the messages u has received earlier, (b) if this is a fast transmission but node u is an intermediate node in a fast stretch, then u simply relays the packet it received in the previous fast transmission round (if any).

3.4 Analysis of the Multi-Message Broadcast Algorithm

In the analysis of our Multi-Message Broadcast Algorithm we combine our new backwards analysis with a carefully designed potential function and the projection analysis from [11] to show that using the schedule from Section 3.2 together with random linear network coding achieves an optimal multi-message broadcast.

The following definitions are taken from [11]:

Definition 3.6 ([11, Definition 4.1]). *A node v is* infected *by a coefficient vector $\vec{\mu} \in \mathbb{F}_2^k$ if v has received a packet with a coefficient vector $\vec{c} \in \mathbb{F}_2^k$ that is not orthogonal to μ, that is, $\langle \vec{\mu}, \vec{c} \rangle \neq 0$.*

Proposition 3.7 ([11, Lemma 4.2]). *If a node v is infected by a coefficient vector $\vec{\mu}$ and after that, a node u receives a packet from node v, then u gets infected by $\vec{\mu}$ with probability at least $1/2$. Furthermore, if a node v is infected by all the 2^k coefficient vectors in $\mathbb{F}_2^k$, then v can decode all the k messages.*

We now present our analysis for the *multi-message broadcast algorithm* presented in Section 3.3.

Proof of Theorem 1.2. For a large enough constant λ let $T = \lambda(D + k\lceil \log_2 n \rceil + 2\lceil \log_2 n \rceil^2)$. We claim that for any node v and any fixed non-zero vector $\vec{\mu} \in \mathbb{F}_2^k$, the probability that node v is not infected by $\vec{\mu}$ in T rounds is at most $2^{-(k+2\log n)}$. Using this claim, we conclude via a union bound over all the 2^k coefficient vectors in $\mathbb{F}_2^k$ that by round T, with high probability, v is infected by all the coefficient vectors in $\mathbb{F}_2^k$. That is, by round T, v can decode all the k messages. Using another union bound over all the choices of node v then shows that by round T, with high probability, all nodes have received all the messages.

Fix a node v and a non-zero vector $\vec{\mu} \in \mathbb{F}_2^k$. To prove the claim, we use *backwards analysis* to view the process of infection spreading of vector $\vec{\mu}$. In this method, we go back in time, from round T to round 1, and we find a sequence of collision-free transmissions from source node s to node v such that all the transmissions in this chain are successful *with respect to* vector $\vec{\mu}$. Since we are moving back in time, we find this sequence starting from v and going backwards till reaching s. More precisely, for each t, we say node u is *transmission-connected* to v by backwards time t" if there is a sequence of transmissions $u = w_1, w_2, \ldots w_\ell = v$ where for each $i \in [1, \ell - 1]$, w_i transmits in a round $r_i \in [T - t, T]$, we have $r_i < r_{i+1}$, and in round r_i, w_{i+1} receives a message from w_i. Let S_t be the set of all nodes that are transmission-connected to v by backwards time t. Moreover, we then define the potential of v with respect to vector $\vec{\mu}$ at backwards time t to be $\Phi_{\vec{\mu}}(t) = \min_{u \in S_t} d_u \lceil \log_2 n \rceil + l_u$. Note that $\Phi_{\vec{\mu}}(0) \leq 2\lceil \log_2 n \rceil^2 + D$. To prove the claim, we show that with probability at least $1 - 2^{-(k+2\log n)}$, we have $\Phi_{\vec{\mu}}(T) = 0$. For this, moving backwards in time, we show that in every $8\lceil \log_2 n \rceil$ interval of consecutive rounds, this potential decreases with probability at least $\frac{1}{16}$ by at least $\lceil \log_2 n \rceil - 1$. For a backwards time t, let node u be the node in S_t that minimizes the potential of v. The proof is now divided into two cases as follows:

Case (A): Suppose u has at least one G-neighbor that has a lower virtual-distance. In this case, Lemma 3.5 guarantees that with probability at least $\frac{1}{8}$ during the rounds in $[T - t - 6\lceil \log_2 n \rceil, T - t]$, there is a collision-free transmission from a node u' with $d_{u'} = d_u - 1$ to u, and is successful with respect to $\vec{\mu}$, with probability $1/2$. Since u' and u are neighbors their levels l_u and $l_{u'}$ differ at most by one, thus a successful transmission decreases the potential by at least $(d_u \lceil \log_2 n \rceil + l_u) - (d_{u'} \lceil \log_2 n \rceil + l_{u'}) = (d_u - d_{u'}) \lceil \log_2 n \rceil - (l_u - l_{u'}) \geq \lceil \log_2 n \rceil - 1$. Thus, if u has a neighbor with a virtual-distance lower than d_u then with probability at least $\frac{1}{16}$ the potential decreases by at least $\lceil \log_2 n \rceil - 1$ within any $8\lceil \log_2 n \rceil$ rounds when moving backwards in time.

Case (B): Suppose u does not have a G-neighbor with a lower virtual-distance. Note that this can only happen if $u = s$ or if there is one directed edge in G' representing a fast stretch, originating from a node u' one level below u in G' and going into u. First observe that the starting node of any fast stretch initiates a "transmission wave" every $6\lceil \log_2 n \rceil$ rounds by creating a new coded packet and sending it as a fast transmission. This packet gets then pipe-lined through the fast stretch with one progress every fast transmission round (that is, once in every two rounds) until it reaches the end of the stretch. Thus, for any node on a fast stretch, there is a new wave arriving every $6\lceil \log_2 n \rceil$ rounds. Moreover, each of these waves is successful with respect to $\vec{\mu}$ with probability at least $1/2$. Thus, at a time $t' \in [T - t - 6\lceil \log_2 n \rceil, T - t]$, a fast transmission wave arrives in u, and with probability $1/2$ leads to an extended sequence of collision-free transmissions that are successful with respect to $\vec{\mu}$. In particular, if the wave originated from u' during the rounds $[T - t' - 2\lceil \log_2 n \rceil, T - t']$, then there is a sequence of transmission from u' to v in round interval $[T - t - 8\lceil \log_2 n \rceil, T - t]$, and otherwise the wave propagated for $\lceil \log_2 n \rceil$ steps and there is a node u'' between u' and u on

the fast stretch with a sequence of transmission to v starting at time $T - t - 8\lceil \log_2 n \rceil$. Thus, in both cases, the potential drops by at least $\lceil \log_2 n \rceil - 1$. In the first case the potential drop comes from the fact that $d_{u'} = d_u - 1$ and $l_{u'} < l_u$, while in the second case we have $d_{u''} \leq d_{u'} + 1 = d_u$ and $l_{u''} \leq l_u - \lceil \log_2 n \rceil$.

The above argument shows that when moving backwards in time, in every $8\lceil \log_2 n \rceil$ consecutive rounds, with probability at least $\frac{1}{16}$, the potential of v decreases by at least $\lceil \log_2 n \rceil - 1 > \lceil \log_2 n \rceil / 2$, until reaching zero. When the potential reaches zero, it means that there is a sequence of successful and collision-free transmission from s to v. Hence, the expected time for such a sequence to appear is thus a constant times the initial potential of v, $\Phi_{\vec{\mu}}(0) \leq 2\lceil \log_2 n \rceil^2 + D$. A Chernoff bound furthermore shows that the probability of not finding such a sequence is exponentially concentrated around this mean. In particular, after $T = \lambda(D + k\lceil \log_2 n \rceil + 2\lceil \log_2 n \rceil)$ rounds, we expect at least $\lambda'(2D/\lceil \log_2 n \rceil + 4\lceil \log_2 n \rceil + k)$ sets of $8\lceil \log_2 n \rceil$ consecutive rounds in which the potential of v drops at least by $\lceil \log_2 n \rceil / 2$, for a constant λ'. Furthermore, the probability that there are less than $(2D/\lceil \log_2 n \rceil + 4\lceil \log_2 n \rceil$ such rounds is exponentially small in the expectation, that is, at most $2^{-(2\lceil \log_2 n \rceil + k)}$. This completes the proof of Theorem 1.2 □

4. REFERENCES

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

[2] Bar-Yehuda, R., Goldreich, O., and Itai, A. 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 (1992), 104–126.

[3] Bar-Yehuda, R., Israeli, A., and Itai, A. Multiple communication in multi-hop radio networks. *SIAM Journal on Computing 22*, 4 (1993), 875–887.

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

[5] Chlebus, B. S., Kowalski, D. R., Pelc, A., and Rokicki, M. A. Efficient distributed communication in ad-hoc radio networks. In *Proceedings of the International Conference on Automata, Languages and Programming* (2011), pp. 613–624.

[6] Czumaj, A., and Rytter, W. Broadcasting algorithms in radio networks with unknown topology. In *Proceedings of the Symposium on Foundations of Computer Science* (2003), pp. 492–501.

[7] Gasieniec, L., Peleg, D., and Xin, Q. Faster communication in known topology radio networks. In *Proceedings of the ACM SIGACT-SIGOPS Symposium on Principles of Distributed Computing* (2005), pp. 129–137.

[8] Gasieniec, L., and Potapov, I. Gossiping with unit messages in known radio networks. In *IFIP TCS* (2002), pp. 193–205.

[9] Ghaffari, M., and Haeupler, B. Near optimal leader election in multi-hop radio networks. In *Proceedings of the ACM-SIAM Symposium on Discrete Algorithms* (2013), pp. 748–766.

[10] Ghaffari, M., Haeupler, B., and Khabbazian, M. A Bound on the Throughput of Radio Networks. *arXiv http://arxiv.org/abs/1302.0264* (2013).

[11] Haeupler, B. Analyzing network coding gossip made easy. In *Proceedings of the Symposium on Theory of Computing* (2011), STOC '11, pp. 293–302.

[12] Haeupler, B., Kim, M., and Medard, M. Optimality of network coding with buffers. In *Proceedings of the IEEE Information Theory Workshop* (2011), pp. 533–537.

[13] Ho, T., Koetter, R., Medard, M., Karger, D. R., and Effros, M. The benefits of coding over routing in a randomized setting. In *Proceedings of IEEE International Symposium on Information Theory* (2003).

[14] Khabbazian, M., and Kowalski, D. Time-efficient randomized multiple-message broadcast in radio networks. In *Proceedings of the ACM SIGACT-SIGOPS Symposium on Principles of Distributed Computing* (2011), pp. 373–380.

[15] Kowalski, D., and Pelc, A. Broadcasting in undirected ad hoc radio networks. In *Proceedings of the ACM SIGACT-SIGOPS Symposium on Principles of Distributed Computing* (2003), pp. 73–82.

[16] Kowalski, D. R., and Pelc, A. Optimal deterministic broadcasting in known topology radio networks. *Distributed Computing 19*, 3 (2007), 185–195.

[17] Kushilevitz, E., and Mansour, Y. An $\Omega(D \log(N/D))$ lower bound for broadcast in radio networks. In *Proceedings of the ACM SIGACT-SIGOPS Symposium on Principles of Distributed Computing* (1993), pp. 65–74.

[18] Manne, F., and Xin, Q. Optimal gossiping with unit size messages in known topology radio networks. In *Proceedings of the Workshop on Combinatorial and Algorithmic Aspects of Networking* (2006), pp. 125–134.

[19] Peleg, D. Time-efficient broadcasting in radio networks: A review. In *Proceedings of The International Conference on Distributed Computing and Internet Technologies* (2007), pp. 1–18.

[20] Schneider, J., and Wattenhofer, R. What is the use of collision detection (in wireless networks)? *Distributed Computing* (2010), 133–147.

[21] Xin, Q. personal communication, May, 2012.