Erasure Correction for Noisy Radio Networks

Keren Censor-Hillel
Technion, Haifa, Israel
ckeren@cs.technion.ac.il

Bernhard Haeupler
Computer Science Department, Carnegie Mellon University, Pittsburgh, PA, USA
haeupler@cs.cmu.edu

D. Ellis Hershkowitz
Computer Science Department, Carnegie Mellon University, Pittsburgh, PA, USA
dhershko@cs.cmu.edu

Goran Zuzic
Computer Science Department, Carnegie Mellon University, Pittsburgh, PA, USA
gzuzic@cs.cmu.edu

Abstract

The radio network model is a well-studied model of wireless, multi-hop networks. However, radio networks make the strong assumption that messages are delivered deterministically. The recently introduced noisy radio network model relaxes this assumption by dropping messages independently at random.

In this work we quantify the relative computational power of noisy radio networks and classic radio networks. In particular, given a non-adaptive protocol for a fixed radio network we show how to reliably simulate this protocol if noise is introduced with a multiplicative cost of $\text{poly}(\log \Delta, \log\log n)$ rounds where n is the number nodes in the network and Δ is the max degree. Moreover, we demonstrate that, even if the simulated protocol is not non-adaptive, it can be simulated with a multiplicative $O(\Delta \log^2 \Delta)$ cost in the number of rounds. Lastly, we argue that simulations with a multiplicative overhead of $o(\log \Delta)$ are unlikely to exist by proving that an $\Omega(\log \Delta)$ multiplicative round overhead is necessary under certain natural assumptions.

2012 ACM Subject Classification Theory of computation → Distributed computing models; Theory of computation → Distributed algorithms; Theory of computation → Graph algorithms analysis

Keywords and phrases radio networks, erasure correction, noisy radio networks, protocol simulation, distributed computing models

Digital Object Identifier 10.4230/LIPIcs.DISC.2019.10

Related Version A full version of the paper is available at `https://arxiv.org/abs/1805.04165`.

Funding Supported in part by NSF grants CCF-1527110, CCF-1618280, CCF-1814603, CCF-1910588, NSF CAREER award CCF-1750808 and a Sloan Research Fellowship.
Keren Censor-Hillel: Supported in part by the Israel Science Foundation (grant 1696/14), the Binational Science Foundation (grant 2015803) and the European Union's Horizon 2020 Research and Innovation Programme under grant agreement no. 755839.
Goran Zuzic: Supported in part by DFINITY Scholarship Program.

1 Introduction

The study of distributed graph algorithms provides precise mathematical models to understand how to accomplish distributed tasks with minimal communication. A classic example of such a model is the radio network model of [13]. Radio networks use synchronous rounds of communication and were designed to model the collisions that occur in multi-hop, wireless networks. However, radio networks make the strong assumption that, provided no collisions occur, messages are guaranteed to be delivered.

This assumption is overly optimistic for real environments in which noise may impede communication, and so previous work of [11] introduced the noisy radio network model where messages are randomly dropped with a constant probability. This prior work demonstrated that the runtime of existing state-of-the-art broadcast protocols deteriorates significantly in the face of random noise. Furthermore, it showed how to design efficient broadcasting protocols that are robust to noise.

However, it has remained unclear how much more computationally powerful radio networks are than noisy radio networks. In particular, it was not known how efficiently an arbitrary protocol from a radio network can be *simulated* by a protocol in a noisy radio network. A simulation with little overhead would demonstrate that radio networks and noisy radio networks are of similar computational power. However, if simulation was necessarily expensive then radio networks would be capable of completing more communication-intensive tasks than their noisy counterparts.

A simple observation regarding the relative power of these two models is that any polynomial-length radio network protocol can be simulated at a multiplicative cost of $O(\log n)$ rounds where n is the number of nodes in the network. In particular, we can simulate any protocol from the non-noisy setting by repeating every round $O(\log n)$ times. A standard Chernoff and union bound argument show that every message sent by the original protocol is successfully sent with high probability. However, a multiplicative $O(\log n)$ is a significant price to pay for noise-robustness for many radio network protocols. For example, the optimal known-topology message broadcast protocol of [25] uses only $O(D + \log^2 n)$ rounds, while the topology-oblivious Decay protocol of [6] takes $O(D \log n + \log^2 n)$ where D is the diameter of the network. Moreover, a dependence on a global property of the network – the number of nodes in the graph – is excessive for correcting for a local issue – faults.

The simple solution from above *globally synchronizes* nodes by forcing all nodes to simulate the same round for $O(\log n)$ repetitions. A natural question is can one more efficiently simulate a noisy protocol by enforcing only *local synchronization.* In this paper we show how to use local synchronization to efficiently simulate radio network protocols in a noisy setting.

The local synchronization technique we use is as follows. Suppose each node tracked the round in the original protocol up to which it has successfully simulated; we call this round a node's *virtual round.* In our simulation in each round each node simulate the virtual round of its neighbor which has successfully simulated the fewest total rounds; i.e. each node simulates the virtual round of its neighbor with the largest "delay", thereby "helping" it. Such a simulation is local as – unlike the above simple simulation – nodes in distant parts of the network may simulate different rounds of the original protocol. Moreover, if a node has successfully simulated fewer rounds than all of its neighbors (and all of its neighbors' neighbors) then after one more round of simulation it will successfully simulate a new round of the original protocol if no random faults occur.

However, there are at least three notable challenges in implementing and proving the efficiency of such a local-synchronization-based simulation.

1. First, one must show that locally synchronizing nodes yields a fast simulation. A priori, it is not clear that locally synchronizing nodes provides an advantage over the simple global synchronization strategy.
2. Second, one must deal with the fact that local synchronization requires nodes to determine the minimal virtual rounds of its neighbors. In particular, nodes cannot easily compute the virtual rounds of their neighbors without communicating: When node v simulates a round of the original protocol and broadcasts should it assume all of its neighbors have

now successfully simulated this round? If a random fault occurred at a receiver then clearly v should not but v has no easy means of determining whether or not any such faults occurred. One must, therefore, determine how nodes can efficiently determine the minimal virtual round of their neighbors.

3. Lastly, one must overcome the fact that not receiving a message is indistinguishable from a randomly dropped message. In particular, a node can interpret not receiving a message in a simulated round as indicating either (1) that it receives no message in the simulated round of the original protocol or (2) that it does receive a message in this simulated round but a random fault dropped this message. Thus, it is not clear how nodes ought to increment their virtual rounds when they do not receive a message. On the one hand, failing to increment a node's virtual round in the former case could needlessly slow down the simulation. On the other hand, if a node incorrectly decides it does not receive a message in a round, its idea of what messages it received will diverge from that of its neighbor who tried to send it a message. This divergence, in turn, may compound into further errors later in the simulation.

1.1 Our Contributions

In this work we present solutions for these challenges which demonstrate that, by using local synchronization, radio network protocols can be simulated by noisy radio networks with a multiplicative dependence on Δ, the maximum degree of the network. Thus, we demonstrate that, for low-degree networks, noisy radio networks are essentially of the same computational power as radio networks. Moreover, we demonstrate that this dependence is more or less tight in two natural settings.

We give our simulations in three increasingly difficult settings, each of which introduces one of the above three challenges. In particular, our first model focuses on the first challenge, our second focuses on the first two challenges and our last model deals with all three challenges. We briefly mention our techniques here but defer more thorough intuition regarding our simulation techniques until Section 4, which is after we have more formally defined our problem.

Local Progress Detection. As a warmup we begin by providing a simulation with $O(\log \Delta)$ multiplicative round overhead in the setting where nodes have access to "*local progress detection*". Roughly, local progress detection enables nodes to know when they experience a random fault and also the virtual rounds of their neighbors. Notice that in this setting challenges 2 and 3 are non-issues: if each node has local progress detection, they can easily determine neighbors' delays and distinguish not receiving a message in the original protocol from a randomly dropped message. The key technique we introduce for the first challenge is a concentration-inequality-type result based on what we call "blaming chains". (Section 5)

Non-Adaptive Protocols. Next, we provide simulations for *non-adaptive* protocols. Roughly, non-adaptive protocols are protocols in which nodes know a priori the rounds in which they receive messages. In this setting, our simulations achieve a multiplicative $\text{poly}(\log \Delta, \log\log n) = O(\log^3 \Delta \cdot \log\log n \cdot \log\log\log n)$ round overhead *without local progress detection.* As we argue in Section 6, a number of well-known protocols (e.g. the optimal broadcast algorithm of [25]) are non-adaptive. Notice that in this setting we need not deal with challenge 3 above since nodes can always distinguish a dropped message from a round in which they receive no messages because they know the rounds of the original protocol in which they receive messages. In this setting, we leverage non-adaptiveness of protocols to overcome challenge 2. In particular, we use a distributed binary search which carefully silences nodes that search over divergent ranges to inform nodes of the minimal virtual round of their neighbors. Moreover, we keep the range over which the binary search must search tractable by slowing down nodes that make progress too quickly. (Section 6)

General Protocols. Lastly, we show how to deal with all three challenges at once by giving simulations for arbitrary radio network protocols with a multiplicative $O(\Delta \log^2 \Delta)$ overhead. To overcome challenge 3 we have nodes exchange "tokens" with all neighbors in every round to preclude the possibility of a dropped message going unnoticed. (Section 7)

Lower Bounds. We also show that our simulations for non-adaptive protocols are likely optimal: We show that two natural classes of simulations necessarily use $\Omega(\log \Delta)$ multiplicatively many more rounds than the protocols which they simulate. In particular, a simulation that either (1) does not use network coding, or (2) sends information in the same way as the original protocol requires $\Omega(\log \Delta)$ multiplicatively many more rounds than the original protocol. We also give a construction which we believe gives an unconditional $\Omega(\log \Delta)$ simulation overhead. (Section 8)

2 Related Work

Several models have been studied to understand robust communication in models similar to the radio network model. [16] introduced the noisy broadcast model, which also assumes random errors, and focuses on studying the computation of functions of the inputs of nodes [18, 40, 26, 38]. The model of [16] differs from the noisy radio network model in that it assumes a complete communication network and single-bit transmissions. [42] introduced a similar model which again assumes single-bit transmissions and also does not have collisions as the noisy radio network model does. Several papers have been written on notions of noisy radio networks which assume the network admits geometric structure [33, 34]. There have also been several papers on noisy single-hop radio networks. See either [24] for a study of an adversarial model or [15] for a nice study of a random noise model. Another model which captures uncertainty is the *dual graph* model [35, 10, 36, 21, 23], in which an adversary chooses a set of unreliable edges in each round. Recent work [29] has also made use of several of our techniques for variants of the SINR model.

There has also been extensive work on two-party interactive communication in the presence of noise [44, 27, 19]. In this setting, Alice and Bob have some conversation in mind they would like to execute over a noisy channel; by adding redundancy they hope to hold a slightly longer conversation from which they can recover the original conversation outcome even when a fraction of the coded conversation is corrupted. Multi-party generalizations of this problem have also been studied [8]. The noisy radio network model can be seen as a radio network analogue of these interactive communication models in which erasures occur rather than corruptions.

Lastly, work on MAC layers has sought to provide abstractions for algorithms that hide low-level uncertainty in wireless communication [22, 35, 43, 30]. Radio networks differ from MAC layers as in radio networks it is not required that a sender receive an acknowledgment from a receiver.

Since its introduction by [13], the classic radio network model has attracted wide attention from researchers. The survey of [41] is an excellent overview of this research area. Here we focus the radio network literature that relates to our work. Much of previous work for the noisy radio network model focused on broadcast [31]. For the classic model, [6] gave a single-message broadcast algorithm for a known topology, which completes in $O(D \log n + \log^2 n)$ rounds. [11] shows that this protocol is robust to noise, completing in $O(\frac{\log n}{1-p}(D + \log n + \log \frac{1}{\delta}))$ rounds, with a probability of failure of at most δ. In the classic model, single-message broadcast was then improved by [25] and [32], who showed that in the case of a known topology, $O(D + \log^2 n)$ rounds suffice. [11] shows that this protocol is *not* robust to noise, requiring in expectation $\Theta(\frac{p}{1-p} D \log n + \frac{1}{1-p} D)$ rounds, for broadcasting a message along a path of length D. They showed an alternative protocol, completing in $O(D + \log n \log \log n (\log n + \log \frac{1}{\delta}))$ rounds, with a probability of failure of at most δ. For an unknown topology in the classic model, [14] give a protocol completing in $O(D \log(n/D) + \log^2 n)$ rounds, which is optimal, due to the $\Omega(\log^2 n)$ and $\Omega(D \log(n/D))$ lower bounds of [2] and [37], respectively. [20] give a $O(D + \text{poly} \log n)$-round protocol that uses *collision detection*.

Lastly, simulations of models of distributed computation by other models of distributed computation is a foundational aspect of distributing computing, dating back to the 80's–90's with many simulations of various shared memory primitives, faults, and more (see a wide variety in, e.g., [3]). It is also a focus for message-passing models with different features, being the motivation for synchronizers [4, 5], and additional simulations [12, 9].

3 Model and Assumptions

In this section we formally define the classic radio network model, the noisy radio network model and discuss various assumptions we make throughout the paper.

Radio Networks. A multi-hop radio network, as introduced in [13], consists of an undirected graph $G = (V, E)$ with $n := |V|$ nodes. Communication occurs in synchronous rounds: in each round, each node either broadcasts a single message containing $\Theta(\log n)$ bits to its neighbors or listens. A node receives a message in a round if and only if it is listening and *exactly* one of its neighbors is transmitting a message. If two or more neighbors of node v transmit in a single round, their transmissions *collide* at v and v does not receive anything. We assume no collision detection, meaning a node cannot differentiate between when none of its neighbors transmit a message and when two or more of its neighbors transmit a message. Nodes are assumed to have unbounded computation, though all of the protocols in our paper use polynomial computation.

We use the following notational conventions throughout this paper when referring to radio networks. Let $\text{dist}(v, w)$ be the hop-distance between u and v in G, let $\Gamma(v) = \{w \in V : \text{dist}(v, w) \leq 1\}$ denote the 1-hop neighborhood of v, and let $\Gamma^{(k)}(v) = \{w \in V : \text{dist}(v, w) \leq k\}$ denote the k-hop neighborhood of v.

Noisy Radio Networks. A multi-hop noisy radio network, as introduced in [11], is a radio network with random erasures. In particular, it is a radio network where node v receives a message in a round if and only if it is listening, exactly one of its neighbors is broadcasting and a *receiver fault* does not occur at v. Receiver faults occur at each node and in each round independently with constant probability $p \in (0, 1)$. As p will be treated as a constant it will be suppressed in our O and Ω notation. We assume that in a given round a node cannot differentiate between a message being dropped because of a fault, i.e. a collision, and all of its neighbors remaining silent.

We consider this model because independent receiver faults model transient environmental interference such as the capture effect [39].[1] Moreover, we consider an *erasure* model – entire messages are dropped – rather than a corruption model – messages are corrupted at the bit level – because, in practice, wireless communication typically incorporates error correction and checksums that can guard against bit corruptions [17]. The noisy radio network model can also be seen as modeling those cases when this error correction fails and the message cannot be reconstructed and is therefore effectively dropped.

Protocols. A protocol governs the broadcast and listening behavior of nodes in a network. In particular, a protocol tells each node in each round to listen or what message to broadcast based on the node's history. This history includes the messages the node received, when it received them and its initial private input. We assume that this private input includes the number of nodes, n, the maximum degree, Δ, the receiver fault probability, p, a private random string, and any other data nodes have as input in a protocol. Formally, a **history** for node v is an $H \in \mathcal{H}$ where $\mathcal{H}$ is all valid histories. $\mathcal{H} = \{(i, R) : i \in \mathcal{I}, R \subseteq M \times \mathbb{N}\}$ where $\mathcal{I}$ is all valid private inputs, R gives what messages v has received in each round and $M = \{0,1\}^{O(\log n)}$ is all $O(\log n)$ bit messages a node could receive in a single round. Formally, a **protocol** P of length T is a function $P : V \times [T] \times \mathcal{H} \to \{\mathit{listen}\} \cup M$ where $P(v, t, H) = \mathit{listen}$ indicates that v listens in round t and $P(v, t, H) = m \in M$ indicates that v broadcasts message $m \in M$ in round t. We let $|P| := T$ stand for the length of a T round protocol. In this paper we assume that T is at most $\text{poly}(n)$.

Simulating a Protocol in the Noisy Setting. We say that protocol P' successfully **simulates** P if, after executing P' in the noisy setting, every node in the network can reconstruct the messages it would receive if P were run in the faultless setting. In particular, for any set of private input $I \in \mathcal{I}^{|V|}$ – letting $H(v, I)$ be v's history after running P with private inputs I – it must hold that after running P' with private inputs I, every v can compute $H(v, I)$. Note that nodes running P' can send messages not sent by P or send messages in a different order than they do in P. We call P the **original protocol** and P' the **simulation protocol**. We measure the efficacy of P' as the limiting ratio of $\frac{|P'|}{|P|}$ when T is sufficiently large and n goes to infinity, which we call the **multiplicative overhead** of a simulation.

4 Techniques Overview and Our Formal Results

As earlier mentioned, we show how to simulate a faultless protocol P in three noisy settings of increasing difficulty. Throughout our simulation results, we use the notion of a **virtual round** of node v, t_v, which tracks how many rounds of P node v has successfully simulated. We say that a node v is **most delayed** in a set of nodes U when $t_v \leq \min_{w \in U} t_w$. All three simulations roughly work by having nodes first exchange their virtual rounds with their neighbors. Nodes then locally synchronize by simulating the virtual round of their most delayed neighbor. Our simulations differ in how each defines a virtual round and how nodes learn virtual rounds.

As a warmup and to study what sort of overhead local synchronization enables, we consider the setting where nodes have access to "local progress detection". Recall that local progress detection gives nodes oracle access to the virtual rounds of their neighbors as well as when faults occur. We show the following theorem which demonstrates that a $O(\log \Delta)$ overhead is possible in this setting.

[1] In contrast, a sender faults model in which an entire broadcast by a node is dropped might model hardware failures where independence of faults would be a poor modeling choice.

▶ **Theorem 1.** *Let P be a general protocol of length T for the faultless radio network model. P can be simulated in the noisy radio setting using local progress detection in $O(T \log \Delta + \log n + k)$ rounds with probability at least $1 - \exp(-k)$ for any $k \geq 0$.*

Blaming Chain Intuition. To prove the above theorem we use the idea of a blaming chain to argue that local synchronization enables small simulation overhead. Since every node simulates the round of its most delayed neighbor, a node v will have all neighbors simulate its virtual round when its virtual round is minimal among virtual rounds of nodes in $\Gamma^{(2)}(v)$. Moreover, as we will show, once v is most delayed in $\Gamma^{(2)}(v)$ it does not take *too many* additional rounds for nodes to successfully simulate v's virtual round. Thus, if v takes many rounds to successfully simulate virtual round x, it must be because there was a $u \in \Gamma^{(2)}(v)$ that required many rounds to simulate virtual round $x - 1$. In this way v can blame its delay on u. Node u, in turn, can blame the fact that it required many rounds to simulate virtual round $x - 1$ on the fact that one of its 2-hop neighbors, say w, required many rounds to simulate virtual round $x - 2$ and so on. Thus, if node v is very delayed we can explain this delay by some such blaming chain. There are exponentially many possible such blaming chains, but – by way of concentration inequalities we prove – we show that long blaming chains can be shown to have exponentially small probability and so a union bound over all long blaming chains will allow us to show that no long blaming chain occurs.

We next show how to efficiently spread virtual round information without using progress detection. In particular, we show a $\text{poly}(\log \Delta, \log \log n)$ overhead for non-adaptive protocols where nodes know a priori the rounds of the original protocol they are sent messages without collision.[2]

▶ **Theorem 2.** *Let P be a non-adaptive protocol of length T for the faultless radio network model. P can be simulated in $O((T + \log n) \log^3 \Delta \log \log n \log \log \log n)$ rounds in the noisy radio setting with high probability by* MainNonAdaptive.

Progress Throttling and Distributed Binary Search Intuition. In addition to blaming chains, the main technique we use to prove the above result is progress throttling and a novel algorithm for binary search in noisy radio networks. Since in this setting nodes can no longer learn their neighbors' virtual rounds using local progress detection, our goal is to provide nodes an alternative means of learning their neighbors virtual rounds; namely, we use progress throttling and distributed binary search. Since virtual rounds can be as small as 1 and as large as the length of the entire protocol which is as large as $\text{poly}(n)$, the range over which we must binary search might seem to be as large as $\text{poly}(n)$; a binary search over this range would require a prohibitive $O(\log n)$ iterations. For this reason, we slow down nodes that make progress too quickly by only increasing their virtual round at most once after $\log \Delta$ simulated rounds. We show that this throttling keeps all virtual rounds within an additive $O(\log n)$ range. This, in turn, allows our binary search to require only $O(\log \log n)$ iterations.

Moreover, actually implementing a distributed binary search in a radio network presents technical challenges of its own. The natural solution we use for node v to learn the smallest virtual round of a neighbor is for v to repeatedly ask its neighbors if any of them have a virtual round below the midpoint of the binary search range. v then updates its binary

[2] When we write that an event occurs **with high probability (w.h.p.)**, we mean that it occurs with probability $1 - \frac{1}{n^c}$, and that the constant $c = \Theta(1)$ can be made arbitrarily large by changing the constants in the routines.

search parameters in the usual way. This strategy enables an efficient binary search in radio networks because to update its binary search parameters, v need only hear from at most one neighbor. However, such a strategy suffers from the following problem of divergent ranges. Suppose node v has a neighbor u. u might have neighbors that are not neighbors of v which influence the range over which u searches. As such u might end up searching over a different range than v. These divergent ranges may render v's binary search nonsensical: v might query u to learn if its virtual round is below v's binary search midpoint and u could respond with whether or not it is below the midpoint of an entirely different search range, namely u's search range. We overcome this issue by carefully marking nodes "silent" if they might interfere with other nodes' binary search. Specifically, we show that, given a node v whose virtual round is minimal in $\Gamma^{(2)}(v)$, a careful silencing of possibly deviating nodes causes nodes in $\Gamma(v)$ to always update their binary search parameters in exactly the same manner as v. Also, we show that, while nodes in $\Gamma^{(2)}(v) \setminus \Gamma(v)$ might update their parameters differently, our silencing ensures that these nodes never interfere with the binary search performed by nodes in $\Gamma(v)$.

Lastly, we describe how to simulate *any* protocol with a $O(\Delta \log^2 \Delta)$ multiplicative overhead.

▶ **Theorem 3.** *Let P be a general protocol of length T for the faultless radio network model.* MainGeneral *simulates P in the noisy radio setting in $O((T \log \Delta + \log n)\Delta \log \Delta)$ rounds w.h.p.*

Token Exchange Intuition. In addition to blaming chains, the main technique we use to show the above theorem is a token exchange strategy. Recall that the main challenge in the general setting is that even if a node knows its neighbors are simulating its virtual round, the node cannot tell if the absence of a message indicates that it receives no message in this round in the original protocol or that a random fault occurred. As such if a node does not have a message to deliver to its neighbor, we force it to send its neighbor a token indicating that it has no message to send. This allows v to distinguish between a random fault and a round in which it simply receives no message.

In Section 8 we give a formal statement of how simulations in two natural settings require $\Omega(\log \Delta)$ overhead but we give some intuition here as to why this might be true here.

Lower Bound Construction Intuition. Consider a star with degree Δ where the center node wants to send T messages to its neighbors. A noiseless protocol requires T rounds but if the center only sends the original messages (and not e.g. an error correcting code) and random faults occur then the center must send each message about $\Omega(\log \Delta)$ times to guarantee all messages are delivered. Likewise consider a complete bipartite graph where each node on the left wants to deliver a message to the right. The existence of collisions in radio networks means that effectively only one node on the left can broadcast per round and, like the star, every node must broadcast about $\Omega(\log \Delta)$ times to deliver its message if there are random faults.

5 Warmup: Simulation with Local Progress Detection

We begin by giving our simulation with $O(\log \Delta)$ multiplicative overhead in the setting where nodes have access to **local progress detection**.

To rigorously define local progress detection, we introduce our notion of how much simulation progress nodes have made, the **virtual round**. We say that a node v successfully completed round t if it has succeeded in taking the action in P that it takes in t: v successfully broadcasts its message m of round t if every node in $\Gamma(v)$ either has received m or had a collision in the original protocol P at round t; v successfully completes its listening action of round t if v receives the message it receives in round t of P or a collision occurs at v in round t of P. We formally define the virtual round and local progress detection.

▶ **Definition 4** (Virtual Round). *The virtual round of node $v \in V$ in a given round of simulation P' is the smallest $t_v \in \mathbb{Z}_{\geq 1}$ such that v has not successfully completed round t_v of P.*

▶ **Definition 5** (Local Progress Detection). *In the noisy radio network model with local progress detection every node v knows the virtual round of every node $w \in \Gamma^{(2)}(v)$ in every simulation round.*

We now sketch a proof of the main theorem of this section; the full proof is in the full version of the paper.

Proof Sketch of Theorem 1. We begin by describing our simulation, P'. Each node v repeatedly does the following in each round of P'. Let u be the node in $\Gamma(v)$ with minimal t_u. v takes the action that it takes in round t_u of P. That is, v tries to "help" u by simulating its virtual round.

Let $v \in V$ and note that the virtual round t_v never decreases. Hence for $x \in \mathbb{Z}_{\geq 1}$ we can define $D_{v,x}$ as the earliest round in P' when $t_v \geq x$. Notice that $D_{v,T}$ just is the number of rounds that v takes to simulate all rounds of the original protocol P. Thus, it will suffice for us to argue that $D_{v,T}$ is not too large for every v. We begin by finding a recurrence relation on $D_{v,x}$ saying that v will advance soon after its 2-hop neighborhood $\Gamma^{(2)}(v)$ has virtual round at least x. In particular we show $D_{v,x} \leq \left(\max_{w \in \Gamma^{(2)}(v)} D_{w,x-1}\right) + Y_{v,x}$ where $Y_{v,x}$ is the maximum of at most Δ many geometric random variables. Next, we show that this recurrence shows that $D_{v,x}$ is given by the longest "blaming chain" – formal definition deferred to the full paper. We then apply a Chernoff-style tail bound (proved in the full version) to show that a fixed blaming chain has small length with very high probability. By union bounding over all blaming chains we have that the length of the maximum blaming chain must be small and therefore $D_{v,T}$ is small for every v. ◀

6 Simulation for Non-Adaptive Protocols

We now give our simulation results for non-adaptive protocols with $\text{poly}(\log \Delta, \log\log n)$ overhead.

▶ **Definition 6** (Non-Adaptive Protocol). *A non-adaptive protocol P is a protocol in which every node can determine if it receives a message in each round of P regardless of the private inputs of the nodes. Formally, each node v is given a list, M_v, of the rounds in which it receives a message without collision in P.*

We let M_v.GETNEXTROUND return the smallest round in M_v such that for every $r \in M_v$ such that $r < M_v$.GETNEXTROUND, v has received the message it receives in round r of P. Since in a non-adaptive protocol nodes know in which rounds they receive messages in P, nodes in a simulation can locally compute M_v.GETNEXTROUND. Also notice that even though non-adaptive protocols assume nodes know a priori when they are supposed to receive messages, nodes do not know the contents of these messages before receiving them.

We note that a number of well-known protocols are non-adaptive. For instance, assuming nodes know the network topology and use public randomness, the optimal broadcast algorithm of [25] is non-adaptive. In particular, under these assumptions nodes can compute the broadcast schedule of their neighbors for this protocol. The same is true of the Decay protocol of [6]. Since broadcast is the most studied problem in radio networks, the fact that the state-of-the-art broadcast algorithm is non-adaptive would seem to make studying non-adaptive protocols worthwhile. Moreover, though requiring that nodes know the network topology and use public randomness may seem restrictive, in the context of simulations this assumption is actually quite weak: we can dispense with both assumptions by simply running any primitive that informs nodes of the network topology or shares randomness before our simulation is run at a one-time additive round overhead. Any primitive to learn the network topology or share randomness from the classic radio network setting coupled with the simple $O(\log n)$ simulation as described in Section 1 suffices here. This overhead is negligible if the simulated protocol is sufficiently long or we run many simulations on our network. Lastly, we note that it is often the case that nodes can even *efficiently* compute the broadcast schedule of their neighbors – see [28] – and so this one-time cost is often quite small.

To prove Theorem 2 we build upon the idea of the preceding section of using virtual rounds to locally synchronize nodes. However, in the current setting it is difficult for nodes to confirm when their broadcasts have succeeded, and so we must relax our definition of virtual rounds. In particular, for the remainder of this section we let the **virtual round** of each node be the minimum between the largest $t_v \in \mathbb{Z}_{\geq 1}$ such that v receives a message without collision in t_v and v has successfully *received* all messages it receives up to round $t_v - 1$ in P in our simulation and a throttling variable L. That is, the virtual round of v is $\min(M_v.\text{GETNEXTROUND}, L)$. L will be a slowly increasing value and, in this way, will slow down the progress of nodes by keeping their virtual rounds small, thereby keeping all virtual rounds within an additive $O(\log n)$ range.

The main subroutine of our simulation is LEARNDELAYS (Algorithm 1). LEARNDELAYS performs a binary search to inform each node of the virtual round of its most delayed neighbor. Initially the search range of every node is between $L - O(\log n)$ and L. In each iteration of LEARNDELAYS nodes with a virtual round less than the mean of their binary search range are "active". If there is an active node within 0 or 1 hop of node v then node v lowers the upper bound of its binary search. If there are no active nodes within 2 hops of v then it raises its binary search lower bound. Otherwise there is an active node 2 hops from v – which intuitively means that v is not most delayed in $\Gamma^{(2)}(v)$ – in which case node v remains silent for the rest of the binary search so as to not interfere with the binary search of its neighbors. LEARNDELAYS uses subroutine DISTTOACTIVE to inform nodes of their nearest active node. The properties of LEARNDELAYS are given by the following lemma. Details of DISTTOACTIVE and a formal proof of the following lemma are deferred to the full version.

▶ **Lemma 7.** *Assume that $L - O(\log n) \leq t_v \leq L$ for all $v \in V$ and let v be a most delayed node in its 2-hop neighborhood. With constant probability the output of* LEARNDELAYS *for every $w \in \Gamma(v)$ is t_v, i.e., every w will try to help v advance. The runtime of* LEARNDELAYS *is $O(\log^2 \Delta \log\log n \log\log\log n)$ rounds.*

We now present the MAINNONADAPTIVE simulation routine that simulates P in the noisy setting (Algorithm 2). We let $P(v, t)$ return the action taken by node v in round t of P. The properties of MAINNONADAPTIVE are given by the following lemma.

▶ **Lemma 8.** *Assume that $L - O(\log n) \leq t_v < L$ for all $v \in V$ and let v be a most delayed node in its 2-hop neighborhood. After an innermost iteration of* MAINNONADAPTIVE, *t_v will increase by one with at least constant probability. Moreover, the running time of each innermost iteration is $O(\log^2 \Delta \log\log n \log\log\log n)$ rounds.*

Algorithm 1 LearnDelays for node v.

```
Require: $L$, $t_v$
  $lo \leftarrow L - O(\log n)$; $hi \leftarrow L$
  while $lo \neq hi$ do                                        ▷ repeats $O(\log \log n)$ rounds
      $v$ is marked as "active" iff $t_v \leq \lfloor \frac{lo+hi}{2} \rfloor$ and $v$ is not marked "silent"
      dist $\leftarrow$ DistToActive
      if dist is "=2" then mark $v$ as "silent" until the end of LearnDelays
      if dist is "=0" or "=1" then
          $hi \leftarrow \lfloor \frac{lo+hi}{2} \rfloor$
      else
          $lo \leftarrow \lfloor \frac{lo+hi}{2} \rfloor + 1$
  return $lo$
```

Proof. Fix a v that is most delayed in its 2-hop neighborhood such that $t_v < L$. By definition of a virtual round and the fact that $t_v < L$ we have that $P(v, t_v)$ is a listening action where in round t_v of P it holds that v is sent a message without collision. By Lemma 7 after LearnDelays terminates with constant probability each 1-hop neighbor w will have $m_w = t_v$. Thus, every node in the 1-hop neighborhood of v will simulate round t_v, i.e. the one neighbor that has a message for v will broadcast its message and the rest of v's neighbors will be silent. This message will be successfully received with probability p which is constant and so t_v will be incremented with constant probability. The running time follows easily by summing runtimes. ◀

Algorithm 2 MainNonAdaptive for node v.

```
  $t_v \leftarrow 1$
  for $L = 1, 2, \ldots, T + O(\log n)$ do                       ▷ "Outermost iteration"
      for repeat $O(\log \Delta)$ times do                        ▷ "Innermost iteration"
          $m_v \leftarrow$ LearnDelays
          do action $P(v, m_v)$
          if $m_v = M_v$.getNextRound and $v$ received a message then
              $t_v \leftarrow \min(L, M_v$.getNextRound$)$                ▷ Throttle $v$
```

We end this section with a proof sketch of Theorem 2; the full proof is in the full version of the paper.

Proof Sketch of Theorem 2. We divide our simulated schedule into length $O(\log n)$ chunks. Next we argue by induction that after each $O(\log n)$ outermost iterations of MainNonAdaptive every node is simulating the same chunk. This allows us to apply to apply Lemma 8 and a blaming chain argument as in Theorem 1 to argue that the current chunk is correctly simulated by all nodes. ◀

7 General Protocol Simulation

Here, we provide our results for arbitrary protocols. Our approach gives a $O(\Delta \log^2 \Delta)$ multiplicative overhead.

▶ **Theorem 3.** *Let P be a general protocol of length T for the faultless radio network model.* MainGeneral *simulates P in the noisy radio setting in $O((T \log \Delta + \log n) \Delta \log \Delta)$ rounds w.h.p.*

Again, we build on the notion of a virtual round for this setting. Our main challenge in this setting is that even if a node knows its neighbors are simulating its virtual round, the node cannot tell if the absence of a message indicates that it receives no message in this round in the original protocol or that a random fault occurred. As such, in order for node v to advance its virtual round after hearing no messages from a neighbor, v must confirm that every neighbor was silent in the simulated round. Let P be the original protocol for the faultless setting. Define the **token** for a node v in round r of P to be either the message that v is sending in round r of P or an arbitrary message indicating "v is not broadcasting" if v is silent in round r of P. Next, we (re-)define the **virtual round** of a node v to be the largest $t_v \in \mathbb{Z}_{\geq 1}$ such that v successfully received *all tokens from all neighbors* for rounds $1, 2, \ldots, t_v - 1$ (for a total of up to $(t_v - 1)\Delta$ tokens).

Our simulation algorithm works in two phases: every node first informs its neighbors of its virtual round; next, nodes help the neighbor with the smallest t_v they saw by sharing the token for that round. We now present pseudocode for the ShareKnowledge routine which shares messages from a node with all of its neighbors and MainGeneral which simulates P in the noisy setting.

Algorithm 3 ShareKnowledge for node v.

```
Require: a message, $B_v$, that $v$ wants to share
  for $O(\Delta \log \Delta)$ rounds do
      $v$ broadcasts $B_v$ with probability $\frac{1}{\Delta}$, independently from other nodes
```

▶ **Lemma 9.** *After* ShareKnowledge *terminates, a fixed node v successfully receives messages from all its neighbors with probability at least $3/4$ and successfully sends its message to all neighbors with probability at least $3/4$. The running time of* ShareKnowledge *is $O(\Delta \log \Delta)$ rounds.*

Proof. Fix an arbitrary node v. Consider the event where v receives a message from a fixed neighbor w in a fixed iteration of ShareKnowledge. This event occurs iff w broadcasts and all other neighbors of v, namely $\Gamma(v) \setminus \{w, v\}$, do not broadcast. This occurs with probability that is at least $\frac{1}{\Delta}(1 - \frac{1}{\Delta})^{|\Gamma(v)\setminus\{w,v\}|} \geq \frac{1}{\Delta}(1 - \frac{1}{\Delta})^{\Delta} \geq \Omega(\frac{1}{\Delta})$ by $(1 - \frac{1}{x})^x = \Omega(1)$. The probability that v does not hear from w after $O(\Delta \log \Delta)$ iterations is $(1 - \Omega(\frac{1}{\Delta}))^{\Delta \log \Delta} \leq \exp(-\Omega(\log \Delta)) \leq \frac{1}{4\Delta}$. Union bounding over all $|\Gamma(v)| \leq \Delta$ possibilities for w we get that the probability of v not sharing knowledge with all neighbors is at most $1/4$. ◀

Algorithm 4 MainGeneral for node v.

```
  $t_v \leftarrow 1$
  for $O(T \log \Delta + \log n)$ do
      $v$ runs ShareKnowledge($t_v$)
      $m_v \leftarrow$ smallest value $v$ receives from all nodes running ShareKnowledge
      ShareKnowledge(token for virtual round $m_v$)
      update $t_v$ if $v$ received all tokens for round $t_v$
```

▶ **Lemma 10.** *Let v be a most delayed node in its 2-hop neighborhood. After one* MainGeneral *loop iteration, t_v will increase by one with at least constant probability.*

Proofs of Lemma 10 and Theorem 3 are deferred to the full version of the paper.

8 Lower bounds

In this section we argue that an $\Omega(\text{poly}\log\Delta)$ multiplicative overhead in simulation is necessary in two natural settings. In the first setting the simulation is not permitted to use network coding [1]. In the second setting the simulation must respect information flow in the sense that we show a lower bound when the graph is directed. It follows that any simulation with constant overhead either uses network coding or does not respect information flow. We also give a construction which we believe could be used to show an $\Omega(\text{poly}\log\Delta)$ multiplicative overhead in simulation, even without any assumptions.

Additionally, we strengthen our lower bounds by proving them in the setting in which the simulation is granted "global control". Informally, global control eliminates the need for control messages by providing a centralized scheduler that can synchronize nodes based on how faults occur. The scheduler, however, cannot read the actual contents of the messages.

▶ **Definition 11** (Global Control). *We say that a noisy radio network has global control when (1) nodes know the network topology, (2) nodes learn which nodes broadcast in each round and at which nodes receiver faults occur in each round, and (3) all nodes have access to public randomness.*

It is not difficult to see that access to global control is sufficient to achieve the local progress detection of Section 5. Moreover, notice that our simulation from Section 5 with $O(\log\Delta)$ multiplicative overhead is both non-coding and respects information flow. As such, it is optimal for both settings.

8.1 Non-Coding Simulations

We now define a non-coding protocol and prove that simulations that do not use network coding suffer a $\Omega(\log\Delta)$ multiplicative overhead.

▶ **Definition 12** (Non-Coding). *We say that a simulation P' in the noisy setting, which is simulating a protocol P in the faultless setting, is non-coding if any message sent in P' is also sent in P (though possibly by different nodes).*

We consider an isolated star with degree Δ where the center node wants to send T messages to its neighbors. One can achieve a constant multiplicative overhead on this protocol by using an error correction code like Reed-Solomon [45]. However, the following lemma shows that if coding is not used no such overhead is possible.

▶ **Lemma 13.** *For any $T \geq 1$ and sufficiently large Δ there exists a faultless protocol of length T on the star network with $\Delta + 1$ nodes such that any non-coding simulation of the protocol in the noisy setting with constant success probability requires $\Omega(T\log\Delta)$ many rounds even if the simulation has access to global control.*

Proof. As noted, the network is a star with Δ leafs. Let r be the central node of the star. In our faultless protocol, r receives T private inputs $M_1, M_2, \ldots, M_T$ each of $\Theta(\log n)$ bits. r takes T rounds to broadcast each input, broadcasting M_i at round i.

Now consider a simulation of our protocol and assume for the sake of contradiction that it succeeds with constant probability. Let $C = C(p)$ be a constant such that $1 - p \geq \exp(-C)$ where $p = \Omega(1)$ is the fault probability of our noisy network. By the non-coding assumption, all messages sent by P' must be in the set $\{M_1, \ldots, M_T\}$. Denote by t_i the number of times r broadcasts M_i. For the sake of contradiction, assume that $\sum_{i=1}^{T} t_i \leq \frac{1}{100C} T\log\Delta$. Hence $\min_{i\in[T]} t_i \leq \frac{1}{100C}\log\Delta$ by an averaging argument. Let $i^* = \arg\min_{i\in[T]} t_i$. Notice that the probability that a fixed node receives a message is independent of that of any other node since WLOG only r ever broadcasts. Therefore, the probability that a fixed node does not receive M_{i^*} is $(1-p)^{t_{i^*}} \geq (1-p)^{\frac{1}{100C}\log\Delta} \geq \exp(-C\frac{1}{100C}\log\Delta) \geq \Delta^{-1/100}$ by definition of C. Consequently, the probability that some node does not receive i^* is at least $1-(1-\Delta^{-1/100})^{\Delta} \geq 1-\exp(\Delta^{99/100})$ which tends to 1 as $\Delta \to \infty$. This contradicts our assumption that our simulation succeeds with constant probability. Therefore, no simulation protocol of length $\Omega(T\log\Delta)$ can deliver all messages with constant probability. ◀

8.2 Simulations that Respect Information Flow

In this section we show how any simulation with less than a $\Omega(\text{poly}(\log\Delta))$ multiplicative overhead must route information along different paths than those in the faultless setting. In particular, we show that a $\Omega(\text{poly}(\log\Delta))$ lower bound holds in a directed network where information in any simulation flows just as it does in the noiseless setting.

▶ **Lemma 14.** *Let $G=(L\cup R, E)$ be a complete directed bipartite graph with $|L|=|R|=\Delta$ that has an arc (l,r) for all $l\in L, r\in R$. There exists a protocol P of length Δ for the faultless setting on the directed network G such that any protocol that works in the noisy setting with constant success probability requires $\Omega(\Delta\log\Delta)$ rounds. This bound holds even in the global control setting.*

Proof. Let $L=\{l_1,l_2,\ldots,l_\Delta\}$ and $R=\{r_1,r_2,\ldots,r_\Delta\}$ be the set of nodes on both sides of the partition. Every node $l_i\in L$ gets private input M_i and needs to broadcast it to all nodes in R. In the faultless protocol P, l_i broadcasts M_i in round $i\in[\Delta]$.

Now consider a noisy protocol P'. Since G is directed, the only node from L that has knowledge of M_i is l_i. Moreover, we can assume without loss of generality that in any one round of P' at most one node in L broadcasts, since if this were not the case either no node in R would be sent a message or a collision would occur at every node in R. Let t_i be the number of rounds in P' that l_i broadcasts M_i. For the sake of contradiction, assume that the length of P' is $c\Delta\log\Delta$ for a sufficiently small constant $c>0$. Therefore, $\sum_{i=1}^{\Delta} t_i \leq c\Delta\log\Delta$ and there exists $i^*\in[\Delta]$ such that $t_{i^*}\leq c\log\Delta$. Since M_{i^*} can only be broadcasted from l_{i^*}, M_{i^*} is broadcasted at most $c\log\Delta$ times by an averaging argument. Thus, we have a star with central node l_{i^*} and leaves given by R with the assumption that $c\log\Delta$ rounds suffices to spread a message from l_{i^*} to every node in R. The remainder of the proof is identical to the strategy given in Lemma 13 and hence is omitted. ◀

8.3 Unconditional Lower Bound Hypothesis

We do not believe there exists an $o(\text{poly}(\log\Delta))$ multiplicative overhead simulation, even for non-adaptive protocols, and in this section we put forward a candidate hard example that might be used to prove this claim.

Construction. Let G be a bipartite network with partition (L,R), where $|L|=|R|=n$. Divide the nodes in L into Δ groups of size $\frac{n}{\Delta}$, namely $L^1,\ldots,L^\Delta$. Let $l_1^i,\ldots,l_{n/\Delta}^i$ be the nodes in L^i. We repeat the following for $t=1,2,\ldots,\Delta$ iterations: pick a fresh independent permutation $\pi:[n]\to[R]$ and divide R into Δ groups of size n/Δ according to π. Specifically, let $R^1=\{\pi(1),\pi(2),\ldots,\pi(n/\Delta)\}, R^2=\{\pi(n/\Delta+1),\ldots,\pi(2n/\Delta)\}$, ..., $R^\Delta=\{\pi(n-n/\Delta+1),\ldots,\pi(n)\}$. Note that the grouping R changes between iterations unlike L which remains fixed. Fully connect l_t^i to all the nodes in R^i for all $i\in[\Delta]$.

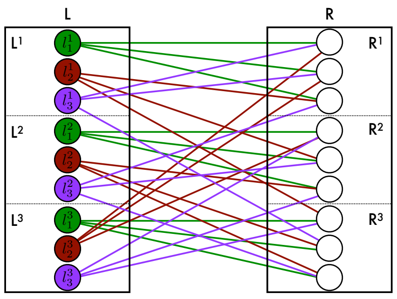

Figure 1 A particular sample of our construction which we believe could help prove an unconditional lower bound. R_i labeled according to the first iteration of our construction. $\Delta = 3$, $n = 9$. Nodes (and incident edges) colored according to the round in which they broadcast in the faultless protocol.

Why we believe this protocol is hard. Directly forwarding messages from a node $l \in L$ to each one of l's neighbors requires a multiplicative $\Omega(\log \Delta)$ overhead before all of the neighbors receive the message. Thus, if we assume there is a $o(\log \Delta)$ overhead protocol, there must be a large fraction of messages that are delivered indirectly. That is, many nodes in R receive many of the messages they need to simulate the original protocol from a different nodes than they do in the faultless protocol. However, indirectly delivering messages seems to require strictly more rounds than directly sending messages. Each $r \in R$ roughly wants to receive private input from a random subset of L. Therefore, if $r \in R$ receives a message indirectly from a neighbor $l \in L$, it is unlikely that the neighbors of l apart from r need this message to simulate the original protocol. Thus, while l delivers a message to r, it blocks all other neighbors of l from receiving messages they need to simulate the protocol. Lastly, we note that this problem roughly corresponds to a random instance of *index coding* [7], for which the bounds are currently not fully understood.

9 Future Work

Recall that the third challenge for local-synchronization-based simulations described in Section 1 is that nodes cannot distinguish between not receiving a message in the original protocol and a message being dropped by a random fault. Our general protocol solves this issue but only through a costly subroutine in which every node shares information with all of its neighbors, thereby incurring an $O(\Delta)$ overhead. We leave as an open question whether this challenge can be overcome with poly$(\log \Delta)$ overhead. "Backtracking" on faulty progress has been the subject of some interactive coding literature – see [27] – and will likely prove insightful on this front.

As a final direction for future work we note that many of our techniques apply to simulations of faulty versions of other models of distributed computing. For instance, applying the techniques of our general protocol simulation to a receiver fault version of CONGEST almost immediately yields a simulation of CONGEST with receiver faults by CONGEST with a $O(\log \Delta)$ multiplicative round overhead. We expect further applications.

References

1 R. Ahlswede, Ning Cai, S. Y.R. Li, and R. W. Yeung. Network Information Flow. *IEEE Transactions on Information Theory*, 46(4):1204–1216, September 2006.

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

3 Hagit Attiya and Jennifer Welch. *Distributed Computing. Fundamentals, Simulations, and Advanced Topics.* McGraw-Hill, 1998.

4 Baruch Awerbuch. Complexity of network synchronization. *Journal of the ACM (JACM)*, 32(4):804–823, October 1985.

5 Baruch Awerbuch and David Peleg. Network synchronization with polylogarithmic overhead. In *Annual ACM Symposium on Theory of Computing (STOC)*, 1990.

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

7 Ziv Bar-Yossef, Yitzhak Birk, TS Jayram, and Tomer Kol. Index coding with side information. *IEEE Transactions on Information Theory*, 57(3):1479–1494, 2011.

8 Mark Braverman, Klim Efremenko, Ran Gelles, and Bernhard Haeupler. Constant-rate coding for multiparty interactive communication is impossible. *Journal of the ACM (JACM)*, 65(1):4, 2017.

9 Keren Censor-Hillel, Ran Gelles, and Bernhard Haeupler. Making Asynchronous Distributed Computations Robust to Channel Noise. In *Innovations in Theoretical Computer Science Conference (ITCS)*, pages 50:1–50:20, 2018.

10 Keren Censor-Hillel, Seth Gilbert, Fabian Kuhn, Nancy A. Lynch, and Calvin C. Newport. Structuring unreliable radio networks. *Distributed Computing*, 27(1):1–19, 2014.

11 Keren Censor-Hillel, Bernhard Haeupler, D Ellis Hershkowitz, and Goran Zuzic. Broadcasting in Noisy Radio Networks. In *ACM Symposium on Principles of Distributed Computing (PODC)*, 2017.

12 Keren Censor-Hillel, Bernhard Haeupler, Jonathan A. Kelner, and Petar Maymounkov. Rumor Spreading with No Dependence on Conductance. *SIAM Journal on Computing (SICOMP)*, 46(1):58–79, 2017.

13 I. Chlamtac and S. Kutten. On Broadcasting in Radio Networks - Problem Analysis and Protocol Design. *IEEE Transactions on Communications*, 33(12):1240–1246, December 1985.

14 Artur Czumaj and Wojciech Rytter. Broadcasting algorithms in radio networks with unknown topology. *Journal of Algorithms*, 60(2):115–143, 2006.

15 Klim Efremenko, Gillat Kol, and Raghuvansh Saxena. Interactive coding over the noisy broadcast channel. In *Annual ACM Symposium on Theory of Computing (STOC)*, pages 507–520. ACM, 2018.

16 Abbas El-Gamal. Open problems presented at the 1984 workshop on specific problems in communication and computation. *Sponsored by bell communication research*, 1984.

17 Moncef Elaoud and Parameswaran Ramanathan. Adaptive use of error-correcting codes for real-time communication in wireless networks. In *INFOCOM*, volume 2, pages 548–555, 1998.

18 Robert G. Gallager. Finding parity in a simple broadcast network. *IEEE Transactions on Information Theory*, 34(2):176–180, 1988.

19 Ran Gelles et al. Coding for interactive communication: A survey. *Foundations and Trends in Theoretical Computer Science*, 13(1–2):1–157, 2017.

20 Mohsen Ghaffari, Bernhard Haeupler, and Majid Khabbazian. Randomized broadcast in radio networks with collision detection. *Distributed Computing*, 28(6):407–422, 2015.

21 Mohsen Ghaffari, Bernhard Haeupler, Nancy A. Lynch, and Calvin C. Newport. Bounds on Contention Management in Radio Networks. In *Distributed Computing (DISC)*, pages 223–237, 2012.

22 Mohsen Ghaffari, Erez Kantor, Nancy Lynch, and Calvin Newport. Multi-message broadcast with abstract mac layers and unreliable links. In *ACM Symposium on Principles of Distributed Computing (PODC)*, pages 56–65, 2014.

23 Mohsen Ghaffari, Nancy A. Lynch, and Calvin C. Newport. The cost of radio network broadcast for different models of unreliable links. In *ACM Symposium on Principles of Distributed Computing (PODC)*, pages 345–354, 2013.

24 Seth Gilbert, Rachid Guerraoui, Dariusz R Kowalski, and Calvin Newport. Interference-resilient information exchange. In *INFOCOM*, 2009.

25 Leszek Gąsieniec, David Peleg, and Qin Xin. Faster communication in known topology radio networks. *Distributed Computing*, 19(4):289–300, 2007.

26 Navin Goyal, Guy Kindler, and Michael E. Saks. Lower Bounds for the Noisy Broadcast Problem. *SIAM Journal on Computing (SICOMP)*, 37(6):1806–1841, 2008.

27 Bernhard Haeupler. Interactive channel capacity revisited. In *Symposium on Foundations of Computer Science (FOCS)*, pages 226–235, 2014.

28 Bernhard Haeupler and David Wajc. A faster distributed radio broadcast primitive. In *ACM Symposium on Principles of Distributed Computing (PODC)*, pages 361–370, 2016.

29 Magnus M Halldorsson and Tigran Tonoyan. Plain SINR is Enough! In *Proceedings of the 2019 ACM Symposium on Principles of Distributed Computing*, pages 127–136. ACM, 2019.

30 Majid Khabbazian, Dariusz R Kowalski, Fabian Kuhn, and Nancy Lynch. Decomposing broadcast algorithms using abstract MAC layers. *Ad Hoc Networks*, 12:219–242, 2014.

31 Dariusz R Kowalski and Andrzej Pelc. Time complexity of radio broadcasting: adaptiveness vs. obliviousness and randomization vs. determinism. *Theoretical Computer Science*, 333(3):355–371, 2005.

32 Dariusz R. Kowalski and Andrzej Pelc. Optimal Deterministic Broadcasting in Known Topology Radio Networks. *Distributed Computing*, 19(3):185–195, 2007.

33 Evangelos Kranakis, Danny Krizanc, and Andrzej Pelc. Fault-tolerant broadcasting in radio networks. In *Annual European Symposium on Algorithms (ESA)*, pages 283–294, 1998.

34 Evangelos Kranakis, Michel Paquette, and Andrzej Pelc. Communication in random geometric radio networks with positively correlated random faults. In *International Conference on Ad-Hoc Networks and Wireless*, pages 108–121. Springer, 2008.

35 Fabian Kuhn, Nancy Lynch, and Calvin Newport. The abstract MAC layer. In *Distributed Computing (DISC)*, pages 48–62, 2009.

36 Fabian Kuhn, Nancy A. Lynch, Calvin C. Newport, Rotem Oshman, and Andréa W. Richa. Broadcasting in unreliable radio networks. In *ACM Symposium on Principles of Distributed Computing (PODC)*, pages 336–345, 2010.

37 Eyal Kushilevitz and Yishay Mansour. An Omega(D Log(N/D)) Lower Bound for Broadcast in Radio Networks. In *ACM Symposium on Principles of Distributed Computing (PODC)*, 1993.

38 Eyal Kushilevitz and Yishay Mansour. Computation in Noisy Radio Networks. In *Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, volume 98, pages 236–243, 1998.

39 Krijn Leentvaar and Jan Flint. The capture effect in FM receivers. *IEEE Transactions on Communications*, 24(5):531–539, 1976.

40 Ilan Newman. Computing in Fault Tolerance Broadcast Networks. In *IEEE Conference on Computational Complexity (CCC)*, pages 113–122, 2004.

41 David Peleg. *Time-Efficient Broadcasting in Radio Networks: A Review*, pages 1–18. Springer Berlin Heidelberg, 2007.

42 Sridhar Rajagopalan and Leonard Schulman. A coding theorem for distributed computation. In *Annual ACM Symposium on Theory of Computing (STOC)*, pages 790–799, 1994.

43 Andréa W Richa and Christian cheideler. Jamming-Resistant MAC Protocols for Wireless Networks. *Encyclopedia of Algorithms*, pages 1–5, 2008.

44 Leonard J Schulman. Coding for interactive communication. *IEEE Transactions on Information Theory*, 42(6):1745–1756, 1996.

45 Stephen B. Wicker. *Reed-Solomon Codes and Their Applications.* IEEE Press, Piscataway, NJ, USA, 1994.