Protecting Single-Hop Radio Networks from Message Drops

Klim Efremenko
Ben-Gurion University, Beer Sheva, Israel

Gillat Kol
Princeton University, NJ, USA

Dmitry Paramonov
Princeton University, NJ, USA

Raghuvansh R. Saxena
Microsoft Research, Cambridge, MA, USA

Abstract

Single-hop radio networks (SHRN) are a well studied abstraction of communication over a *wireless* channel. In this model, in every round, each of the n participating parties may decide to *broadcast* a message to all the others, potentially causing collisions. We consider the SHRN model in the presence of *stochastic message drops* (i.e., *erasures*), where in every round, the message received by each party is *erased* (replaced by $\perp$) with some small constant probability, independently.

Our main result is a *constant rate coding scheme*, allowing one to run protocols designed to work over the (noiseless) SHRN model over the SHRN model with erasures. Our scheme converts any protocol Π of length at most exponential in n over the SHRN model to a protocol Π' that is resilient to constant fraction of erasures and has length linear in the length of Π.

We mention that for the special case where the protocol Π is *non-adaptive*, i.e., the order of communication is fixed in advance, such a scheme was known. Nevertheless, adaptivity is widely used and is known to hugely boost the power of wireless channels, which makes handling the general case of adaptive protocols Π both important and more challenging. Indeed, to the best of our knowledge, our result is the first constant rate scheme that converts adaptive protocols to noise resilient ones in any multi-party model.

2012 ACM Subject Classification Theory of computation → Communication complexity

Keywords and phrases Radio Networks, Interactive Coding, Error Correcting Codes

Digital Object Identifier 10.4230/LIPIcs.ICALP.2023.53

Category Track A: Algorithms, Complexity and Games

Related Version *Full Version*: `https://eccc.weizmann.ac.il/report/2023/066/`

Funding *Klim Efremenko*: Supported by the Israel Science Foundation (ISF) through grant No. 1456/18 and European Research Council Grant number: 949707.
Gillat Kol: supported by a National Science Foundation CAREER award CCF-1750443 and by a BSF grant No. 2018325.

1 Introduction

Over the last decades, wireless communication found many applications and has transformed technology. On the theoretical side, wireless systems were studied by numerous works, many of which consider the *single-hop radio networks* (SHRN) model of Chlamtac and Kutten [7], which abstracts a simple broadcast channel.

The classical model of SHRN assumes that the communication is *noiseless*, guaranteeing that (if no "collisions" occur) the message broadcast in a round will be received correctly by all the parties. In contrast, recently, Censor-Hillel, Haeupler, Hershkowitz, and Zuzic [6],

initiated the study of the radio networks model under *stochastic message drops* (*a.k.a*, *stochastic erasures*). In their model, each party only gets the message that was broadcast with probability $1-\epsilon$, independently, for some small constant ϵ. Otherwise, the round is "erased" for this party, meaning that it is received as a silent round, as if nothing was broadcast.

While the (noiseless) radio networks model is, by now, mostly well understood, and while noise is inherent in almost all communication systems, the relative power of noisy radio networks is far less explored. In this work we study the power of the SHRN model under the message drop noise of [6].

1.1 Our Result

Our main result is that the model of SHRN with message drops is as powerful as that of (noiseless) SHRN, in the sense that any protocol that was designed to work over the latter can be made to work over the former with a small overhead to the communication. An informal statement of our main result is in Theorem 1 (see Theorem 2 for a formal statement, the assumed model is discussed next).

▶ **Theorem 1.** *Let $n \in \mathbb{N}$ be the number of participants, $\epsilon \in (0,1)$ be the noise rate, and Γ be a non-empty alphabet set. For any protocol Π of length $T \leq 2^n$ over the (n,Γ)-broadcast channel, there is a protocol Π' with $\mathcal{O}(T)$ rounds over the (n,ϵ,Γ)-noisy broadcast channel that simulates[1] Π, and errs with probability polynomially small in T.*

We mention that our scheme works for protocols of length $T \leq 2^n$, as, if T is much larger than 2^n, there will be rounds where the messages received by all parties are erased (see Section 2.4). We also mention that our scheme uses a combinatorial building block called a *tree code* (see Section 2.2), and like other works that use tree codes, it is not computationally efficient, as no efficient tree code construction is known. Whether or not longer protocols can be handled with constant rate, and whether computationally efficient schemes are possible, are two intriguing questions we leave open.

The *collision-as-silence-as-erasures* SHRN model

We next overview the noise model of [6] used by Theorem 1 (for formal definitions, see Section 3): A protocol over the (n,ϵ,Γ)-*noisy broadcast channel* is a communication protocol between n communicating parties that proceeds in synchronous rounds. In each round, each party can decide to either broadcast a symbol from Γ or stay silent. If more than one party broadcasts in a given round (a *collision*), or none of the parties broadcast (a *silent round*), then the "$\perp$" symbol is received by all the parties[2]. Otherwise, exactly one of the parties broadcasts a symbol, and each party receives the broadcast symbol with probability $1-\epsilon$, and $\perp$ with probability ϵ, independently[3]. A protocol over the (n,Γ)-*broadcast channel* is a protocol over the $(n,0,\Gamma)$-noisy broadcast channel, i.e., one where erasures do not occur.

---

[1] By "Π' that simulates Π", we mean that a transcript for Π can be retrieved from a transcript for Π', see Theorem 2.

[2] The name *collision-as-silence* is because the same $\perp$ symbol is received in both collision and silent rounds. This model is, perhaps, the most common model in the literature. Another very popular model is the *collision detection* model, where collision and silence are perceived as different symbols. Theorem 1 is stated for the collision-as-silence model, but applies to the collision detection model as well.

[3] Modeling erasures as the same symbol as collisions/silences only makes our result stronger. As explained in Section 2.3, this makes our erasure model closer to the corruption model.

1.2 Corruption Noise and Adaptivity

The corruption noise model

One of the original motivations for our work was exploring the power of the SHRN model under stochastic *corruption noise*, a noise model that received quite a bit of attention over the last few years (see, e.g., [10, 11]). In this model, in every round, each party receives the correct symbol output by the channel with probability $1 - \epsilon$, and receives one of the other symbols with probability ϵ, independently[4]. Observe that protecting protocols against corruptions is at least as hard as protecting them against message drops.

Adaptivity and the [10] scheme

An encouraging piece of evidence, indicating that it may be possible to make SHRN protocols resilient to corruption noise with small overhead, was recently given by Efremenko, Kol, and Saxena [10], who designed such a scheme for a restricted set of protocols called *non-adaptive* protocols. Still, our initial belief was that such a scheme is impossible in the general case of *adaptive* protocols.

Non-adaptive (*a.k.a*, *oblivious* or *static*) protocols are a restricted set of protocols where it is known ahead of time which party broadcasts in what round, while adaptive protocols allow the parties to decide whether or not they wish to broadcast at a given round based on their input and their received transcript up until the current round.

While non-adaptive protocols are useful, they do not fully utilize the power of the wireless channel, and communication-efficient protocols for some central problems are, in fact, adaptive (e.g., the celebrated Decay protocol for computing the size of a network [3]). This additional power of adaptive protocols is what makes their conversion to noise-resilient ones more challenging, and, indeed, the [10] scheme may fail when applied to adaptive protocols Π.

When starting this project, we identified two inherent reasons (see Section 2.1) for the failure of [10] when applied to adaptive protocols and hoped to show that these must lead to a blowup of $\tilde{\Omega}(\log n)$ in the communication. As most interactive coding lower bounds for multi-party protocols also extend to the message drop model (e.g., [4, 11]), as a first step, we attempted to convince ourselves that no constant rate simulation scheme exists even for the SHRN model with message drop noise.

To our surprise, we were able to overcome both problems in the message drop model and design a scheme that also works for adaptive protocols. As far as we know, the scheme converting noiseless to noise-resilient protocols we construct in our proof of Theorem 1 is the first constant overhead scheme that handles adaptive protocols in any multi-party setting.

We are still very interested in the more general question of making SHRN protocols resilient to corruption noise, as we believe it is a basic and "clean" coding question. Our result can be interpreted as saying that (at least for protocols that are not extremely long) either a high-rate scheme is possible or a novel lower bound approach is required.

1.3 Related Work

Interactive coding. *Interactive error correcting codes* encode interactive communication protocols designed to work over noiseless channels to protocols that also work over noisy channels. The study of interactive codes was initiated by a seminal paper of Schulman [25]

[4] Care needs to be taken while defining an error model for corruptions, as some definitions may allow for signaling-based protocols [20].

that considered two-party protocols, which was also the topic of many follow-up works. Interactive codes for multi-party distributed channels received quite a bit of attention over the last few years. These include codes for *peer-to-peer* channels [24, 21, 20, 1, 4, 16, 17] and codes for various *wireless* channels [5, 10, 6, 11, 12, 2, 8].

Coding for wireless systems. The models of wireless communication considered in the context of noise-resilience differ on a few axes. The first axis is the *adaptivity* of the simulation protocol: in some papers the target simulation protocol is allowed to be adaptive and in others it must be non-adaptive. (Of course, if the noiseless protocols considered are adaptive, the simulation needs to be adaptive. However, simulations of non-adaptive noiseless protocols by adaptive noise-resilient protocols have been considered). The second axis is whether *single or multi-hop* networks are considered. Finally, the last axis is whether the noise is modeled as stochastic *erasures* (message drops) or stochastic *corruptions* (change of symbols).

Non-adaptive simulations. The study of noise in wireless systems can be traced back to [14] that answered an open problem of [15] by giving an $\mathcal{O}(n \log \log n)$ length communication protocol for the *bit exchange* problem (all n parties have an input bit and all parties want to know the input of all the other parties). The underlying model was the *noisy broadcast channel*, which is a non-adaptive, single-hop model with corruption errors. A matching lower bound for this problem was later given by [18]. The communication complexity of other specific n-bit functions, like the *OR*, *majority*, and *parity* functions, were studied under related models by [27, 22, 13, 23, 18]. The non-adaptive single-hop model was studied under erasure noise by [19], where an $\mathcal{O}(n \log^* n)$ protocol is given for the bit exchange problem, breaking the $\Omega(n \log \log n)$ lower bound proved for corruption errors. The general case of simulating *any* non-adaptive protocol by an noise resilient non-adaptive protocol was very recently studied by [9]. Their main result is that, for protocols of length polynomial in n, such a simulation requires $\tilde{\Theta}(\sqrt{\log n})$ multiplicative overhead in the communication complexity.

Adaptive simulations. The work of [10] gave a scheme for converting any non-adaptive noiseless protocol to an adaptive noise-resilient one with only a constant multiplicative overhead, over a single-hop network with corruption errors (in particular, implying an adaptive noise-resilient bit exchange protocol with $\mathcal{O}(n)$ communication).

Multi-hop radio networks. The work of [10] (and our current work) consider the setting where the parties are connected in a *clique* (a *single-hop* network), as it is assumed that when a party transmits, all other parties can hear the transmission. As mentioned above, this topology is the single most extensively studied, as it represents the simplest broadcast channel. However, wireless systems can have arbitrary topologies.

In contrast to [10], in [11] it is shown that such a scheme is impossible over general multi-hop networks, where each of the n communicating parties is associated with a node in the graph, and when a party broadcasts, its message is only received by its neighbors in the graph (if there are no collisions). Specifically, [11] shows that in some networks, the cost of noise-resilience is $\Omega(\log n)$, even for simulating non-adaptive protocols by adaptive protocols. A matching $\mathcal{O}(\log n)$-overhead scheme for converting any noiseless protocol to a noise resilient one over any network is also given by [11].

The recent work of [6], considered general radio networks under message drop noise. They show that any protocol over any network can be converted to a noise resilient one with a multiplicative $\mathcal{O}(\Delta \log^2 \Delta)$ overhead to the communication, where Δ is the maximum degree of a node in the network. For the special case in which the noiseless protocol we wish to convert is non-adaptive, a scheme with an improved overhead of

$\text{poly}(\log\Delta, \log\log n)$ is shown [6]. For networks with small Δ, this implies an efficient simulation of noiseless protocols. However, for networks with large Δ, the [6] simulation can have a huge overhead. This is not for no reason, as the $\Omega(\log n)$ lower bound of [11] mentioned above also applies to the message drop noise and implies that there exist network topologies with large Δ for which an $\Omega(\log\Delta)$ overhead is necessary. Our result shows for the important single-hop topology, these communication overheads can be avoided altogether.

2 Proof Sketch

In this section, we give a detailed sketch of our protocol.

As mentioned in Section 1.2, one of the main motivations for our work was studying the rate of interactive codes over the SHRN model with corruptions. The restricted case where the protocol Π to be simulated is non-adaptive was studied by [10], but their scheme fails for adaptive protocols. We next explain the inherent reasons for this failure and then outline our solutions for erasure noise.

2.1 The [10] Scheme

The rewind-if-error framework

The [10] scheme utilizes the *rewind-if-error* framework, which was initially designed for the two-party setting [25]. Rewind-if-error coding schemes consist of many iterations, where each iteration consists of two phases: a *simulation phase*, where a small number of rounds of the noiseless protocol Π are executed, and a *consistency check phase* where the parties attempt to check if they have the same received transcript or whether an error occurred (e.g., by comparing hashes of their received transcripts). If the check phase passes, parties continue the simulation, otherwise they *rewind* and re-simulate the last few rounds.

A careful examination of the [10] scheme shows that it breaks down when applied to adaptive protocols for the following two fundamental reasons:

Repeated rewinds. The first problem is that with noise rate ϵ, we should expect about ϵn parties to experience message drops in every round of the simulation phase. Since ϵ is constant, $\epsilon n \gg 1$. This implies that the consistency check phase will almost always fail and trigger a rewind, and no progress will ever be made. This situation can be trivially corrected by repeating each broadcast symbol $\mathcal{O}(\log n)$ times, and thereby effectively reducing the noise rate to less than $\frac{1}{n}$. However, this is unaffordable for a constant overhead simulation.

We note that this repeated rewinds problem is avoided by [10] as, although the total number of parties n is large, the assumed non-adaptivity of Π can be used to determine a small subset S of parties that *critically* need to know the simulated transcript. These are the parties that will broadcast in the rounds immediately following the current one. The remaining parties broadcast later in the future and therefore have more time to decode the symbol broadcast in the current round. Then, [10] show that it is enough to make sure that parties in S are not experiencing message drops, which helps reduce overhead down to a constant. Since in the adaptive case, it is possible that *any* of the n parties broadcasts next, this approach cannot be implemented.

Message certification. An even bigger problem we encounter when attempting to run the [10] scheme on adaptive protocols is that it crucially uses the fact that the symbol received from the channel in every round can be *certified* by at least one of the parties: Since Π is

assumed to be non-adaptive, it can also be assumed that a single party broadcasts in every round (collisions and silences can be eliminated ahead of time). Furthermore, this party (and all other parties) knows that it is the only one to broadcast. Therefore, if party i broadcast the symbol σ in round t of Π and some claimed transcript of Π has a symbol different from σ in round t, party i can "object" to this transcript to trigger a rewind.

The adaptive setting is different though. Consider, for example, the case where Π is adaptive and in some rounds has multiple parties broadcasting simultaneously, causing a collision. We call such collisions *intended collisions*. Suppose, however, that in round t, party i was the only one to broadcast, but the claimed transcript for Π has $\perp$ in round t. Since party i may no longer know that it is the only one to broadcast in this round, it may deem it possible that others have broadcast as well, leading to an intended collision, and thus will not object. The other, silent, parties may not object either as they may think that this is a collision or a silent round.

2.2 Avoiding The Repeated Rewinds Problem

A protocol Π exhibiting repeated rewinds

To explain how our scheme handles the first (and easier) repeated rewinds problem described above, consider the following protocol Π that exhibits it (the second, message certification problem, does not occur): The protocol is played over an underlying complete binary tree of depth $T < 2^n$. Each of the n parties gets as input, one symbol $b_v \in \{0, 1, \star\}$ for each vertex v in the tree, where the inputs are sampled as follows: First, we select one of the root-to-leaf paths in the tree uniformly at random and call it the "*correct path*". We assign each of the vertices v on this path to exactly one of the n parties uniformly at random. Here, by "assigning vertex v to party i" we mean that party i gets a bit $b_v \in \{0, 1\}$ for vertex v. If vertex v is not assigned to party i, party i gets $b_v = \star$. Additionally, each of the vertices v outside this path is assigned to many parties, say, to a set of $\frac{n}{2}$ parties selected uniformly at random.

In the noiseless protocol Π, all parties start from the root of the tree, and, upon reaching node v, a party that was not assigned v (has $b_v = \star$) stays silent, and a party that was assigned v broadcasts its bit b_v. Since each of the vertices on the correct path was assigned to exactly one party, exactly one party broadcasts a bit, and all parties then progress to the child of v indicated by this bit (that is, if 0 is broadcast they update v to be the left child of v, otherwise to the right child). This is done until a leaf is reached, which is also the output of the protocol.

Observe that since on every vertex of the correct path a single party broadcasts (and the parties know that this is the case), the message certification problem does not occur. However, since any of the n parties may potentially be the one to broadcast in the next round, the repeated rewinds problem occurs.

The *play-it-safe* simulation scheme

To avoid repeated rewinds in our simulation of Π, we make sure that parties never go off the correct path (i.e., no party ever reaches a vertex v that is not on the correct path) by guaranteeing that the parties never broadcast when it is not their turn to broadcast. To this end, our policy for the parties is that they always *play it safe* and *never broadcast unless they know the entire transcript so far.*

Of course, it may be the case that the received transcript of the party who should broadcast next contains erasures, causing it to refrain from broadcasting. Since no other party broadcasts, this will be a silent round and all parties will receive $\perp$. Upon receiving $\perp$, parties do not update their current node v in the tree. Thus, no progress is made in this round, where progress is measured as the number of steps taken on the correct path (the depth of v in the tree). Note, however, that indeed in this protocol parties never go off the correct path.

To allow progress to resume, we need to ensure that the erasures in the transcript of the party that should broadcast next are resolved (hopefully, within a few rounds). To this end, we pick one of the parties (say, the first party) to be the *leader*. After every communication round, this leader re-broadcasts the symbol it received from the channel on a *tree code* [26]. A tree code is essentially an error correcting code that can be computed "online" and ensures that the messages sent until round t will *eventually* be decoded correctly, where the probability of correct decoding greatly increases with the number of rounds that have passed since round t. Thus, parties that suffer an erasure will be able to recover the missing symbol over the next few iterations by observing what was received from the leader on the tree code. This means that, while progress may pause, it will resume within a few rounds.

2.3 Avoiding The Message Certification Problem

A harder-to-simulate protocol Π

Now let us address the second (and more severe) problem of message certification. Observe that in our simulation of the above protocol Π we did not encounter this problem. The reason is that on every vertex on the correct path a single party is scheduled to broadcast. We now consider the more general case where some of the vertices on the correct path are given to more than one party. For concreteness, say that a quarter of the vertices v on the correct path are given to exactly 2 parties, and an additional quarter is given to $\frac{n}{2}$ parties (that is, in total, there is an intended collision on half of the vertices on the correct path). Additionally, assume that the underlying tree is ternary (instead of binary), and the children of every non-leaf vertex are labeled by $\{0, 1, \perp\}$. In a case of an intended collision, the $\perp$ child of the current vertex should be taken.

Erasures can cause errors

Observe that the play-it-safe simulation protocol we had before has to change: When designing it, we assumed that there are no collisions on the correct path, thus progress was paused when a $\perp$ symbol was received (that is, the parties did not update their current vertex v in the tree). As intended collisions are now possible, we ask that, upon receiving $\perp$, the parties update v to the $\perp$ child of v.

Observe however, that since the parties are *unable to differentiate intended collisions from erasures*, as both are received as $\perp$, they may go off the correct path and will need to eventually detect the error and rewind. We note that working in the erasure model typically means that a party that does not have the correct transcript knows that it does not have the correct transcript. However, as is evident here, this reasoning does not apply to our erasure model. In this sense, our model is *closer to the corruptions model* than other erasure models.

In our simulation, parties can go off the correct path in round t if the party that was supposed to broadcast in round t (say party i) did not do so as it did not know the full transcript so far. By not broadcasting, party i potentially converts the output of the channel in round t from a bit to $\perp$ (this happens when party i was supposed to be the only one

to broadcast) or from $\perp$ to a bit (this happens when one additional party was supposed to broadcast). Recall that, owing to the usage of a tree code, party i eventually learns the complete transcript until the missed round t. When this happens, we can have party i object in the next consistency check in order to trigger a rewind. However, because the rate of erasures is constant and parties broadcast very often (recall that a quarter of the vertices on the correct path are given to $\frac{n}{2}$ parties), there are likely to be too many missed rounds and such objections will once again cause repeated rewinds.

Critical parties

To implement a rewind-if-error mechanism without repeated rewinds, we observe that rewinds are required only when the output symbol was changed due to party i (a party that was scheduled to broadcast in round t) not broadcasting in round t. Note that this only happens if the output symbol in round t is not a collision. In this case, we say that party i is *critical*[5] for round t. We use the policy that party i only objects to round t if it is critical to round t[6]. Note that this policy does not cause repeated rewinds: if many parties were supposed to broadcast in round t, none of them is critical (this round will be a collision round even if one of these parties will not broadcast). Otherwise, if few parties were supposed to broadcast in round t, then there is a good chance that round t is not erased in any of the received transcripts of these parties.

Collision-*not*-as-silence

To be able to implement the policy, party i needs to know if it was critical to the round t that it missed. Observe that if round t was a collision round even without party i broadcasting, then party i is not critical for round t, and no rewind is necessary. It is not hard to see that this is in fact the only case where a party who missed a round is not critical for this round. This means that testing criticality boils down to the ability to differentiate a collision round from a silent round.

To differentiate collision rounds from silent rounds, we use a known radio networks *collision detection* trick. Assume for the purposes of this sketch that there is some player, say the leader, that is known to not broadcast in this round[7]. We "run" the round twice, once in a black-box way (without the leader broadcasting), and once again while having the leader broadcast. If the round was a silent round, then the parties receive a $\perp$ in the first run, and a bit (non-$\perp$ symbol) in the second, while if the round was a collision, they will receive $\perp$ in both the runs. As they receive a different combination of symbols, they can distinguish between collisions and silences[8]. Note that the argument above assumes sender collision-detection, i.e., the parties that are transmitting also receive a symbol in that round. However, this assumption is not needed, see Footnote 10 and Remark 3.

---

[5] We mention that this definition differs slightly from the technical sections, but implements a similar idea.

[6] Observe that a priori, it is not clear if the parties know they are critical. We deal with this later in this section. We also note that the notion of critical parties does not appear in the algorithm description and is used only in the analysis.

[7] This assumption can easily be removed by, *e.g*, running the round an extra time where only the leader will broadcast.

[8] We note that noise can erase the symbol broadcast by the leader in the second run and effectively erase a silence out to look like a collision. We distinguish between these and regular collisions using the method described in Section 2.4.

2.4 Erasures *To* And *From* The Leader

Recall from Section 2.2 that after every round the leader re-transmits the symbol that it received from the channel in this round. We next discuss issues that can arise when the communication to/from the leader is erased.

Erasures *to* the leader

Consider the case where the true output of the channel in a given round is a bit, but the leader receives $\perp$ due to an erasure (re-transmitting this $\perp$ may cause the execution of the protocol to go off the correct path). However, since erasures are assumed to happen independently, then with probability exponentially small in n, at least one of the other parties receives the erased bit and can object in the next consistency check to trigger a rewind. Using the assumption that the length of the protocol is at most exponential in n, we get that all such leader errors will be corrected with high probability. We mention that this is the only place in our proof where we use the bound on the length of the protocol.

Erasures *from* the leader: Collision-as-silence-*not*-as-erasures

Now consider the situation where the leader receives a bit and re-transmits it, but, due to erasures, some parties receive a $\perp$. By updating their current node v using this $\perp$, these parties may fall off the correct path. As mentioned in Section 2.3, this type of error occurs as the channel does not distinguish between erasures and collisions/silences.

To circumvent this problem, we convert our collision-as-silence-as-erasures channel to a collision-as-silence-*not*-as-erasures channel. This is done by having the leader broadcast a special symbol[9] other than 0, 1, and $\perp$, in the case it receives $\perp$. As the other parties know that the leader never broadcasts $\perp$, they can deduce that any $\perp$ they may receive from it is due to an erasure. On the other hand, if they receive the special symbol, they can conclude that the round is a collision/silence.

2.5 Implementing Check Phases

The simulation scheme we discuss so far is in the rewind-if-error framework. In this sketch we attempted to show that whenever the parties go off the correct path due to erasures, at least one of the parties is able to detect the problem and object in the next check phase.

To implement a check phase, we ask parties that wish to object to broadcast a bit (say, 1), and ask all other parties to keep silent. Then, the collision detection subroutine described above allows the parties to tell whether 0, 1, or more than 1 parties were broadcasting, and thus also allows them to tell whether there exists an objecting party and a rewind should take place.

3 The Model

In this paper, we study the broadcast channel with random erasures, assuming the *collision-as-silence-as-erasures* model. To define the model and throughout this paper, we will use the following notation. For a string s, we shall use $|s|$ to denote the length of s. For $i \in [|s|]$, let s_i denote the i^{th} coordinate of s and $s_{<i}, s_{\leq i}$ denote the prefix of the first $i-1$ and i

[9] The actual proof does not require an additional symbol. Rather, we encode every symbol by two symbols.

coordinates of s, respectively. For two strings s, t over the same alphabet, denote by $\Delta(s,t)$ the Hamming distance between s and t, by $\mathsf{LCP}(s,t)$ the longest common prefix of the strings s and t, and by $s \| t$ the concatenation of s and t.

The (n, ϵ, Γ)-*noisy broadcast channel* is defined by a number $n \geq 0$ of parties, an error parameter $\epsilon > 0$, and an alphabet set Γ satisfying $|\Gamma| > 1$. We shall refer to player 1 as the *leader* Ld, use $\perp$ to denote a special symbol not in Γ (this symbol will represent collisions, silences, and deletions), and define $\Gamma_c = \Gamma \cup \{\perp\}$. We also define the (n, Γ)-*broadcast channel* to be the *noiseless* version of this channel, i.e., when $\epsilon = 0$.

Definition of a protocol

A (deterministic) *protocol* Π over the (n, ϵ, Γ)-noisy broadcast channel is defined as:

$$\Pi = \left(T, \{\mathcal{X}^i\}_{i\in[n]}, \mathcal{Y}, \{M_j^i\}_{i\in[n], j\in[T]}, \mathsf{out}\right). \tag{1}$$

Here, $T = \|\Pi\|$ is the number of rounds (or the *length*) of the protocol, $\mathcal{X}^i$ is the input space for player i, $\mathcal{Y}$ is the output space of the protocol, $M_j^i : \mathcal{X}^i \times \Gamma_c^{j-1} \to \Gamma_c$ is the function player i uses to determine what message to send in round j, and $\mathsf{out} : \Gamma_c^T \to \mathcal{Y}$ is the function the leader uses to determine the output from its received transcript. As usual, we define a *randomized protocol* to be a distribution over (deterministic) protocols.

Execution of a protocol

The protocol Π starts with all players $i \in [n]$ having an input $x^i \in \mathcal{X}^i$ and proceeds in T rounds, maintaining the invariant that before round j, for all $j \in [T]$, all players i have a transcript $\pi^i_{<j} \in \Gamma_c^{j-1}$. In round j, player i broadcasts $z^i = M_j^i(x^i, \pi^i_{<j}) \in \Gamma_c$. Define the function:

$$\mathtt{combine}(z^1, \cdots, z^n) = \begin{cases} z^i, & \text{if } \exists \text{ unique } i \in [n] \text{ such that } z^i \neq \perp \\ \perp, & \text{otherwise} \end{cases}. \tag{2}$$

Now, the symbol π^i_j received by player i in round j equals $\mathtt{combine}(z^1, \cdots, z^n)$, with probability $1-\epsilon$, and equals $\perp$, with probability ϵ, independently for all $i \in [n]$ and $j \in [T]$.[10] In the latter case, we say the message to player i in round j was *erased* by the noise. Player i appends π^i_j to $\pi^i_{<j}$ to get a transcript $\pi^i_{\leq j}$ and continues the execution of the protocol.

After T rounds, the leader outputs $\Pi^{\mathsf{Ld}}(X) = \mathsf{out}(\pi^{\mathsf{Ld}}_{\leq T}) \in \mathcal{Y}$. (Note that using only $\mathcal{O}(\max\{T, \log n\})$ additional transmissions, the leader can communicate the output to all the other parties in a reliable manner by encoding with a standard error correcting code.) We shall sometimes omit Ld when the channel is noiseless, as in this case, all the players receive the same transcript and can compute the output.

4 Our Simulation Protocol

We formalize Theorem 1 as Theorem 2 (below). (Note that by having the parties repeat every round of the original protocol Π constantly many times and taking the majority of the outputs, we get the channel noise rate to be smaller than 10^{-10}).

[10] We remark that in the literature (e.g., [6]), the broadcast channel (single-hop radio networks) is often defined such that a player that broadcasts a symbol (other than $\perp$) in a round does not receive any symbol from the channel in that round (in other words, there is no sender collision-detection). However, for simplicity of presentation, in this paper we assume this stronger model. We explain how to make our protocol work with no sender collision-detection in Remark 3.

▶ **Theorem 2** (Formal Version of Theorem 1). *There exists a constant C such that the following holds: Fix $\epsilon = 10^{-10}$, $n > 0$, an alphabet set Γ satisfying $|\Gamma| > 1$. For any protocol Π of length $T \leq 2^n$ in the (n, Γ)-broadcast channel, there is a protocol Π' over the (n, ϵ, Γ)-noisy broadcast channel, with $\|\Pi'\| \leq CT$, and such that for all inputs $X = \left(x^1, x^2, \cdots, x^n\right)$ for the players, we have:*

$$\Pr\left(\Pi'^{\mathsf{Ld}}(X) \neq \Pi(X)\right) \leq 2^{-\min(n,T)},$$

where the probability is over the noise in the channel.

We note that when n is small, so is T, so Π can be simulated by simply repeating each round sufficiently many times. As such, without loss of generality, we may assume that n is large.

The proof of Theorem 2 spans the rest of this paper. In this section we give the simulation protocol Π', and in Appendix B we give its analysis.

Let n, ϵ, Γ be as in the theorem statement and assume without loss of generality that $\Gamma = [|\Gamma|]$. Fix a protocol Π. Observe that fixing Π also fixes $T, \left\{\mathcal{X}^i\right\}_{i\in[n]}, \left\{M_j^i\right\}_{i\in[n],j\in[T]}$, etc. as in Equation (1). As a randomized protocol is simply a distribution over deterministic protocols, we can assume without loss of generality, that the protocol Π is deterministic. We also assume without loss of generality that the output of Π is just its transcript. In order to define the protocol Π', we first set up some notation.

Protocol notation

Define the sets $\mathcal{P}^{\mathsf{Ld}} = [n]$ (all parties including the leader), and $\mathcal{P} = \{2, 3, \ldots, n\}$ (all parties excluding the leader).

As motivated in Section 2, our protocol shall implicitly implement a collision detection model, having two separate symbols for collisions and silences. We shall use a special symbol $\perp_C \notin \Gamma$ to denote a collision and $\perp_S \notin \Gamma$ to denote a silence. Define $\Gamma_{cs} = \Gamma \cup \{\perp_C, \perp_S\}$.

Additionally let $\mathsf{R} \notin \Gamma$ be a special symbol indicating that the leader wants to rewind a round, and denote by $\Gamma_{csr} = \Gamma_{cs} \cup \{\mathsf{R}\}$. We shall treat both $\perp_C$ and $\perp_S$ as $\perp$ in our protocol, and output a string in Γ_{cs}^T. We also redefine the message functions, M_j^i, to take inputs from Γ_{cs}^{j-1} instead of Γ_c^{j-1}, treating both $\perp_C$ and $\perp_S$ as $\perp$, e.g., $M_j^i(x^i, \perp_C \| \perp_S) = M_j^i(x^i, \perp \| \perp)$. For simplicity, we shall pad the protocol Π with $\perp$ infinitely many times and correspondingly define, for all $i \in [n]$, $j > T$, the value $M_j^i(\cdot, \cdot) = \perp$.

Our protocol will use a $(\Gamma_{csr}, \Gamma, R_{\mathsf{TC}}, 0.4)$-tree code TC, where $R_{\mathsf{TC}} \geq \max\left(10^5, 10R\right)$ is a sufficiently large constant and R is as promised by Theorem 5. This tree code will only be written to by the leader, and will be used to log the leader's simulated transcript. In our protocol, when we say *the leader writes $s \in \Gamma_{csr}$ to the tree code*, we mean that it computes and broadcasts $\mathsf{TC}(\rho \| s)$, where ρ is the string of all the symbols it wrote to the tree code before the current s. We shall also use $\mathsf{D\text{-}TC}$ to denote the tree code decoding function from Definition 6.

We give a formal description of our protocol Π' in Algorithm 1.

▶ Remark 3. Recall from Footnote 10 that we are assuming a broadcast model with sender collision-detection. In other words, we assume that players that are talking (broadcasting a symbol other than $\perp$) also receive an output symbol from the channel. We next claim that our simulation protocol Π' can be made to work over the channel with no sender collision-detection, that is, when only players that listen (broadcast $\perp$), get the output symbol from the channel.

Algorithm 1 The simulation protocol Π'.

Input: Each party $i \in \mathcal{P}^{\mathsf{Ld}}$ holds an input $x^i \in \mathcal{X}^i$.
Output: The leader outputs $\pi \in \Gamma_{cs}^T$, that represents a transcript for Π.

1: **for** $t \in \left[10^5 T\right]$ **do**
2: Each player $i \in \mathcal{P}^{\mathsf{Ld}}$ runs parse on τ^i to get output $\left(\pi^i, r^i\right)$, where:
- τ^i, for $i \in \mathcal{P}$, is the concatenation of all messages received by player i at Line 8 up to this point (possibly none).
- τ^{Ld} is the concatenation of all messages broadcast (as opposed to received) by the leader at Line 8 up to this point (possibly none).

3: Each player $i \in \mathcal{P}^{\mathsf{Ld}}$ computes $z^i \leftarrow M^i_{|\pi^i|+1}\left(x^i, \pi^i\right)$. Set $z^i \leftarrow \perp$ if $\pi^i = \mathsf{fail}$.
4: The parties run detect-collisions, using z^i as the input for player $i \in \mathcal{P}$.
Let w^i be the output for player $i \in \mathcal{P}^{\mathsf{Ld}}$.
5: The leader represents $w^{\mathsf{Ld}} \in \Gamma_{cs}$ as an element of Γ^4 and broadcasts it in 4 rounds.
Let $\tilde{w}^i$ be the symbol decoded by player $i \in \mathcal{P}$, or $\perp$ if the player fails to decode.
6: Each player $i \in \mathcal{P}$ sets a flag $e^i \in \{1, \perp\}$ as follows:

$$e^i \leftarrow \begin{cases} 1, & \text{if } r^i = \mathsf{true} \text{ or } \tilde{w}^i = \perp_C \neq w^i \\ \perp, & \text{otherwise} \end{cases}.$$

7: The parties run detect-collisions, using e^i as the input for player $i \in \mathcal{P}$.
Let e^{Ld} be the output for the leader.
8: The leader writes $s^{\mathsf{Ld}} \in \Gamma_{csr}$ to the tree code, where

$$s^{\mathsf{Ld}} \leftarrow \begin{cases} \mathsf{R}, & \text{if } e^{\mathsf{Ld}} \neq \perp_S \\ w^{\mathsf{Ld}}, & \text{else if } z^{\mathsf{Ld}} = \perp \\ z^{\mathsf{Ld}}, & \text{else if } w^{\mathsf{Ld}} = \perp_S \\ \perp_C, & \text{otherwise} \end{cases}.$$

9: **end for**
10: The leader runs parse on τ^{Ld} to get output $\left(\pi^{\mathsf{Ld}}, r^{\mathsf{Ld}}\right)$, where τ^{Ld} is as in Line 2. The leader then outputs $\pi^{\mathsf{Ld}}_{\leq T}$.

Algorithm 2 Algorithm detect-collisions, that distinguishes between collisions and silence.

Input: Each player $i \in \mathcal{P}$ has a symbol $z^i \in \Gamma_c$ that it wishes to broadcast in this round.
Output: Each player $i \in \mathcal{P}^{\mathsf{Ld}}$ outputs a guess $w^i \in \Gamma_{cs}$ for the combined symbol.

11: In one round of communication, each player $i \in \mathcal{P}$ broadcasts z^i and the leader broadcasts $\perp$.
Let u^i be the symbol heard by player $i \in \mathcal{P}^{\mathsf{Ld}}$.
12: In one round of communication, each player $i \in \mathcal{P}$ broadcasts z^i and leader broadcasts 1.
Let $\overline{u}^i$ be the symbol heard by player $i \in \mathcal{P}^{\mathsf{Ld}}$.
13: Each player $i \in \mathcal{P}^{\mathsf{Ld}}$ returns w^i, where

$$w^i \leftarrow \begin{cases} u^i, & \text{if } u^i \neq \perp \\ \perp_S, & \text{else if } \overline{u}^i \neq \perp \\ \perp_C, & \text{otherwise} \end{cases}.$$

Algorithm 3 Algorithm parse, run locally by a player $i \in \mathcal{P}^{\mathsf{Ld}}$ to decode and parse the tree code.

Input: Player i has $\tau \in \Gamma_c^*$, its view of the symbols encoded over the tree code.
Output: Player i outputs a transcript $\pi \in \Gamma_c^*$ or fail if it failed to decode the tree code, and a rewind flag $r \in \{\mathsf{true}, \mathsf{false}\}$ which is true if the player found a problem with π.

14: Initialize π to be the empty string, $\ell \leftarrow \infty$.
15: Let $\rho \leftarrow \mathsf{D\text{-}TC}(\tau)$.
16: If $\rho = \mathsf{fail}$, terminate and return $(\mathsf{fail}, \mathsf{false})$.
17: **for** $k \in [|\rho|]$ **do**
18: **if** $\rho_k = \mathsf{R}$ **then**
19: $\pi \leftarrow \pi_{<|\pi|}$.
20: **if** $|\pi| < \ell$ **then**
21: $\ell \leftarrow \infty$.
22: **end if**
23: **else**
24: $\pi \leftarrow \pi \| \rho_k$.
25: **if** $\mathsf{D\text{-}TC}\left(\tau_{\leq (k-1)R_{\mathsf{TC}}}\right) = \mathsf{fail}$ **and** $M^i_{|\pi|}\left(x^i, \pi_{<|\pi|}\right) \neq \perp$ **and** $\rho_k \neq \perp_C$ **then**
26: $\ell \leftarrow \min(\ell, |\pi|)$.
27: **end if**
28: **end if**
29: **end for**
30: Return $(\pi, \ell \neq \infty)$.

There are two sources of problems if we assume no sender collision-detection. The first is that players $i \in \mathcal{P}$ are expected to get their own w^i at Line 4, which they use to detect erasures experienced by the leader (compute e^i in Line 6). However, as erasures are one-sided, if at least two different players $i \neq i' \in \mathcal{P}$ talk in the same round, the leader and all listening players will receive the correct symbol, i.e., $\perp_C$, as the value of w^i. As such, if an erasure causes the leader to get an incorrect w^{Ld}, there is at most one player $i \in \mathcal{P}$ who is talking. Thus, almost all players in $\mathcal{P}$ are listening, so they will have their own w^i, and this erasure is likely to be detected.

The second issue that arises is that the leader is expected to both talk and listen at Line 12. Recall that the purpose of algorithm detect-collisions is to run a round of the original protocol and essentially tell whether 0, 1, or ≥ 2 players in $\mathcal{P}$ are talking. The leader acts as a "noisemaker" in Line 12 to distinguish the case of 0 talking players from the case of ≥ 2 talking players. However, the role of a noisemaker can be handled by any other player, as long as that player would never have talked in this round otherwise.

This gives rise to the following modification of algorithm detect-collisions: We partition the parties in $\mathcal{P}$ into two non-empty sets $\mathcal{P}_1$ and $\mathcal{P}_2$. We then have parties in $\mathcal{P}_1$ perform algorithm detect-collisions with an arbitrary player in $\mathcal{P}_2$ acting as a noisemaker, and vice versa. This allows the leader to determine whether there were 0, 1, or ≥ 2 players talking in $\mathcal{P}_1$ and in $\mathcal{P}_2$, from which they can tell if there were 0, 1, or ≥ 2 players talking in $\mathcal{P}$.

As there are no other cases in the protocol Π' where a player both talks and uses the value given to it by the channel, these changes are sufficient to make the algorithm work with no sender collision-detection.

References

1 Noga Alon, Mark Braverman, Klim Efremenko, Ran Gelles, and Bernhard Haeupler. Reliable communication over highly connected noisy networks. In *Symposium on Principles of Distributed Computing (DISC)*, pages 165–173. ACM, 2016.

2 Yagel Ashkenazi, Ran Gelles, and Amir Leshem. Brief announcement: Noisy beeping networks. In *Symposium on Principles of Distributed Computing (PODC)*, pages 458–460, 2020.

3 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.

4 Mark Braverman, Klim Efremenko, Ran Gelles, and Bernhard Haeupler. Constant-rate coding for multiparty interactive communication is impossible. In *Symposium on Theory of Computing (STOC)*, pages 999–1010. ACM, 2016.

5 Keren Censor-Hillel, Bernhard Haeupler, D Ellis Hershkowitz, and Goran Zuzic. Broadcasting in noisy radio networks. In *Symposium on Principles of Distributed Computing (PODC)*, pages 33–42, 2017.

6 Keren Censor-Hillel, Bernhard Haeupler, D Ellis Hershkowitz, and Goran Zuzic. Erasure correction for noisy radio networks. In *International Symposium on Distributed Computing (DISC)*, 2019.

7 Imrich Chlamtac and Shay Kutten. On broadcasting in radio networks-problem analysis and protocol design. *IEEE Trans. Communications*, 33(12):1240–1246, 1985.

8 Klim Efremenko, Gillat Kol, Dmitry Paramonov, and Raghuvansh R. Saxena. Computation over the noisy broadcast channel with malicious parties. In *Innovations in Theoretical Computer Science Conference, (ITCS)*, volume 185, pages 82:1–82:19, 2021.

9 Klim Efremenko, Gillat Kol, Dmitry Paramonov, and Raghuvansh R. Saxena. Tight bounds for general computation in noisy broadcast networks. In *Symposium on Foundations of Computer Science (FOCS)*, pages 634–645, 2021.

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

11 Klim Efremenko, Gillat Kol, and Raghuvansh Saxena. Radio network coding requires logarithmic overhead. In *Foundations of Computer Science (FOCS)*, pages 348–369, 2019.

12 Klim Efremenko, Gillat Kol, and Raghuvansh R. Saxena. Noisy beeps. In Yuval Emek and Christian Cachin, editors, *Symposium on Principles of Distributed Computing (PODC)*, pages 418–427, 2020.

13 Uriel Feige and Joe Kilian. Finding OR in a noisy broadcast network. *Information Processing Letters*, 73(1-2):69–75, 2000.

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

15 Abbas El Gamal. Open problems presented at the 1984 workshop on specific problems in communication and computation sponsored by bell communication research. *"Open Problems in Communication and Computation", by Thomas M. Cover and B. Gopinath (editors). Springer-Verlag*, 1987.

16 Ran Gelles and Yael T Kalai. Constant-rate interactive coding is impossible, even in constant-degree networks. *IEEE Transactions on Information Theory*, 65(6):3812–3829, 2019.

17 Ran Gelles, Yael Tauman Kalai, and Govind Ramnarayan. Efficient multiparty interactive coding for insertions, deletions, and substitutions. In *Symposium on Principles of Distributed Computing (PODC)*, pages 137–146, 2019.

18 Navin Goyal, Guy Kindler, and Michael Saks. Lower bounds for the noisy broadcast problem. *SIAM Journal on Computing*, 37(6):1806–1841, 2008.

19 Ofer Grossman, Bernhard Haeupler, and Sidhanth Mohanty. Algorithms for noisy broadcast with erasures. In *Colloquium on Automata, Languages, and Programming (ICALP)*, volume 107 of *LIPIcs*, pages 153:1–153:12, 2018.

20 William M. Hoza and Leonard J. Schulman. The adversarial noise threshold for distributed protocols. In *Symposium on Discrete Algorithms (SODA)*, pages 240–258, 2016.

21 Abhishek Jain, Yael Tauman Kalai, and Allison Bishop Lewko. Interactive coding for multiparty protocols. In *Symposium on Theory of computing (STOC)*, pages 1–10, 2015.

22 Eyal Kushilevitz and Yishay Mansour. Computation in noisy radio networks. *SIAM Journal on Discrete Mathematics (SIDMA)*, 19(1):96–108, 2005.

23 Ilan Newman. Computing in fault tolerance broadcast networks. In *Computational Complexity Conference (CCC)*, pages 113–122, 2004.

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

25 Leonard J Schulman. Communication on noisy channels: A coding theorem for computation. In *Foundations of Computer Science (FOCS)*, pages 724–733. IEEE, 1992.

26 Leonard J Schulman. Deterministic coding for interactive communication. In *Symposium on Theory of computing (STOC)*, pages 747–756. ACM, 1993.

27 Andrew Chi-Chih Yao. On the complexity of communication under noise. *invited talk in the 5th ISTCS Conference*, 1997.

A Technical Preliminaries

A.1 Tree Codes

Our algorithms make use of *tree codes*, first introduced in [26].

▶ **Definition 4** (Tree Codes). *Let $\mathcal{X}$ and Γ be two alphabet sets, $R_{\mathsf{TC}} > 0$ be an integer, and $\delta \in (0,1)$. An $(\mathcal{X}, \Gamma, R_{\mathsf{TC}}, \delta)$-*tree code *is a function $\mathsf{TC} : \mathcal{X}^* \to \Gamma^{R_{\mathsf{TC}}}$ such that for any integer $k \geq 0$ and strings $x, x' \in \mathcal{X}^k$, defining $\overline{\mathsf{TC}}(x) = \mathsf{TC}(x_{\leq 1}) \| \mathsf{TC}(x_{\leq 2}) \| \cdots \| \mathsf{TC}(x)$, we have:*

$$\Delta\big(\overline{\mathsf{TC}}(x), \overline{\mathsf{TC}}(x')\big) \geq \delta R_{\mathsf{TC}} \cdot (k - |\mathsf{LCP}(x, x')|).$$

▶ **Theorem 5** ([26]). *There exists a constant $R \geq 0$ such that for any alphabet sets $\mathcal{X}$, Γ and all $R_{\mathsf{TC}} \geq R \cdot \frac{\log|\mathcal{X}|}{\log|\Gamma|}$, there exists an $(\mathcal{X}, \Gamma, R_{\mathsf{TC}}, 0.4)$-tree code.*

We will also need a way of decoding tree codes from erasures. Recall the notation $\Gamma, \perp, \Gamma_c$ from above, and let $w_{i,j} = (w_i)_j$.

▶ **Definition 6** (Decoding from Erasures). *Let TC be an $(\mathcal{X}, \Gamma, R_{\mathsf{TC}}, \delta)$-tree code. The* decoding function of TC, *denoted $\mathsf{D\text{-}TC} : (\Gamma_c^{R_{\mathsf{TC}}})^* \to \mathcal{X}^* \cup \{\mathsf{fail}\}$, is given by the following: For an integer $k \geq 0$ and $w \in (\Gamma_c^{R_{\mathsf{TC}}})^k$,*

$$\mathsf{D\text{-}TC}(w) = \begin{cases} z, & \text{if } \exists \text{ unique } z \in \mathcal{X}^k : \forall i \in [k], j \in [R_{\mathsf{TC}}] : w_{i,j} \in \{\perp, \mathsf{TC}_j(z_{\leq i})\} \\ \mathsf{fail}, & \text{otherwise} \end{cases}.$$

▷ Claim 7. Let TC be an $(\mathcal{X}, \Gamma, R_{\mathsf{TC}}, \delta)$-tree code and let $\mathsf{D\text{-}TC}$ be its decoding function. Let $k \geq 0$ be an integer and let $z \in \mathcal{X}^k$. Then, for any $\tilde{\tau} \in \left(\Gamma_c^{R_{\mathsf{TC}}}\right)^k$ such that $\forall i \in [k], j \in [R_{\mathsf{TC}}] : \tilde{\tau}_{i,j} \in \{\perp, \mathsf{TC}_j(z_{\leq i})\}$, it holds that $\mathsf{D\text{-}TC}(\tilde{\tau}) \in \{z, \mathsf{fail}\}$.

B Analyzing the Protocol

In this section we prove that the simulation protocol Π' given in Section 4 satisfies Theorem 2.

We omit the proofs of lemmas in this section for space. They can be found in the full version of the paper.

Iterations and rounds

Observe that our protocol Π' has $T' = 10^5 T$ *iterations* and each iteration has $R' = R_{\mathsf{TC}} + 8$ *rounds* of communication: 2 rounds in the call to detect-collisions in Line 4, 4 rounds in Line 5, 2 rounds in the call to detect-collisions in Line 7, and R_{TC} rounds in Line 8.

The noise indicator

For $t \in [T']$, $r \in [R']$, and $i \in [n]$, we define the indicator random variable $\mathsf{N}_{t,r,i}$ to be 1 if and only if the message received by player i in the r^{th} round of communication in iteration t is erased due to noise. For a set $S \subseteq [T']$, we shall use N_S to denote the collection $\mathsf{N} = \{\mathsf{N}_{t,r,i}\}_{t\in S, r\in[R'], i\in[n]}$ and sometimes abbreviate $\mathsf{N}_{[T']}$ as N and $\mathsf{N}_{[t]}$ as $\mathsf{N}_{\leq t}$ for all $t \in [T']$. Observe that our definition implies that the variables in N are mutually independent and identically distributed, and take the value 1 with probability ϵ.

Note that fixing any instantiation N of N together with the inputs X to the parties fixes the entire execution of Π'. In fact, for all $t \in [T']$, fixing any instantiation $N_{\leq t}$ of $\mathsf{N}_{\leq t}$ fixes the execution of the first t iterations of Π'. This means that it also fixes the values of all the variables in these iterations.

Variables

For $i \in [n]$ and a variable *var* in Algorithms 1 and 3,[11] we shall use $var_t^i(N)$ to denote the value of variable *var* as seen by player i at the end of iteration t when the noise is N. We shall use $t = 0$ to denote the values at the start of the execution and drop N when it is clear from context. As explained above, these values are determined by $N_{\leq t}$. We also use $\pi^{\mathsf{Ld}}_{T'+1}(N)$ to refer to the leader's π^{Ld} at Line 10.

The collision-not-as-silence model

To help with our analysis, we define a function `combine-CD` that intuitively captures the behavior of a broadcast channel with collision-detection. Formally, we have, for $z^1, z^2, \cdots, z^n \in \Gamma_c^*$,

$$\texttt{combine-CD}\big(z^1, \cdots, z^n\big) = \begin{cases} \perp_S, & \text{if } \forall i \in [n] : z^i = \perp \\ z^i, & \text{if } \exists \text{ unique } i \in [n] \text{ such that } z^i \neq \perp \\ \perp_C, & \text{otherwise} \end{cases}. \tag{3}$$

For the rest of the text, fix inputs $X = \big(x^1, x^2, \cdots, x^n\big)$ for the players. We abuse notation slightly and denote by $\Pi = \Pi(X)$ the transcript of the noiseless protocol Π when the inputs to the parties are as in X and the model uses `combine-CD` in place of `combine` (thus, $\Pi \in \Gamma_{cs}^T$). This is without loss of generality as a transcript in the collision-not-as-silence model only has more information than one in the collision-as-silence model.

B.1 Technical Lemmas and One-Sided Error

A key property of our model is the fact that our noise is one-sided: After collisions are resolved, the resulting symbol will either be received correctly, or will be replaced by a $\perp$. This means that if a player hears a symbol that is not $\perp$, that player will accurately know that that is the "correct" symbol, and that they were not affected by noise.

[11] We do not use this notation for variables in Algorithm 2 as that is invoked twice in every iteration.

This property means that we can make several very useful claims, which we use throughout the rest of this paper.

▶ **Lemma 8.** *For all $t \in [T']$, all $i \in [n]$, and all instantiations N of* N*,*

$$\pi_t^i(N) \in \left\{\mathsf{fail}, \pi_t^{\mathsf{Ld}}(N)\right\}.$$

As a player $i \in [n]$ sets z^i as a deterministic function of x^i and π^i at Line 3, we also directly get the following corollary.

▶ **Corollary 9.** *For all $t \in [T']$, all $i \in [n]$, and all instantiations N of* N*,*

$$z_t^i(N) \in \left\{\perp, M^i_{|\pi_t^{\mathsf{Ld}}(N)|+1}\left(x^i, \pi_t^{\mathsf{Ld}}(N)\right)\right\}.$$

Likewise, we can also analyse the behaviour of Algorithm 2, during the two calls at Line 4 and Line 7, to see the way the noise can affect the executions of this algorithm.

▶ **Lemma 10.** *For all $t \in [T']$, all $i \in [n]$, and all instantiations N of* N*,*

$$w_t^i(N) \in \left\{\perp_C, \texttt{combine-CD}\left(\perp, z_t^2(N), \ldots, z_t^n(N)\right)\right\}.$$

▶ **Lemma 11.** *For $t \in [T']$ and any instantiation N of* N*, we have:*

$$e_t^{\mathsf{Ld}}(N) = \perp_S \implies \texttt{combine-CD}\left(\perp, e_t^2(N), e_t^3(N), \cdots, e_t^n(N)\right) = \perp_S.$$

We also show some properties of the symbol s^{Ld}, and how it relates to the transcript that players maintain.

▶ **Lemma 12.** *For all $t \in [T']$ and any instantiation N of* N *such that $e_t^{\mathsf{Ld}}(N) = \perp_S$ and $w_t^{\mathsf{Ld}}(N) = \texttt{combine-CD}\left(\perp, z_t^2(N), \ldots, z_t^n(N)\right)$, we have*

$$s_t^{\mathsf{Ld}}(N) = \texttt{combine-CD}\left(z_t^{\mathsf{Ld}}(N), z_t^2(N), \ldots, z_t^n(N)\right).$$

We also analyse the behaviour of Algorithm 3, and in particular how π and ρ behave in that algorithm.

▶ **Lemma 13.** *For all $t \in [T']$ and all instantiations N of* N*,*

$$\rho_t^{\mathsf{Ld}}(N) = s_1^{\mathsf{Ld}}(N) \| \cdots \| s_{t-1}^{\mathsf{Ld}}(N).$$

▶ **Lemma 14.** *For all $t \in [T']$, all $i \in [n]$, and all instantiations N of* N*,*

- *If $s_t^{\mathsf{Ld}}(N) \neq \mathsf{R}$, then*

$$\pi_{t+1}^{\mathsf{Ld}}(N) = \pi_t^{\mathsf{Ld}}(N) \| s_t^{\mathsf{Ld}}(N).$$

- *If $s_t^{\mathsf{Ld}}(N) = \mathsf{R}$, then*

$$\pi_{t+1}^{\mathsf{Ld}}(N) = \left(\pi_t^{\mathsf{Ld}}(N)\right)_{<|\pi_t^{\mathsf{Ld}}(N)|}.$$

B.2 Bad Events

B.2.1 Noise Events

Next, we define and analyze some events based on the variable N.

The event $\mathcal{E}^{\mathsf{wo}}_{t,r}$

For $t \in [T']$, $r \in [R']$, the event $\mathcal{E}^{\mathsf{wo}}_{t,r}$ occurs if the communication in round r in iteration t is erased for a significant fraction of the players (it is "wiped out"). Formally, we have:

$$\mathcal{E}^{\mathsf{wo}}_{t,r} \coloneqq \left(\sum_{i \in [n]} \mathsf{N}_{t,r,i} \geq \frac{n}{10} \right). \tag{4}$$

The event $\mathcal{E}^{\mathsf{dc}}_{t,i}$

For $t \in [T']$, $i \in [n]$, the event $\mathcal{E}^{\mathsf{dc}}_{t,i}$ occurs if the communication in the first execution of detect-collisions, i.e., at least one of rounds 1 and 2, in iteration t is erased for player i. Formally, we have:

$$\mathcal{E}^{\mathsf{dc}}_{t,i} \coloneqq (\exists r \in [2] : \mathsf{N}_{t,r,i} = 1). \tag{5}$$

The event $\mathcal{E}^{\mathsf{or}}_{t}$

For $t \in [T']$, we define the event $\mathcal{E}^{\mathsf{or}}_{t}$ to occur if the communication in the second execution of detect-collisions (which effectively computes a logical OR of the e^i's), i.e., in at least one of rounds 7 and 8 in iteration t is erased for the leader. Formally, we have:

$$\mathcal{E}^{\mathsf{or}}_{t} \coloneqq (\exists r \in \{7, 8\} : \mathsf{N}_{t,r,\mathsf{Ld}} = 1). \tag{6}$$

The event $\mathcal{E}^{\mathsf{tc}}_{t',t,i}$

For $0 \leq t' < t \leq T'$ and $i \in [n]$, define the following event concerning the rounds 9 to R' in each iteration, i.e., the rounds where the leader broadcasts on the tree code:

$$\mathcal{E}^{\mathsf{tc}}_{t',t,i} \coloneqq \left(\sum_{s=t'+1}^{t} \sum_{r=9}^{R'} \mathsf{N}_{s,r,i} \geq \frac{2R_{\mathsf{TC}}}{5} \cdot (t - t') \right). \tag{7}$$

B.2.2 Bad Iterations

We now define sets of "bad" iterations for a given execution. Intuitively, these are iterations where our protocol does not make progress. For an instantiation N of N, we have:

$$\begin{aligned} \mathcal{B}^{\mathsf{wo}}(N) &= \left\{ t \in [T'] \mid \exists r \in [R'] : N \in \mathcal{E}^{\mathsf{wo}}_{t,r} \right\}. \\ \mathcal{B}^{\mathsf{dc}}(N) &= \left\{ t \in [T'] \mid N \in \mathcal{E}^{\mathsf{dc}}_{t,\mathsf{Ld}} \right\}. \\ \mathcal{B}^{\mathsf{or}}(N) &= \{ t \in [T'] \mid N \in \mathcal{E}^{\mathsf{or}}_{t} \}. \end{aligned} \tag{8}$$

▶ **Lemma 15.** *It holds that:*

1. $\Pr(\mathcal{B}^{\mathsf{wo}}(\mathsf{N}) \neq \emptyset) \leq 2.25^{-n}$.
2. $\Pr\left(\left| \mathcal{B}^{\mathsf{dc}}(\mathsf{N}) \cup \mathcal{B}^{\mathsf{or}}(\mathsf{N}) \right| \geq \frac{T'}{50} \right) \leq \mathrm{e}^{-\frac{T'}{100}}$.

We note that our assumption that $T \leq 2^n$ is only used in Item 1 of Lemma 15.

B.3 Bad Intervals

B.3.1 Critical Players

We now define and show results about players "critical" to the protocol, i.e., those needed to make sure we make progress in our simulation. For a set S of integers and an integer k define $S_{\leq k}$ to be the set consisting of the k smallest elements of S. If $|S| \leq k$, we define $S_{\leq k} = S$. For a transcript $\pi \in \Gamma^*_{cs}$, we define the set $\mathcal{S}(\pi)$ to be the set of all non-leader players who would broadcast in the noiseless protocol when their received transcript is π. Formally,

$$\mathcal{S}(\pi) = \left\{ i \in \mathcal{P} \mid M^i_{|\pi|+1}(x^i, \pi) \neq \perp \right\}.$$

▶ **Definition 16** (Critical Players). *For $\pi \in \Gamma^*_{cs}$, we define the set of players that are π-*critical *as* $\mathsf{Crit}(\pi) = \mathcal{S}(\mathsf{LCP}(\pi, \Pi))_{\leq 2}$.

We note that this definition is made for analysis purposes and no single player can necessarily compute the set $\mathsf{Crit}(\cdot)$.

B.3.2 Bad Intervals

Next, we define the set of possible augmented transcripts and bad intervals.

▶ **Definition 17.** *For $0 \leq t' \leq t \leq T'$ and an instantiation $N_{\leq t'}$ of $\mathsf{N}_{\leq t'}$, define the set:*

$$\mathsf{Aug}_t(N_{\leq t'}) = \left\{ \pi \in \Gamma^*_{cs} \mid \exists N_{(t',t]} : \pi^{\mathsf{Ld}}_t(N_{\leq t}) = \pi \right\}.$$

▶ **Definition 18.** *Let N be an instantiation of N. We define $\mathcal{B}^\dagger(N)$ to be the set of all intervals $(t', t]$ satisfying $0 \leq t' < t \leq T'$ for which there exists $\pi \in \mathsf{Aug}_t(N_{\leq t'})$ and $i \in \mathsf{Crit}(\pi)$ such that $\mathcal{E}^{\mathsf{tc}}_{t',t,i}$ occurs when $\mathsf{N} = N$. We also define:*

$$\mathcal{B}(N) = \bigcup_{(t',t] \in \mathcal{B}^\dagger(N)} (t', t].$$

▶ **Lemma 19.** *It holds that:*

$$\Pr\left(|\mathcal{B}(\mathsf{N})| \geq \frac{T'}{50} \right) \leq 10^{-\frac{T'}{50}}.$$

To finish this subsection, we show that $\mathcal{B}(\cdot)$ has all the iterations where a critical player fails to decode the tree code.

▶ **Lemma 20.** *For any instantiation N of N and all $t \notin \mathcal{B}(N)$, for all $i \in \mathsf{Crit}(\pi^{\mathsf{Ld}}_t(N))$, we have $\pi^i_t(N) = \pi^{\mathsf{Ld}}_t(N)$.*

B.4 A Potential Function

We now define the potential function that we shall use in the analysis. For $t \in \{0\} \cup [T']$ and an instantiation N of N, we define:

$$\Phi_t(N) = 2 \cdot \left| \mathsf{LCP}\left(\pi^{\mathsf{Ld}}_{t+1}(N), \Pi\right) \right| - \left| \pi^{\mathsf{Ld}}_{t+1}(N) \right|. \tag{9}$$

Our definition clearly implies $\Phi_0(N) = 0$ and $\Phi_t(N) \leq \left| \mathsf{LCP}\left(\pi^{\mathsf{Ld}}_{t+1}(N), \Pi\right) \right|$ for all N. Moreover, as either one symbol is appended to or removed from the end of π^{Ld} in every iteration, we have that $\Phi_t(N) \geq \Phi_{t-1}(N) - 1$ for all N and $t \in [T']$. In Lemma 25 we will now show that if t is not in one of the bad sets defined above, then the potential increases by at least 1. But first, we state some helpful lemmas.

▶ **Lemma 21.** *For any instantiation N of N and any $t \notin \mathcal{B}^{\mathsf{wo}}(N)$, we have:*

$$s_t^{\mathsf{Ld}}(N) \in \left\{\mathsf{R}, \texttt{combine-CD}\left(z_t^{\mathsf{Ld}}(N), z_t^2(N), \ldots, z_t^n(N)\right)\right\}$$

For $i \in [n]$, define the variable $\hat{r}^i$ to be the value of r output by Algorithm 3, when run by player i, with Line 15 replaced[12] by $\rho \leftarrow \mathsf{D\text{-}TC}\left(\tau^{\mathsf{Ld}}\right)$. This value is only used for analysis purposes and cannot be computed by the player during the execution of the protocol (as they may not know τ^{Ld}). We now claim several useful properties of $\hat{r}_t^i$, and how it relates to r_t^i.

▶ **Lemma 22.** *For $t \in [T']$ and any instantiation N of N such that $\mathcal{B}^{\mathsf{wo}}(N) = \emptyset$, we have:*

$$\left(\nexists j \in \left[\left|\pi_t^{\mathsf{Ld}}(N)\right|\right] : \left(\pi_t^{\mathsf{Ld}}(N)\right)_j \neq \Pi_j\right) \implies \left(\nexists i \in \mathcal{P} : \hat{r}_t^i(N) = \mathsf{true}\right).$$

We also prove a modified converse version of the previous lemma.

▶ **Lemma 23.** *For $t \in [T']$ and any instantiation N of N such that $\mathcal{B}^{\mathsf{wo}}(N) = \emptyset$, we have:*

$$\left(\exists j \in \left[\left|\pi_t^{\mathsf{Ld}}(N)\right|\right] : \left(\pi_t^{\mathsf{Ld}}(N)\right)_j \neq \Pi_j\right) \implies \left(\exists i \in \mathsf{Crit}(\pi_t^{\mathsf{Ld}}(N)) : \hat{r}_t^i(N) = \mathsf{true}\right),$$

▶ **Lemma 24.** *For $t \in [T']$, $i \in [n]$ and any instantiation N of N, $r_t^i(N) \in \left\{\hat{r}_t^i(N), \mathsf{false}\right\}$. Furthermore, if $\pi_t^i(N) = \pi_t^{\mathsf{Ld}}(N)$, then $r_t^i(N) = \hat{r}_t^i(N)$.*

▶ **Lemma 25.** *For $t \in [T']$ and any instantiation N of N such that $\mathcal{B}^{\mathsf{wo}}(N) = \emptyset$, we have:*

$$t \notin \mathcal{B}^{\mathsf{dc}}(N) \cup \mathcal{B}^{\mathsf{or}}(N) \cup \mathcal{B}(N) \implies \Phi_t(N) \geq \Phi_{t-1}(N) + 1.$$

B.5 Finishing the proof of Theorem 2

We are now ready to finish the proof of Theorem 2.

Proof of Theorem 2. Let $C \geq 100 R_{\mathsf{TC}}$. Fix ϵ, n and Γ as in the statement of the theorem. We claim that the algorithm provided in Algorithm 1 satisfies all the properties claimed by the theorem. It can be observed that Algorithm 1 takes at most CT rounds of communication, so it just suffices to just show that $\Pr\left(\Pi'^{\mathsf{Ld}}(X) \neq \Pi(X)\right) \leq 2^{-\min(n,T)}$.

By Lemmas 15 and 19 and a union bound, we get that an instantiation N of N satisfies $\left|\mathcal{B}^{\mathsf{dc}}(N) \cup \mathcal{B}^{\mathsf{or}}(N) \cup \mathcal{B}(N)\right| \leq \frac{T'}{25}$ and $\mathcal{B}^{\mathsf{wo}}(N) = \emptyset$ except with probability at most

$$10^{-\frac{1}{50}T'} + \mathrm{e}^{-\frac{1}{100}T'} + 2.25^{-n} \leq 2^{-\min\left(n, \frac{1}{100}T'\right)} \leq 2^{-\min(n,T)}.$$

Lemma 25 then states that for all such N, for all $t \notin \mathcal{B}^{\mathsf{dc}}(N) \cup \mathcal{B}^{\mathsf{or}}(N) \cup \mathcal{B}(N)$, $\Phi_t(N) \geq \Phi_{t-1}(N) + 1$. At the same time, we recall that Equation (9) also gives that for all $t \in [T']$, $\Phi_t(N) \geq \Phi_{t-1}(N) - 1$. Thus, we see that

$$\Phi_{T'}(N) \geq \left(T' - \frac{T'}{25}\right) - \frac{T'}{25} \geq \frac{9}{10}T' \geq T.$$

Furthermore, we consult Equation (9) to get that

$$\left|\mathsf{LCP}\left(\pi_{T'+1}^{\mathsf{Ld}}(N), \Pi\right)\right| \geq \Phi_{T'}(N) \geq T,$$

which implies that $\left(\pi_{T'+1}^{\mathsf{Ld}}(N)\right)_{\leq T} = \Pi_{\leq T}$. so the leader's output at Line 10 is equal to $\Pi_{\leq T}$.

As this happens except with probability at most $2^{-\min(n,T)}$, this concludes the proof. ◀

[12] We stress that Line 25 still uses τ^i and *not* τ^{Ld}.