2014 IEEE Annual Symposium on Foundations of Computer Science

Optimal Error Rates for Interactive Coding II: Efficiency and List Decoding

Mohsen Ghaffari
CSAIL, MIT
ghaffari@mit.edu

Bernhard Haeupler
Microsoft Research
haeupler@cs.cmu.edu

***Abstract*—We study coding schemes for error correction in interactive communications. Such *interactive coding schemes* simulate any n-round interactive protocol using N rounds over an adversarial channel that corrupts up to ρN transmissions. Important performance measures for a coding scheme are its maximum *tolerable error rate* ρ, *communication complexity* N, and *computational complexity*.**

We give the first coding scheme for the standard setting which performs optimally in all three measures: Our randomized non-adaptive coding scheme has a near-linear computational complexity and tolerates any error rate $\delta < 1/4$ with a linear $N = \Theta(n)$ communication complexity. This improves over prior results [1]–[4] which each performed well in two of these measures.

We also give results for other settings of interest, namely, the first computationally and communication efficient schemes that tolerate $\rho < \frac{2}{7}$ adaptively, $\rho < \frac{1}{3}$ if only one party is required to decode, and $\rho < \frac{1}{2}$ if list decoding is allowed. These are the optimal tolerable error rates for the respective settings. These coding schemes also have near linear computational and communication complexity.

These results are obtained via two techniques: We give a *general black-box reduction* which reduces unique decoding, in various settings, to list decoding. We also show how to boost the computational and communication efficiency of any list decoder to become near linear[1].

***Keywords*-Coding; Interactive Communication; Efficiency; Error-Rate; List-Decoding**

I. Introduction

"Interactive Coding" or "Coding for Interactive Communication" can be viewed as an extension of error correcting codes to interactive communications. Error correcting codes enable a party to communicate a message through a channel to another party even if a constant fraction of the symbols of the transmission are corrupted by the channel. This coding for "one-way communication" is achieved by adding redundancy to the message, that is, by coding an n-bit message into a slightly longer N-symbol coded message over some finite alphabet. Interactive coding schemes, as introduced by Schulman [5], generalize this to two-way interactive communication: they enable two parties to perform their interaction even if a constant fraction of the symbols are corrupted by the channel. This robustness against *errors* is again achieved by adding redundancy to the interactive communication, now by transforming the original interaction protocol Π which uses n *communication rounds* into a new coded protocol Π' which has longer length, ideally still $N = O(n)$ rounds. Running this coded protocol Π' both parties can recover the outcome of Π even if a constant fraction ρ of the symbols are corrupted during the execution of Π'.

Similar to the classical error correcting codes, important performance measures of an interactive coding scheme are: the *maximum tolerable error-rate* ρ that can be tolerated, the *communication complexity* N, and the *computational complexity* of the coding and decoding procedures.

For error correcting codes the classical results of Shannon show that for any constant error-rate below $1/2$, there exist codes with $N = O(n)$, that is, with a constant redundancy factor. Deterministic linear time encoding and decoding procedures that achieve this optimal error rate and redundancy are also known [6]. Interestingly, error correcting codes can also tolerate any constant error rate below 1 if one relaxes the decoding requirement to *list decoding* [7], that is, allows the receiver to output a (constant size) list of outputs of which one has to be correct. Computationally efficient list decoders are however a much more recent discovery [8], [9].

The interactive coding setting is more involved and less well understood: In 1993 Schulman [5] gave an interactive coding scheme that tolerates an adversarial error rate of $\rho = 1/240$ with a linear communication complexity $N = O(n)$. In a more recent result that revived this area, Braverman and Rao [1] increased the tolerable error rate to $\rho \leq 1/4 - \epsilon$, for any constant $\epsilon > 0$, and showed this bound to be tight if one assumes the schedule of which party transmits at what round to be fixed ahead of time, that is, if the coding scheme is required to be *non-adaptive*. Both protocols have an *exponential computational complexity*.

More efficient protocols were given in [2], [3], [10]–[13]: Gelles, Moitra, and Sahai [10] give efficient randomized coding schemes for random instead of adversarial errors. The protocol presented by Braverman in [11] uses sub-exponential time and tolerates an error rate of at most $1/40$. Kol and Raz [12], and Haeupler [13] provide efficient randomized coding schemes tolerating random and adversarial errors respectively that achieve (near) optimal communication rates but only for sufficiently small error rates. Most related to this paper is the randomized coding scheme of

0272-5428/14 $31.00 © 2014 IEEE
DOI 10.1109/FOCS.2014.49

IEEE computer society

Brakerski and Kalai [2], which runs in quadratic time and tolerates any error rate below $1/16$, and its extension by Brakerski and Naor [3], which runs in near-linear time and tolerates some small unspecified constant error rate. These protocols therefore *compromise on the maximum tolerable error-rate* to achieve computational efficiency.

Our first result shows that, in this standard setting, both computational complexity and an optimal maximum tolerable error-rate are achievable simultaneously:

Theorem I.1. *For any constant $\epsilon > 0$ and n-round protocol Π there is a randomized non-adaptive coding scheme that robustly simulates Π against an adversarial error rate of $\rho \leq \frac{1}{4} - \epsilon$ using $N = O(n)$ rounds, a near-linear $n \log^{O(1)} n$ computational complexity, and failure probability $2^{-\Theta(n)}$.*

Protocols without the non-adaptivity restriction and other interactive coding settings of interest were studied in [4], [14]–[16]: Particularly, [4], [14] study different notions of adaptivity, [15] studies a setting with shared (hidden) randomness, and the concurrent work of [16] investigates list decoding and also tolerable error rate regions for settings with two separate unidirectional channels with different error rates α, β. Most related to this paper is [4], which showed that the maximum tolerable error rate can be improved from $1/4$ to $2/7$ by using *adaptive* coding schemes, or to $1/2$ by allowing list decoding. They also showed these bounds on the maximum tolerable error rate to be optimal even if an unbounded amount of communication is allowed. However, the coding schemes achieving these error rates required *polynomially large communication complexity* $N = O(n^2)$.

We give the first computationally and communication efficient coding schemes that achieve the optimal error-rates in these settings:

Theorem I.2. *For any constant $\epsilon > 0$ and n-round protocol Π, there are the following coding schemes that robustly simulate Π:*

(A) *An adaptive unique decoding protocol tolerating error-rate $\rho \leq \frac{2}{7} - \epsilon$.*

(B) *A non-adaptive one-sided unique decoding protocol, in which only one fixed party uniquely decodes, tolerating error-rate $\rho \leq \frac{1}{3} - \epsilon$.*

(C) *A non-adaptive list decoding protocol with an $O(\frac{1}{\epsilon^2})$ list size tolerating error-rate $\rho \leq \frac{1}{2} - \epsilon$.*

These coding schemes are all randomized, use $N = O(n)$ communication rounds[1], and near-linear $n \log^{O(1)} n$ computational complexity, and have a failure probability of $2^{-\Omega(n)}$.

[1] A part in achieving these coding schemes is to boost list-decoders. While boosting always reduces the computational complexity and communication to near linear the exact communication complexity of the final scheme depends on which initial list decoder is used. If one starts with the simple list decoder from [4], which has a quadratic communication complexity, the final communication complexity becomes $N = n2^{O(\log^* n \cdot \log\log^* n)} = no(\log\log \ldots \log n)$. If one starts with the list decoder of Braverman and Efremenko [16], which has linear communication complexity, the final communication complexity stays linear, that is, $N = O(n)$, while the computational complexity improves from exponential to near linear time.

An interesting remaining question is to achieve the above results deterministically. In this paper, we already take a first step in that direction by providing non-uniform deterministic coding schemes:

Remark I.3. *For each of the coding schemes in Theorems I.1 and I.2, there exists a (non-uniform) deterministic near linear time variant with the same tolerable error-rate and linear communication complexity. It remains open whether these deterministic schemes can be found efficiently.*

A. Techniques

Our results rely on two main technical components, a *reduction* from unique decoding to list decoding, and a *boosting technique* for list-decoders. Next, we give a brief overview over these components:

1) Black-box Reductions from Unique Decoding to List Decoding: The reduction technique shows a strong connection between unique-decoding and list-decoding for interactive coding. This technique can be roughly viewed as follows: given a "good" list-decodable coding scheme, we can construct "good" unique-decoding coding schemes for various settings in a black-box manner:

Theorem I.4. *Given any non-adaptive list decodable coding scheme for an error rate of $1/2 - \epsilon$ with constant list size, one can construct unique decoding schemes with optimal error rates for various settings, while preserving the asymptotic round complexity and computational efficiency of the list decoding coding scheme. In particular, one can obtain a non-adaptive coding scheme with error rate $1/4 - O(\epsilon)$, an adaptive coding scheme with error rate $2/7 - O(\epsilon)$, and a coding scheme with one-sided decoding and error rate $1/3 - O(\epsilon)$.*

The general idea of the reduction is easier to explain in the non-adaptive setting. Intuitively, we use $O(\frac{1}{\epsilon})$ repetitions, in each of which we simulate the protocol using the provided list decoder. Thus, in each repetition, each party gets constant many guesses (i.e., decodings) for the correct path. The parties keep all edges of these paths and simultaneous with the simulations of each repetition, they send this accumulated collection of edges to each other using error correcting codes. At the end, each party outputs the path that appeared most frequently in (the decodings of) the received collections. Since we the overall error-rate is always less than what the list-decoder tolerates, in some (early enough) repetition the correct path will be added to the collections. From there on, any repetition corrupted with an error-rate less than what the list-decoder tolerates will reinforce the correct path. This will be such that at the end, a majority-based rule is sufficient for finding the correct path.

Remark: For the reduction in the non-adaptive setting, it suffices if we start with a list-decoder that tolerates a

suboptimal error-rate of $1/4 - \epsilon$ instead of the $1/2 - \epsilon$ stated in Theorem I.4. This allows us to interpret the unique decoder of Braverman and Rao [1] as a list decoder (with list size one), boost it to become computationally efficient, and finally transform it back to a unique decoder with the same tolerable error rate. We note the reductions for the adaptive setting and the one-sided decoding setting are much more involved and do not allow for such a slack in the error rate of the initial list decoder. This makes it imperative to start with a list decoder tolerating the optimal $1/2-\epsilon$ error-rate, at least if one wants unique decoding algorithms with optimal tolerable error rates of $2/7$ and $1/3$.

2) Boosting the Efficiency of List Decoding: Given these reductions, the remaining problem is to construct "good" list-decodable coding schemes. The main technical element we use in this direction is an approach that allows us to *boost* the performance measures of list-decoders. Particularly, this approach takes a list decoder for any short, poly-logarithmic-round, protocol and transforms it into a list decoder for a long, n-round, protocol while (essentially) maintaining the same error rate and list size but significantly improving over the communication and computational complexity and failure probability:

Theorem I.5 (simpler version of Theorem IV.1)**.** *Suppose there is a list-decodable coding scheme for any $\Theta(\log^2 n)$-round protocol that tolerates an error rate of ρ with failure probability $o(1)$ using $O(R\log^2 n)$ rounds, a list size of s and computational complexity $O(T\log^2 n)$. Then, for any $\epsilon > 0$, there is a list-decodable coding scheme for any n-round protocol that tolerates an error rate of $\rho - \epsilon$ with failure probability $2^{-\Omega(n)}$ using $O(Rn)$ rounds, a list size of $O(s/\epsilon)$ and computational complexity $O(Tn\log^{O(1)} n)$.*

Our boosting is inspired by ideas of Brakerski and Kalai [2] and Brakerski and Naor [3], which achieve computationally efficient unique decoders. The general approach is to protect the protocol only in (poly-)logarithmic size blocks, which can be done efficiently, and then use hashing to ensure progress and consistency between blocks. The method in [2] sacrifices a factor of 4 in the tolerable error rate and that of [3], which makes the computation near-linear time, sacrifices an additional unspecified constant factor. See the remark in [3, Section 3.1] for discussions.

Our boosting loses only a small additive ϵ in the tolerable error rate. The intuitive reason for this difference is as follows: In [2], one factor of 2 is lost because the algorithm tries to uniquely decode blocks. This allows the adversary to corrupt any blocks by corrupting only half of the transmissions during this block. In our boosting, this is circumvented by using list decoding. This also allows us to apply boosting recursively which further enables the use of block sizes that are poly-logarithmic instead of logarithmic as in [2], [3]. The second factor of 2 is lost for trying to keep the pointers of both parties consistent while they are following the correct path. As such, these pointers can only move in constant block size steps and any step in the wrong direction costs two good steps (one back and one forward in the right direction). These (ideally) lock-step moves are conceptually close to the approach of Schulman [5]. Our boosting instead is closer to the approach of Braverman and Rao [1]; it continuously adds new blocks and in each step tries to interactively find what the best starting point for an extension is. This interactive search is also protected using the same list decoder. Being able to have poly-logarithmic block sizes, instead of logarithmic, proves important in this interactive search.

ORGANIZATION

The rest of this paper is organized as follows: Section II provides the formal definitions of interactive coding setting and its different variations. In Sections III and IV we present our *reduction* and *boosting* results. The boosting in Section IV leads to coding schemes with an $\tilde{O}(n^2)$ computational complexity. A more involved boosting which leads to an $\tilde{O}(n)$ computational complexity is presented in the full version [17].

II. INTERACTIVE CODING SETTINGS

In this section, we define the interactive coding setup and summarize the different interactive coding settings considered in [4]. We also define the new one-sided decoding setting which is introduced in this work for the first time. We defer an in-depth discussion of the motivation and results for this new setting to [17] and provide here only its definition.

We mainly adopt the terminology from [4]: An n-round *interactive protocol* Π between two players Alice and Bob is given by two functions Π_A and Π_B. For each *round* of communication, these functions map (possibly probabilistically) the history of communication and the player's private input to a decision on whether to listen or transmit, and in the latter case also to a symbol of the *communication alphabet* Σ. For *non-adaptive* protocols the decision of a player to listen or transmit deterministically depends only on the round number and ensures that exactly one party transmits in each round. In this case, the *channel* delivers the chosen symbol of the transmitting party to the listening party, unless the adversary interferes and alters the symbol arbitrarily. In the adversarial channel model with *error rate* ρ, the number of such errors is at most ρn. For *adaptive* protocols the communicating players are allowed to base their decision on whether to transmit or listen (probabilistically) on the complete communication history (see [4] for an in-length discussion of this model). This can lead to rounds in which both parties transmit or listen simultaneously. In the first case no symbols are delivered while in the latter case the symbols received by the two listening parties are chosen by the adversary, without it being counted as an error. The

outcome of a protocol is defined to be the transcript of the interaction.

Robust Simulation: A protocol Π' is said to *robustly simulate* a protocol Π for an error rate ρ if the following holds: Given any inputs to Π, both parties can *uniquely decode* the transcript of an error free execution of Π on these inputs from the transcript of any execution of Π' in which at most a ρ fraction of the transmissions were corrupted. This definition extends easily to *s-list decoding* by allowing both parties to produce a list of s transcripts that is required to include the correct decoding, i.e., the transcript of Π. Another natural extension is the *one-sided decoding* in which only one of the two parties is required to decode. For a *one-sided decoding* interactive coding setting we assume that the party to decode is fixed and known a priori. We also consider *randomized protocols* in which both parties have access to independent private randomness. We say a randomized protocol robustly simulates a protocol Π with *failure probability* p if, for any input and any adversary, the probability that the parties correctly (list) decode is at least $1 - p$. We remark that in all settings the protocol Π' typically uses a larger alphabet Σ' and a larger number of rounds N. Throughout this paper we only consider protocols with constant size alphabets. We furthermore denote with a *coding scheme* any algorithm that given oracle access to Π gives oracle access to Π'. We denote with the *computational complexity* of a coding scheme the number of computation steps performed over N accesses to Π' assuming that each oracle access to Π is a one step operation.

Canonical Form: A non-adaptive protocol is called *balanced* if in any execution both parties talk equally often. We say a balanced protocol is of *canonical form* if it is over binary alphabet and the two parties take turns sending. Any (non-adaptive) protocol with $m = O(n)$ rounds over an alphabet with size $\sigma = O(1)$ can be transformed to a protocol of canonical form with at most $O(m \log \sigma) = O(n)$ rounds which is equivalent when used over an error free channel. We therefore assume that any protocol Π to be simulated is an n-round protocol of canonical form. To define the protocol Π, we take a rooted complete binary tree $\mathcal{T}$ of depths n. For each node, one of the edges towards children is *preferred*, and these *preferred* edges determine a unique path from the root to a leaf. The set $\mathcal{X}$ of the preferred edges at odd levels is given to Alice as input and the set $\mathcal{Y}$ of the preferred edges at even levels is given to Bob. The output of the protocol is the unique path $\mathcal{P}$, called the *common path*, from the root to a leaf formed by following the preferred edges. The protocol succeeds if both Alice and Bob learn the common path $\mathcal{P}$.

III. Reducing Unique Decoding to List Decoding

For the rest of this paper, we consider the standard information theoretic setting in which the adversary is computationally unbounded.

In this section, we show how to use a list decodable interactive coding scheme to build equally-efficient coding schemes for adaptive or non-adaptive unique decoding and also one-sided unique decoding.

A. Results

We start with the non-adaptive unique-decoding, which is the more standard setting:

Theorem III.1. *For any constant $\epsilon > 0$, given a balanced list decodable coding scheme with constant list size that tolerates error-rate $1/4 - \epsilon$, we get a (non-adaptive) unique decodable coding scheme that tolerates error-rate $1/4 - 2\epsilon$ with asymptotically the same round complexity, alphabet size and computational complexity.*

Theorem III.1 is interesting because of two somewhat curious aspects: (a) As list decoding is a strictly weaker guarantee than unique decoding, this theorem shows that one can get the uniqueness of the decoding essentially for free in the non-adaptive setting. (b) This theorem takes a list decoder that tolerates a suboptimal error-rate—as it is known that list decoders can tolerate error-rate $1/2 - \epsilon$—and generates a non-adaptive unique decoder which tolerates an optimal error-rate.

Next, we present the reduction for the adaptive unique-decoding setting:

Theorem III.2. *For any constant $\epsilon > 0$, given a balanced list decodable coding scheme with constant list size that tolerates error-rate $1/2 - \epsilon$, we get an adaptive unique decodable coding scheme that tolerates error-rate $2/7 - 2\epsilon$ with asymptotically the same round complexity, alphabet size and computational complexity.*

We next present the reduction for the newly introduced setting of one-sided unique decoding, where only one party—which is determined a priori—has to uniquely decode:

Theorem III.3. *For any constant $\epsilon > 0$, given a balanced list decodable coding scheme with constant list size that tolerates error-rate $1/2 - \epsilon$, we get a (non-adaptive) one-sided unique decodable coding scheme that tolerates error-rate $1/3 - 2\epsilon$ with asymptotically the same round complexity, alphabet size and computational complexity.*

Note that the $1/3 - \epsilon$ error-rate that can be tolerated by Theorem III.3 is larger than the $2/7$ error rate of the more standard two-sided setting (or $1/4$ if protocols are not allowed to be adaptive), in which both parties have to decode uniquely. This means that this slightly weaker decoding requirement, which might be all that is needed in some applications, allows for a higher error tolerance. This makes one-sidedness a useful distinction. We also remark that the $1/3$ tolerable error rate is optimal (see [17]).

Lastly, we also explain that using the same techniques, we can reduce the list size of the list decodable coding schemes to $O(1/\epsilon^2)$:

Theorem III.4. *For any constant $\epsilon > 0$, given a balanced list decodable coding scheme for n-round protocols that tolerates error-rate $1/2-\epsilon$ with list size s and round complexity $N' = \Omega(ns/\epsilon)$, we get a balanced list decodable coding scheme that tolerates error-rate $1/2 - 2\epsilon$, with constant list size $s' = O(\frac{1}{\epsilon^2})$ and round complexity $O(N')$, while having asymptotically the same alphabet size and computational complexity.*

B. Coding Schemes

In this section we describe our coding scheme which underlie the reductions stated in Theorems III.1, III.2, and III.3. All three reductions are built in a very similar manner and can be viewed as following a common template. We describe this coding scheme template in Section III-B1 and then give the concrete instantiations for each of the coding schemes in Section III-B2.

1) The Template of the Reductions: **Parameters and Structure**: We denote the n-round protocol to be simulated with Π, and we assume that Π is in the canonical form. We further denote with Π' the balanced list decoder coding scheme that we assume to exist, which robustly simulates Π and we use N', ρ' and Σ' to respectively denote the number of rounds, the tolerable error-rate, and the alphabet of Π'. As Π' is balanced, each party transmits for $N'/2$ rounds during Π'. We denote with Π'' the new coding scheme that achieves unique decoding and we use N'', ρ'' and Σ'' to respectively denote the number of rounds, the tolerable error-rate, and the alphabet of Π''.

We partition the N'' rounds of Π'' into two parts: The first part consists of $b_1 N'$ rounds, which are grouped into b_1 *blocks* of N' rounds each. This part is called the *joint part* and the blocks are called *joint blocks*. The second part consists of $b_2 N'/2$ rounds, which are grouped into b_2 blocks, consisting of $N'/2$ rounds each. This part is called the *exclusive part* and the blocks in it are called *exclusive blocks*. During the joint blocks, Alice and Bob send equally often. In the exclusive part, only one party is supposed to talk. Which party talks in the exclusive part is either agreed upon in advance (as for Theorem III.3) or decided adaptively (as for Theorem III.2).

Encoding: During the protocol Π'', Alice and Bob respectively maintain sets $E_A \subset \mathcal{X}$ and $E_B \subset \mathcal{Y}$, which are subsets of their *preferred edges* (see the *the canonical form* paragraph in Section II for the definitions). In the beginning of the simulation Π'', these edge-sets are initialized to be empty. Intuitively, these edge-sets correspond to the set of edges Alice and Bob believe could be on their *common path*.

In each joint block Alice and Bob run the list-decodable simulation Π' and obtain a list of s potentially correct common paths. Each party first discards obviously incorrect paths from its list (those containing non-preferred edges owned by themselves) and then adds all owned edges from all remaining paths to its respective edge-set E_A or E_B. The size of these edge-sets increases therefore by at most sn edges per block for a total of at most $b_1 sn$ edges. The edges furthermore form a tree, that is, for every edge all the ancestor edges owned by the same party are included as well. This allows one to encode each such edge-set using $4(b_1 sn)$ bits, because the number of size $b_1 sn$ subtrees of the complete binary tree is at most $2^{4(b_1 sn)}$.

In addition to running the list decoder in each block and adding edges to the sets E_A and E_B (which we refer to as E-sets), both parties also send their current E-sets to each other using error correcting codes. At the beginning of each block, both parties encode their current E-set into a codeword consisting of $N'/2$ symbols from an alphabet of size $\sigma_{ECC} = O(1/\epsilon)$ using an error correcting code with relative distance of $1-\epsilon$. This is where the assumption of $N' = \Omega(ns/\epsilon)$ comes in, as then $N'/2$ is large enough that can contain an error-correcting coded version of messages of length $4b_1 ns$ with relative distance $1-\epsilon$. During the block they add this codeword symbol by symbol to their transmission leading to the output alphabet size being $[\sigma_{ECC}] \times \Sigma'$. In the exclusive part, the party that is speaking uses the $N'/2$ rounds of each block to send the codeword of its (final) E-set symbol by symbol over the same output alphabet.

Decoding: All decoding decisions only rely on the received possibly-corrupted codewords. We describe how the decoding works for Alice; Bob's decoding procedure is the same. For every i, Alice combines the symbols received from Bob during block i to the string x_i. Without any channel corruptions x_i would correspond to the codeword encoding the set E_B at the beginning of block i. Alice decodes x_i to the closest codeword $\hat{x_i}$ which corresponds to the edge-set $\hat{E_i}$ and assigns this decoding a *confidence* c_i is defined as $c_i = 1 - \frac{2\Delta(x_i, \hat{x_i})}{N'/2}$, where $N'/2$ is the length of error-correcting code. Alice then combines $\hat{E_i}$ with all preferred edges she owns and determines whether these edges together give a unique path. If so Alice calls this path $\hat{\tau_i}$ and otherwise she sets $\hat{\tau_i} = \emptyset$. Given a set of decoded paths $\hat{\tau_1}, \hat{\tau_2}, \ldots$ and their confidences $c_1, c_2, \ldots$ we denote for any path τ its confidence with $c(\tau) = \sum_{i:\hat{\tau_i}=\tau} c_i$ and define the majority path $\tau_{\max}$ to be the non-empty path that maximizes this confidence. Lastly we define the combined confidence C as $C = \sum_i c_i$.

For Theorem III.4, the decoding procedure is slightly different: In each block, Alice list-decodes x_i to the $L = O(1/\epsilon)$ closest codewords $\hat{x}_i^1, \ldots, \hat{x}_i^L$ which respectively correspond to edge-sets $\hat{E}_i^1, \ldots, \hat{E}_i^L$, and thus paths $\hat{\tau}_i^1, \ldots, \hat{\tau}_i^L$. All these paths are output in the end of the algorithm.

2) Setting the Parameters for the Reductions: We now describe with what parameters the template from Sec-

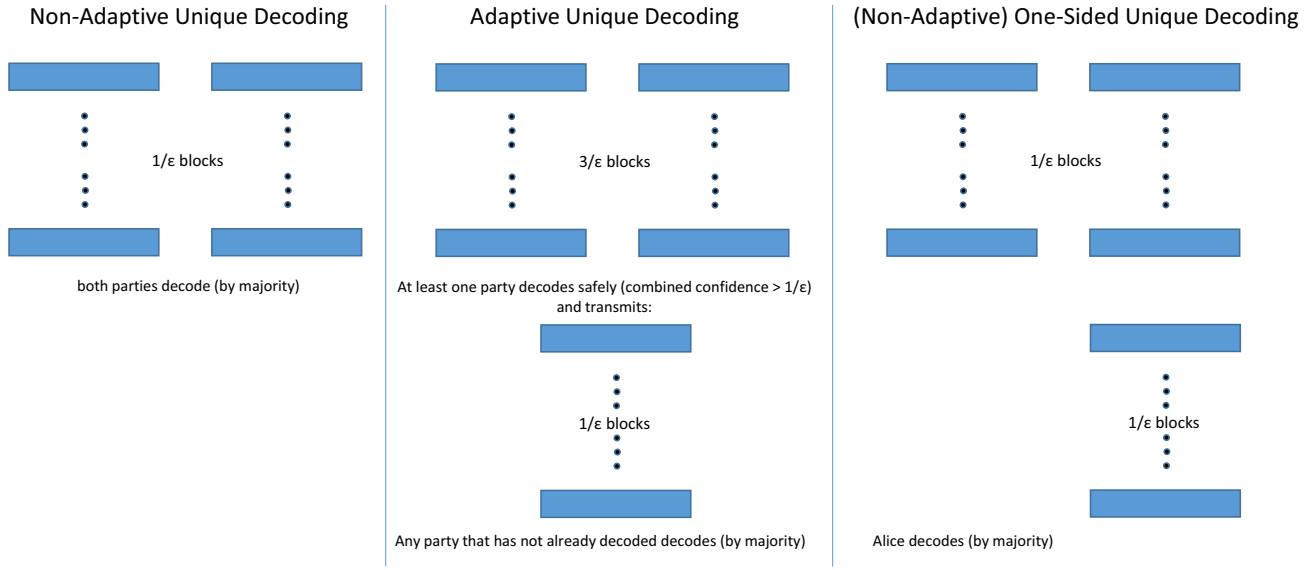

Figure 1. The instantiations of the coding scheme template from Section III-B1 for the three settings considered in Section III-A, namely Theorem III.1, Theorem III.2, Theorem III.3. The instantiation for Theorem III.4 is similar to that of Theorem III.1.

tion III-B1 is employed to lead to the three coding scheme claimed in Section III-A:

For Theorem III.1, we use a very simple version of the template from Section III-B1, in particular, we do not use the exclusive part. The complete protocol consists $N'' = \frac{1}{\epsilon}N'$ rounds, all in the joint part, which are grouped into $b_1 = \frac{1}{\epsilon}$ joint blocks. To decode both parties simply output the majority path in the end.

For Theorem III.2, we use the template from Section III-B1 with $N'' = \frac{3.5}{\epsilon}N'$ rounds which are grouped into $b_1 = \frac{3}{\epsilon}$ joint blocks and $b_2 = \frac{1}{\epsilon}$ exclusive blocks. After the joint part both parties determine the quantity $C' = 2(c(\tau_{\max}) + c(\emptyset)) - C$ and declare the majority path a safe decoding if $C' > 1/\epsilon$. For the exclusive part both parties make the following adaptive decision: If a party has safely decoded it will transmit in the exclusive part otherwise it will listen and declare the majority path as a decoding in the end.

For Theorem III.3, we use the template from Section III-B1 with $N'' = \frac{1.5}{\epsilon}N'$ rounds which are grouped into $b_1 = \frac{1}{\epsilon}$ joint blocks and $b_2 = \frac{1}{\epsilon}$ exclusive blocks. Assuming that Alice is the party which is interested in decoding in the end, during the exclusive blocks Alice will be listening while Bob is transmitting. To decode Alice outputs the majority path in the end.

For Theorem III.4, we use the template from Section III-B1 with no exclusive part: $N'' = \frac{1}{\epsilon}N'$ rounds, all in the joint part, which are grouped into $b_1 = \frac{1}{\epsilon}$ joint blocks. At the end, each party outputs all the paths created in the decoding procedure, which are $s' = O(1/\epsilon^2)$ many.

C. Analysis

As a sample, here we present the analysis for the non-adaptive setting, i.e., Theorem III.1. The proofs of the other reductions are deferred to [17]. In particular, the analysis of the adaptive setting is considerably more involved.

Proof of Theorem III.1: As described in Section III-B2 we use the template from Section III-B1 with $b_1 = 1/\epsilon$ joint blocks. We show that for the correct path τ and for both parties the inequality $c(\tau) > (C - c(\tau) - c(\emptyset))$ holds. This means that for both parties the confidence into the path τ is larger than the confidence in all other (non-emtpy) paths combined. This also implies that the majority path $\tau_{\max}$ is a correct decoding for both parties.

To prove this we fix one party, say Alice, consider the quantity $C' = c(\tau) - (C - c(\tau) - c(\emptyset))$ for her, and analyze the contribution of each block towards this quantity. We split the execution into two parts according to the first block in which the list decoder succeeded and prove the claim for both parts separately. In particular we define i^* to be the first block at which the list decoder succeeded, that is, the first block after which the edge set $E_A \cup E_B$ contains the common path $\mathcal{P}$ of Π. We claim that the contribution towards C' of any block $i \neq i^*$ is at least $1 - 4e_i - 4\epsilon$ where e_i is the fraction of transmissions with an error in block i.

We first prove this claim for block $i > i^*$. In these blocks the codeword transmitted by Bob corresponds to the correct path τ. Since the error correcting code employed has a distance of at least $1-\epsilon$ we get that Alice correctly decodes to τ if less than a $1/2-\epsilon/2$ fraction of Bob's transmissions are corrupted. The confidence c_i of this block then contributes positively to C'. It furthermore holds that $c_i = 1 - 2e_A$ where e_A is fraction of errors on Alice which is at most

twice the fraction of errors e_i in this block. This makes the contribution of block i at least $1 - 4e_i > 1 - 4e_i - 4\epsilon$ as desired. For the case that more than a $1/2 - \epsilon/2$ fraction of the transmissions to Alice are corrupted she might decode to a different path which makes the confidence of this block contribute negatively to C'. We still get that the contribution $-c_i$ is at least $-(1 - 2(1 - \epsilon - e_A)) > 1 - 2\epsilon - 4e_i$.

We also need to show that our claim holds for blocks $i < i^*$. The analysis for these blocks is the same except that the codeword sent out by Bob might correspond to the empty path. If few transmissions towards Alice get corrupted she might decode to $\hat{\tau}_i = \emptyset$ which leads to a zero contribution towards C'. We need to verify that in this case we do not claim a positive contribution. What saves us is that for any block $i < i^*$ it holds that $e_i > 1/4 - \epsilon$ since otherwise the list decoder would have been successful in block i. This means our assumed contribution of $1 - 4e_i - 4\epsilon$ is never larger than zero which completes the proof of the claim for all blocks $i \neq i^*$.

Finally, using the claim, the assumption that the global fraction of errors e_{ave} is at most $\frac{1}{4} - 2\epsilon$, and summing over the contributions of all blocks we get:

$$\begin{aligned} C' &\geq \sum_{i \neq i^*} (1 - 4\, e_i - \frac{4}{\epsilon}) - |c(i^*)| \\ &> (b_1 - 1) - 4 \sum_i e_i - \frac{4b_1}{\epsilon} - 1 \\ &= (\frac{1}{\epsilon} - 1) - 4\, b_1\, e_{ave} - 4 - 1 \\ &\geq \frac{1}{\epsilon} - \frac{4}{\epsilon}\left(\frac{1}{4} - 2\epsilon\right) - 6 = 2 > 0. \end{aligned}$$

As desired, this shows that an error rate of at most $\frac{1}{4} - 2\epsilon$ results in both parties recovering the correct path τ by choosing the majority path. ∎

IV. Boosting List-Decoders

In this section, we present a generic *boosting* approach for improving the efficiency of list decoders. In particular, the boosting we present here improves the *round complexity* (blow up) of the list decoders and also leads to *computationally efficient* list-decoders, even from list-decoders with, e.g., exponential computational complexity. More concretely, as the basic boosting step, we explain that assuming a list-decoder coding scheme for $O(\log^2 n)$-round protocols, we can create a list-decoder coding scheme for n-round protocols with round complexity blow up similar to that of the $O(\log^2 n)$-rounds protocol and near-cubic computational complexity. A more advanced version with near-linear computational complexity appears in [17]. We explain in [17] how to recursively apply this boosting to get efficient list-decoders and then combine them with the reduction results to prove Theorems I.1 and I.2. For ease of readability we will use $\tilde{O}$-notation to hide $\log^{O(1)} n$ factors.

A. Basic Boosting Step: From Poly-logarithmic Protocols to Linear Protocols

Here we show how to boost any list-decoder for protocols with $O(\log^2 n)$ rounds to a list-decoder for protocols with n rounds, while loosing only an additive ϵ' term in the tolerable error rate and $\frac{1}{\epsilon'}$ factors in the round complexity and list size. More formally, we prove the following:

Theorem IV.1. *For any failure-exponent $C = \Omega(1)$, any $C' = \Omega(C)$, and any error-rate loss ϵ' such that $2\log\frac{5}{\epsilon'} \leq C\log^2 n$, the following holds: Suppose there is a list-decodable coding scheme that robustly simulates any $C'\log^2 n$-round protocol, while tolerating error rate ρ, and such that it has list size $s = \tilde{O}(1)$, round complexity $RC'\log^2 n$, computational complexity T, and failure probability at most $2^{-C\log^2 n}$. Then, there exists a randomized list decoding coding scheme for n-round protocols that tolerates error rate $\rho - \epsilon'$ and has list size $s' = O(\frac{s}{\epsilon'})$, round complexity $O(\frac{RC'}{\epsilon'}\cdot n)$, computational complexity $\tilde{O}(\frac{n^3}{\epsilon'^2}\cdot T)$, and failure probability 2^{-Cn}.*

For simplicity, the reader might think of C and C' as large enough constants. Furthermore, the condition $2\log\frac{5}{\epsilon'} \leq C\log^2 n$ is a technical condition that is needed for the generality of this boosting but it is readily satisfied in all applications of interest in this paper.

Remark about the Computational Complexity of Theorem IV.1: For simplicity, in this section we present a boosting step that has computational complexity of $\tilde{O}(n^3)$. In [17], we present a more advanced version which has a computational complexity of $\tilde{O}(n)$.

Proof Outline of Theorem IV.1: Let Π be the original n-rounds protocol in the canonical form (see Section II), let $\mathcal{T}$ be its binary tree in the canonical form, and let E^{odd} and E^{even} respectively represent the edges of $\mathcal{T}$ starting from odd and even levels. Furthermore, let $\mathcal{X} \subset E^{odd}$ and $Y \subset E^{even}$ respectively be the *preferred edges* inputs of Alice and Bob. Finally, let $\mathcal{P}$ be the *common path* in $\mathcal{X} \cup \mathcal{Y}$.

The new coding scheme runs in $N = \frac{10\,RC'}{\epsilon'}\cdot n$ rounds. These rounds are partitioned into $N' = \frac{10}{\epsilon'}\cdot\frac{n}{\log^2 n}$ *meta-rounds*, each of length $RC'\log^2 n$ rounds. Furthermore, we break Π into *blocks* of length $\log^2 n$ rounds. In the simulation Π' of Π, Alice and Bob always maintain edge-sets $\bar{E}_A$ and $\bar{E}_B$, respectively, which are rooted sub-trees of $\mathcal{T}$ and such that we have $\bar{E}_A \cap E^{odd} \subseteq \mathcal{X}$ and $\bar{E}_B \cap E^{even} \subseteq \mathcal{Y}$. Hence, always $\bar{E}_A \cap \bar{E}_B$ is a rooted sub-path of the common path $\mathcal{P}$. Initially, we have $\bar{E}_A = \bar{E}_B = \emptyset$. In the course of the simulation, we grow the edge-sets $\bar{E}_A$ and $\bar{E}_B$ by adding at most s many blocks per meta-round. If a block added to $\bar{E}_A$ ends at a leaf of $\mathcal{T}$, then Alice adds a vote to this leaf, and Bob does similarly with respect to $\bar{E}_B$. We show that, at the end, if the total error-rate is less than $\rho - \epsilon'$, then for both Alice and Bob, the leaf of the common path is among the $s' = O(\frac{s}{\epsilon'})$ many leaves with the most votes.

Algorithm 1 Boosting List-Decoder, at Alice's Side

$\mathcal{X} \leftarrow$ the set of Alice's preferred edges;
$\bar{E}_A \leftarrow \emptyset$;
$N' = \frac{10}{\epsilon'} \cdot \frac{n}{\log^2 n}$;
for $i = 1$ to N' **do**
  Simulate the following $C' \log^2 n$-rounds protocol in $R \cdot C' \log^2 n$ rounds:
    Search Phase: `Find the deepest common path in` $\bar{E}_A \cap \bar{E}_B$`, let it be` P'`.`
    Path-Extension Phase: `Execute the protocol on the block extending` P'`.`
  $S \leftarrow s$ list-decodings of possible original outcomes of this protocol
  for each outcome $\sigma \in S$ **do**
    $B \leftarrow$ block executed in the path-extension phase of σ
    if B is a path in $\mathcal{T}$ and $B \cap E^{odd} \subset \mathcal{X}$ **then**
      $\bar{E}_A \leftarrow \bar{E}_A \cup \{B\}$
      if B ends at a leaf v **then**
        $v.vote \leftarrow v.vote + 1$
Output the $O(s/\epsilon')$ leaves with the most votes

Ideally, we would like each meta-round to simulate one block and if the error-rate is at most ρ, then this meta-round should make a progress of length $\log^2 n$ along the common path $\mathcal{P}$. That is, we would like that in each meta-round in which error-rate is at most ρ, $|(\bar{E}_A \cap \bar{E}_B) \cap P|$ grows by one block. However, realizing this ideal case faces one important challenge: in each meta-round, we cannot be sure of the correctness of the past blocks as the adversary could have corrupted them by investing enough errors. To remedy this issue, in each meta-round, the two parties first try to find the deepest block that has been computed correctly in the past; we call this the *search phase*. Then the two parties simulate the next block extending from there downwards on $\mathcal{P}$; we call this the *path-extension phase*. The search phase takes $\Theta(C \log^2 n)$ rounds while the path-extension phase takes $\log^2 n$ rounds. We choose the constants such that the total number of communications in search phase plus that of the path-extension phase is at most $C' \log^2 n$ rounds. This is doable because of the condition $C' = \Omega(10C)$ in the statement of the lemma. Then, these $C' \log^2 n$ rounds of communication are wrapped in (and thus protected via) the list decodable coding scheme of $C' \log^2 n$ rounds, in the $RC' \log^2 n$ rounds of the meta-round. What we do on top of this list-decoder in each meta-round is as follows: for each meta-round, there are at most s suggested transcripts. The parties add the extension blocks of these s transcripts to their edge-sets $\bar{E}_A$ and $\bar{E}_B$ (but of course only if the block is consistent with the party's own local input $\mathcal{X}$ or $\mathcal{Y}$, otherwise the block gets discarded). Furthermore, for each of the s transcripts, there is one path which is found in the search phase. If this path ends at a leaf, we add one *vote* to this leaf. At the end of the whole simulation, each party outputs the $s' = O(\frac{s}{\epsilon'})$ leaves with the most votes. A pseudocode is presented in Algorithm 1.

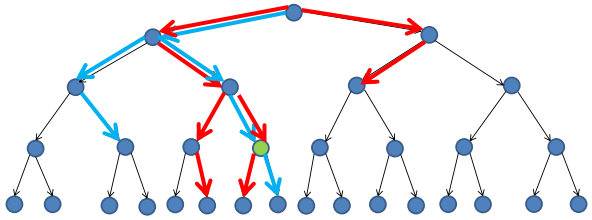

Figure 2. The Tree-Intersection Problem

In the above sketch, we did not explain how to solve the search phase. We abstract this phase as a new two party communication complexity problem over a noiseless channel, which we call the *tree-intersection problem*, and discuss in Section IV-A1. In particular, we explain how this problem can be solved in $O(\log^2 n)$ rounds with failure probability $1 - 2^{C \log^2 n}$, and with computational complexity of $\tilde{O}(n)$. Having solved the search phase, we complete the proof with a few simple arguments in Section IV-A2. In particular, we show that in each meta-round in which the error-rate is less than ρ, with probability at least $1 - 2^{C \log^2 n}$, either $|(\bar{E}_A \cap \bar{E}_B) \cap P|$ grows by at least one block or if $|(\bar{E}_A \cap \bar{E}_B) \cap P|$ already contains a leaf, then this leaf receives one more vote. We then show that with probability at least $1 - 2^{-Cn}$, the leaf at the end of the common path $\mathcal{P}$ receives at least $\Theta(N'\epsilon')$ votes. On the other hand, each of $\bar{E}_A$ and $\bar{E}_B$ can contain a total vote of at most $N' \cdot s$. Hence, we get that the correct path is among the $s' = O(\frac{s}{\epsilon'})$ leaves with the most votes, with probability at least $1 - 2^{-Cn}$. ■

1) The Tree-Intersection Problem:

Definition IV.2. ***(The Tree-Intersection Problem)*** *Suppose that Alice and Bob respectively have edge sets $\bar{E}_A$ and $\bar{E}_B$ that correspond to subtrees of a complete binary tree $\mathcal{T}$ of depth n rooted at the root of $\mathcal{T}$, and that $|\bar{E}_A| \leq M$ and $|\bar{E}_B| \leq M$, where $M = \tilde{O}(n)$. Now, given the promise that $P = \bar{E}_A \cap \bar{E}_B$ is a path, Alice and Bob want to recover the path P with as little communications over a noiseless binary channel as possible, while failing to so only with negligible probability.*

Figure 2 shows an example of this problem where edges in $\bar{E}_A$ and $\bar{E}_B$ are indicated with blue and red arrows, respectively, and the path P is the path from the root to the green node.

We present a simpler $O(C \log^2 n)$ rounds solution, in Lemma IV.3, which explains the main approach but has failure probability at most $2^{-\Omega(C \log n)}$. In [17], we give an improved version with similar round complexity but a failure probability of at most $2^{-\Omega(C \log^2 n)}$.

Lemma IV.3. *For any C, there is a tree-intersection protocol which uses $O(C \log^2 n)$ rounds of communication on a binary channel, $O(C \log n)$ bits of randomness, and*

polynomial-time computation, and finds path P with failure probability at most $2^{-C\log n}$. This protocol has computation complexity $\tilde{O}(CM)$.

Proof: Alice samples $\Theta(C\log n)$ random bits and sends them to Bob. This defines a hash function $h : \{0,1\}^n \to \{0,1\}^{\Theta(C\log n)}$. Choosing large enough constants, we have that the probability that there are two different paths (among the paths of Alice and Bob) that have equal hash-values is at most $M^2 \cdot 2^{-\Theta(C\log n)} \leq 2^{-C\log n}$, where the M^2 is for a union bound over all pairs. In the rest, we assume that no two different paths have equal hash values and we use this hash in a binary search for finding the intersection $P = \bar{E}_A \cap \bar{E}_B$.

Alice finds an edge $e \in \bar{E}_A$ that cuts her edge-set $\bar{E}_A$ in two "semi-balanced" parts, each containing at least $|\bar{E}_A|/3$ edges. That is, an edge e such that the following holds: let $\mathcal{T}_e$ be the subtree of $\mathcal{T}$ below edge e. Then, edge e should satisfy $|\bar{E}_A|/3 \leq |\bar{E}_A \cap \mathcal{T}_e| \leq 2|\bar{E}_A|/3$. Note that such an edge e exists and also can be found in $\tilde{O}(CM)$ time. Once Alice finds such an edge e, she then sends $h(P_e)$ to Bob, where $h(P_e)$ is the hash-value of the path P_e starting from the root and ending with edge e. Bob checks whether he has a path with the same hash-value $h(P_e)$ and reports the outcome to Alice by sending one bit. If there is a path with matching hash value, then e is construed to belong to the common path. Otherwise, if there is no such path with matching hash-value, this is construed as e not belonging to the common path. In either case, Alice can discard at least a $1/3$ fraction of $\bar{E}_A$. This is because, if e is not on the common path, then every edge in $\bar{E}_A \cap \mathcal{T}_e$ can be discarded. On the other hand, if e is on the common path, then we are sure that the path starting from the root and ending with e is in the common path. Thus, edges on this path can be also ignored from now on as certainly being on the path and the remaining problem is to only solve the tree-intersection in T_e. Note that any edge that diverges from P_e before e gets discarded as well as it cannot be on the common path P.

Iterating the above step $\log_{3/2} n$ times leads to Alice finding the common path. Alice can then report this path to Bob by just sending the related hash value. The whole procedure succeeds if the hash-values of different paths in $\bar{E}_A \cup \bar{E}_B$ are different which as discussed before happens with probability at least $1 - 2^{-C\log n}$. ∎

To reduce the failure probability to $2^{-\Omega(\log^2 n)}$, the key change is that we use a probabilistic binary search approach instead of the deterministic binary search used above. The main point is to try to cover for the possibility that each hash-value checking step can fail with probability $2^{-\Theta(\log n)}$ by allowing backtracking in the binary search. We note that getting this better $2^{-\Omega(\log^2 n)}$ failure probability, that is a failure probability that is exponential in the communication complexity of the tree-intersection solution, is most interesting for our non-uniform deterministic coding schemes, which are given in [17].

2) Completing the Basic Boosting Step: We now complete the proof of Theorem IV.1. We first show that for each meta-round with small error-rate, this meta-round either makes a block of progress on the common path or it adds a vote to the leaf at the end of the common path.

Lemma IV.4. *In each meta-round in which error-rate is at most ρ, with probability at least $1 - O(2^{-C\log^2 n})$, either $|\bar{E}_A \cap \bar{E}_B \cap \mathcal{P}|$ increases by $\log^2 n$ or one vote is added to the leaf at the end of $\mathcal{P}$, on both of Alice and Bob's sides.*

Proof: Note that in the absence of errors, each meta-round would with probability at least $1-2^{-2C\log^2 n}$ find the deepest path in $\bar{E}_A \cap \bar{E}_B$ and then extend it by one block along $\mathcal{P}$ (if it already does not end in a leaf). The list-decoding coding scheme for $C'\log^2 n$ round protocols provides the following guarantee: if the error-rate in this meta-round is at most ρ, with probability at least $1-2^{-2C\log^2 n}$, we get a list of s possible transcripts of this $\log^2 n$ round protocol, one of which is correct. In the algorithm, we add all of the s possible new blocks, one for each transcript, to $\bar{E}_A$ and $\bar{E}_B$, and also if the blocks end at a leaf, we add one vote to the respective leaf. Hence, if the meta-round has error-rate at most ρ, with probability at least $1-O(2^{-2C\log^2 n})$, either $|\bar{E}_A \cap \bar{E}_B \cap \mathcal{P}|$ increases by one block or each of Alice and Bob add one vote to the leaf at the end of $\mathcal{P}$. ∎

Proof of Theorem IV.1: Let us call a meta-round bad if one of the following holds: (a) its error-rate is greater than ρ, (b) its error-rate is less than ρ but the parties neither make one block of progress along $\mathcal{P}$ together nor they both add a vote to the leaf at the end of $\mathcal{P}$. At most $\frac{\rho-\epsilon'}{\rho}$ fraction of the meta-rounds have error-rate greater than ρ. On the other hand, Lemma IV.4 tell us that in each meta-round with error-rate at most ρ, with probability at least $1 - O(2^{-2C\log^2 n})$, parties either both make one block of progress along $\mathcal{P}$ or both add a vote to the leaf at the end of $\mathcal{P}$. Thus, with probability at least $1-2^{-Cn}$, the number of bad meta-rounds in which error-rate is less than ρ is at most $\epsilon' N'/2$. This is because, the probability that there are more than $\epsilon' N'/2$ such meta-rounds is at most

$$\sum_{i=\epsilon' N'/2}^{N'} \binom{N'}{i} O(2^{-C\log^2 n})^i \leq \sum_{i=\epsilon' N'/2}^{N'} (\frac{5}{\epsilon'})^i \cdot 2^{-iC\log^2 n}$$
$$\leq \sum_{i=\epsilon' N'/2}^{N'} 2^{i(\log\frac{5}{\epsilon'} - C\log^2 n)} \leq 2^{-\frac{N'\epsilon'}{4}\cdot C\cdot\log^2 n},$$

which is less than or equal to 2^{-Cn}. Hence, with probability at least $1-2^{-Cn}$, the fraction of bad meta-rounds is at most $\frac{\rho-\epsilon'}{\rho} + \frac{\epsilon'}{2} \leq 1 - \epsilon'/2$. Therefore, there are at least $N' \cdot \frac{\epsilon'}{2}$ good meta-rounds. Note that each good meta-round either extends the common path along $\mathcal{P}$ by one block or adds a vote to the leaf at the end of $\mathcal{P}$. On the other hand, at most $\frac{n}{\log^2 n} \leq \frac{N'\epsilon'}{4}$ meta-rounds can be spent on extending the

common path along $\mathcal{P}$ by one block each. Hence, the leaf at the end of $\mathcal{P}$ receives at least $\frac{N'\epsilon'}{2} - \frac{N'\epsilon'}{4} \geq \frac{N'\epsilon'}{4}$ votes. On the other hand, each of $\bar{E}_A$ and $\bar{E}_B$ can contain at most $N' \cdot s$ votes. Therefore, with probability at least $1 - 2^{-Cn}$, the correct path is among the $O(\frac{s}{\epsilon'})$ leaves with the most votes. ■

V. Proving the End Results

Lastly, we explain how to combine the reduction technique of Section III with the list-decoders derived via boosting in Section IV to prove the main end results of this paper, namely Theorems I.1 and I.2 and Remark I.3.

Proof of Theorem I.1: To prove this theorem, we view the unique decoding coding scheme of Braverman and Rao [1] as a list-decoding that tolerates error rate $1/4-\epsilon/2$. Thus, we get a deterministic list decodable coding scheme, with list size 1, for any $O((\log\log\log n)^2)$-round protocol over a channel with alphabet size $O(1/\epsilon)$ and error-rate $1/4 - \epsilon/2$, round complexity $O((\log\log\log n)^2)$ and computational complexity $\tilde{O}(\log n)$. We then boost this list-decoder to a list-decoder for n-round protocols with communication complexity $N = O(n)$. As the result, we get a list-decoder for any n-round protocol that has round complexity $O(n)$, constant list size of $O(1/\epsilon^2)$, failure probability $2^{-\Theta(n)}$, and computational complexity $\tilde{O}(n)$. Then, we apply Theorem III.1, which gets us to a unique decoder that tolerates error-rate $1/4-\epsilon$ and thus finishes the proof of Theorem I.1. ■

Proof of Theorem I.2: Braverman and Efremenko [16] present a list-decodable coding scheme that tolerates error-rate $1/2 - \epsilon/2$ and has linear communication complexity and exponential compuational complexity. By recursively boosting this list-decoder, we get a randomized list-decodable coding scheme that robustly simulates any n-round protocol in $O(n)$ rounds tolerating error rate $1/2 - \epsilon$ with list size $O(1/\epsilon^2)$, computational complexity $\tilde{O}(n)$, and failure probability at most $2^{-\omega(n)}$. This already gives item (C) of Theorem I.2. Combining this list decoder with Theorems III.2 and III.3 provides items (A) and (B) of Theorem I.2 respectively. ■

The proof for Remark I.3 is similar to the proofs of Theorems I.1 and I.2 with the exception that we use our non-uniform deterministic list-decoder boosting procedures instead of the randomized ones. It is deferred to the full version [17].

Acknowledgements

We thank Madhu Sudan for helpful discussions in the preliminary stages of this work that took place while the second author was a consulting researcher at Microsoft Research New England.

References

[1] M. Braverman and A. Rao, "Towards coding for maximum errors in interactive communication," in *Proceedings of the ACM Symposium on Theory of Computing (STOC)*, 2011, pp. 159–166.

[2] Z. Brakerski and Y. Kalai, "Efficient Interactive Coding against Adversarial Noise," in *Proceedings of the IEEE Symposium on Foundations of Computer Science (FOCS)*, 2012, pp. 160–166.

[3] Z. Brakerski and M. Naor, "Fast algorithms for interactive coding," in *Proceedings of the ACM-SIAM Symposium on Discrete Algorithms (SODA)*, 2013, pp. 443–456.

[4] M. Ghaffari, B. Haeupler, and M. Sudan, "Optimal Error Rates for Interactive Coding I: Adaptivity and Other Settings," in *Proceedings of the ACM Symposium on Theory of Computing (STOC)*, 2014, pp. 794–803.

[5] L. J. Schulman, "Coding for interactive communication," *IEEE Transactions on Information Theory (TransInf)*, vol. 42, no. 6, pp. 1745–1756, 1996.

[6] D. A. Spielman, "Linear-time encodable and decodable error-correcting codes," *IEEE Transactions on Information Theory (TransInf)*, vol. 42, no. 6, pp. 1723–1731, 1996.

[7] P. Elias, "List decoding for noisy channels," MIT, Tech. Rep. 335, 1957.

[8] M. Sudan, "Decoding of Reed Solomon codes beyond the error-correction bound," *Journal of Complexity*, vol. 13, no. 1, pp. 180 – 193, 1997.

[9] V. Guruswami, "List decoding of error-correcting codes," Ph.D. dissertation, Massachusetts Institute of Technology, 2001.

[10] R. Gelles, A. Moitra, and A. Sahai, "Efficient and explicit coding for interactive communication," in *Proceedings of the IEEE Symposium on Foundations of Computer Science (FOCS)*, 2011, pp. 768–777.

[11] M. Braverman, "Towards deterministic tree code constructions," in *Proceedings of the Innovations in Theoretical Computer Science Conference (ITCS)*, 2012, pp. 161–167.

[12] G. Kol and R. Raz, "Interactive channel capacity," in *Proceedings of the ACM Symposium on Theory of Computing (STOC)*, 2013, pp. 715–724.

[13] B. Haeupler, "Interactive Channel Capacity Revisited," in *Proceedings of the IEEE Symposium on Foundations of Computer Science (FOCS)*, 2014.

[14] S. Agrawal, R. Gelles, and A. Sahai, "Adaptive protocols for interactive communication," in *arXiv:1312.4182*, 2014.

[15] M. Franklin, R. Gelles, R. Ostrovsky, and L. J. Schulman, "Optimal coding for streaming authentication and interactive communication," in *Proceedings of the International Cryptology Conference (CRYPTO)*, 2013, pp. 258–276.

[16] M. Braverman and K. Efremenko, "List and unique coding for interactive communication in the presence of adversarial noise," in *Proceedings of the IEEE Symposium on Foundations of Computer Science (FOCS)*, 2014.

[17] M. Ghaffari and B. Haeupler, "Optimal Error Rates for Interactive Coding II: Efficiency and List decoding," in *arXiv:1312.1763*, 2013.