Interactive Compression for Multi-Party Protocols

Gillat Kol[1], **Rotem Oshman**[*2], **and Dafna Sadeh**[†3]

1 **Princeton University, USA**
gillat.kol@gmail.com
2 **Tel Aviv University, Israel**
rotem.oshman@gmail.com
3 **Tel Aviv University, Israel**
dafnasadeh@mail.tau.ac.il

Abstract

The field of compression studies the question of how many bits of communication are necessary to convey a given piece of data. For one-way communication between a sender and a receiver, the seminal work of Shannon and Huffman showed that the communication required is characterized by the *entropy* of the data; in recent years, there has been a great amount of interest in extending this line of research to *interactive communication*, where instead of a sender and a receiver we have two parties communication back-and-forth. In this paper we initiate the study of interactive compression for distributed multi-player protocols. We consider the classical *shared blackboard model*, where players take turns speaking, and each player's message is immediately seen by all the other players. We show that in the shared blackboard model with k players, one can compress protocols down to $\tilde{O}(I \cdot k)$, where I is the information content of the protocol and k is the number of players. We complement this result with an almost matching lower bound of $\tilde{\Omega}(I \cdot k)$, which shows that a nearly-linear dependence on the number of players cannot be avoided.

1998 ACM Subject Classification E.4 Coding and Information Theory

Keywords and phrases interactive compression, multi-party communication

Digital Object Identifier 10.4230/LIPIcs.DISC.2017.31

1 Introduction

In their seminal work, Shannon, Fano and Huffman considered the *data compression problem*: a sender wants to send a message x to a receiver. We think of x as a random variable generated from some distribution μ. How many bits does the sender need to send, so that the receiver will be able to recover x with high probability? The answer given in [15, 9, 11] is that he needs to send only $\lceil \mathrm{H}(x) \rceil$ bits, in expectation, where H denotes Shannon's entropy function. Roughly speaking, this means that every message can be compressed to its information content.

While classical information theory studied the case of one-way transmission, over the last decades, *interactive* communication protocols were also studied extensively. The *interactive compression problem* [2] is the analog of the data compression problem in the interactive setting: it asks whether the transcript of any interactive protocol can be *compressed* to its information content. Roughly speaking, compressing a protocol Π means constructing

* Rotem Oshman is supported by the Israeli Centers of Research Excellence (I-CORE) program (Center No.4/11) and by BSF Grant No. 2014256.
† Dafna Sadeh is supported by the Israeli Centers of Research Excellence (I-CORE) program (Center No.4/11) and by BSF Grant No. 2014256.

a different protocol Π', hopefully with smaller communication, which computes the same function. The interactive compression problem for the two players setting attracted a lot of attention in recent years, resulting in several compression protocols for different settings [10, 2, 6, 4, 7, 3, 14, 12, 16].

Interactive compression can be viewed as a tool for protocol design: one first designs a communication-inefficient protocol, making sure only that it does not reveal a lot of information about the inputs; interactive compression then allows us to convert the protocol into a communication-efficient one.

In this work we initiate the study of *distributed interactive compression*. We show how to *compress* a given k-party communication protocol, in order to reduce its communication, and we also show the limitation of such compression schemes.

We study the *shared blackboard* model of multi-party communication. In this classical model, the players communicate over a "shared blackboard", taking turns to write messages on the board. All players can see the contents of the board, and the player whose turn it is to write next is determined by what was written so far. The model can be viewed as a single-hop radio network: when a player sends a message, the message is immediately received by all the other players.

Measuring information content. As discussed above, in the case of one-way communication, the information content of the data $\boldsymbol{D}$ we wish to compress is measured by its Shannon entropy, $\mathrm{H}(\boldsymbol{D})$. The analog of entropy for interactive communication is *information cost* [8, 1, 2], which measures how much information the players reveal about their inputs.

The precise notion we work with here is called *external information cost*: it measures, in mutual information, the amount of information an external observer learns about the players' inputs from observing the transcript of the protocol. Formally, if $\boldsymbol{X}_1, \ldots, \boldsymbol{X}_k$ are random variables denoting the inputs to the k players, sampled according to the joint distribution μ, and $\boldsymbol{\Pi}(\boldsymbol{X}_1, \ldots, \boldsymbol{X}_k)$ is a random variable denoting the transcript of the protocol Π when it is run with the inputs $\boldsymbol{X}_1, \ldots, \boldsymbol{X}_k$, then the external information cost of the protocol Π with respect to the distribution μ is defined as

$$\underset{\mu}{\mathrm{IC}}(\Pi) = \mathrm{I}(\boldsymbol{\Pi}(\boldsymbol{X}_1, \ldots, \boldsymbol{X}_k); \boldsymbol{X}_1, \ldots, \boldsymbol{X}_k),$$

where I denotes the mutual information, $\mathrm{I}(\boldsymbol{A}; \boldsymbol{B}) = \mathrm{H}(\boldsymbol{A}) - \mathrm{H}(\boldsymbol{A}|\boldsymbol{B}) = \mathrm{H}(\boldsymbol{B}) - \mathrm{H}(\boldsymbol{B}|\boldsymbol{A})$.

We mention that in the two-player case, another notion of information cost, called *internal* information cost, was studies extensively. This notions measures the amount of information that the players learn about *each other's* inputs from their interaction. Internal information cost differs from external information cost when the players' inputs are not independent, because in this case the players potentially know something about each other's inputs just by looking at their own inputs, while an external observer has no such prior information. However, it is unclear how to adapt the definition of internal information to the multi-player setting in a meaningful way (and indeed, this is an interesting open problem). In the sequel, when we say "information cost", we mean external information cost.

1.1 Our Results

1.1.1 A compression protocol in the shared blackboard model

We show that a protocol with information cost I can be compressed down to $\tilde{O}(I \cdot k)$ bits of communication, where the $\tilde{O}$-notation hides polylogarithmic factors in I, k, and the original communication cost of the protocol.

What does it mean to "compress" a protocol? As in the case of non-interactive data compression, we give a *compression scheme* and a *decoding function* dec. Given any protocol Π, we can apply the compression scheme to obtain a compressed protocol Π' (hopefully with less communication), and we can apply the decoder dec to Π''s transcripts (and its public randomness) to extract from them transcripts of Π. Our compression has some error: for any input X, the transcript we extract from Π' on X is *close in distribution* to the transcript of Π on X.

More formally, let π_X be the distribution of the transcript of Π on a specific input X, and let $\mathsf{dec}(\pi'_X)$ be the distribution of the extracted transcript obtained by running Π' on X and then applying the decoding function dec. Then our compression result can be stated (slightly informally) as follows:

▶ **Theorem 1** (Compression in the Shared Blackboard Model, Informal)**.** *Let $\rho > 0$ and $k \in \mathbb{N}$. Let Π be a randomized shared blackboard protocol between k players, and let μ be a joint distribution over the inputs for the players in Π. Then there exists a randomized protocol Π' in the* shared blackboard model *satisfying the following properties:*

1. *The worst case communication complexity of Π' is $\tilde{O}(k \cdot \mathrm{IC}_\mu(\Pi)/\operatorname{poly}(\rho))$.*
2. *There exists a deterministic function* dec *that given a transcript of Π', outputs a corresponding transcript of Π, such that for any global input X we have $\mathrm{SD}(\pi_X, \mathsf{dec}(\pi'_X)) \le \rho$.*

Here, $\mathrm{SD}(\mu, \eta) = \sup_{A \subseteq \Omega} |\mu(A) - \eta(A)|$ denotes the statistical distance between the distributions μ, η over the universe Ω, and $\mu(A), \eta(A)$ denote the probability of event A under μ and η respectively.

Our compression scheme is based on the beautiful two-player compression scheme of [2] for external information, but we face several non-trivial challenges in adapting the scheme to work with multi-player protocols.

1.1.2 Compression lower bound

Our compression scheme achieves communication $\tilde{O}(I \cdot k)$. For any compression scheme, the information cost $\mathrm{IC}(\Pi)$ is a lower bound on the communication of the compressed protocol, as any bit communicated by the protocol can give at most one bit of information about the inputs. However, it is natural to ask whether the blowup by a factor of k in the communication complexity of our above compression result is necessary, and we show that indeed it is: there is a communication protocol with information cost I, which cannot be compressed to a protocol that uses less than $\tilde{\Omega}(k \cdot I)$ bits of communication. This rules out the existence of a better compression scheme than the one suggested by Theorem 1, up to logarithmic factors.

To show this lower bound, we construct a function f on k inputs, and show that f can be computed by a protocol Π with information cost I. In contrast, we prove an $\tilde{\Omega}(k \cdot I)$ lower bound on the distributional communication complexity of f, that is, we show that no protocol with communication cost $\tilde{O}(k \cdot I)$ can compute f. This gives a *separation* between information and communication in the distributed multi-player setting. Observe that this also means that the protocol Π, which has information cost I, cannot be simulated by a protocol with communication complexity less than $\tilde{\Omega}(k \cdot I)$, because this would give us a low-communication protocol for solving f.

▶ **Theorem 2.** *Let $I, k \in \mathbb{N}$. There exists a function $f(X_1, \ldots, X_k)$ on k inputs, and a joint distribution μ over the inputs $X = (X_1, \ldots, X_k)$ for f, such that the following hold:*

1. *There exists a deterministic communication protocol Π with $\mathrm{IC}_\mu(\Pi) = I$ for which the output of $\Pi(X)$ is $f(X)$ for every $X \in \operatorname{supp}(\mu)$.*

2. *Any randomized (public coin) communication protocol* Π' *satisfying*

$$\Pr[\Pi'(X) \text{ outputs } f(X)] \geq 0.99,$$

must communicate $\tilde{\Omega}(k \cdot I)$ *bits on average. Here the error probability and the average communication are over inputs* X *drawn from* μ *and the randomness used by* Π'*,*

We sketch the proof in Section 4.

A gap of $\tilde{\Omega}(k)$ between information and communication in the shared blackboard model was first shown in [5], where it is shown that the AND function on k input bits has a protocol with information cost $O(\log k)$, but any protocol that computes AND with high probability communicates at least $\Omega(k)$ bits. However, this leaves open the possibility that the difference is *additive* in k: that is, it could conceivably be that every protocol Π can be compressed to a protocol with communication complexity $\tilde{O}(I + k)$. Indeed, consider the Disjointness problem, $\text{DISJ}_{n,k}$, where each player i gets a set $X_i \subseteq [n]$, and we want to determine if $\bigcap_i X_i = \emptyset$. We can view $\text{DISJ}_{m,k}$ as the OR of n instances of AND: $\text{DISJ}_{n,k}(X_1, \ldots, X_k) = \bigwedge_{j=0}^{n-1} \bigvee_{i=1}^{k} \neg X_j^i$. It is shown in [5] that $\text{DISJ}_{n,k}$ has information $\Theta(n \log k)$ and communication cost $\tilde{\Theta}(n \log k + k)$ in the shared blackboard model, so even though AND exhibits a gap of $k/\log k$ between communication and information, somehow "many instances of AND" no longer exhibit the same gap. Nevertheless, Theorem 2 above shows that compression to $\tilde{O}(I + k)$ is impossible in general, so $\text{DISJ}_{n,k}$ is the exception and not the rule.

1.2 Organization of the Paper

The remainder of the paper is organized as follows. In Section 2, we review the basic notions of communication complexity and information theory required to state and prove our compression results. In Section 3 we sketch the compression scheme; for lack of space, some technical details are omitted here, and will appear in the full version of the paper. Finally, in Section 4 we describe our compression lower bound.

2 Preliminaries

Notation. We use bold-face letters to denote random variables. For variables $\boldsymbol{A}_1, \ldots, \boldsymbol{A}_\ell$ with joint distribution μ, we let $\mu(\boldsymbol{A}_i)$ denote the marginal distribution of $\boldsymbol{A}_i$, and $\mu(\boldsymbol{A}_i \mid \boldsymbol{A}_j = a_j)$ denote the distribution of $\boldsymbol{A}_i$ conditioned on $\boldsymbol{A}_j = a_j$ (and similarly for more variables). For a string S, we let $|S|$ denote the length of S.

Communication complexity. For a protocol Π in the shared blackboard model, we define the *communication complexity of* Π, denoted $\text{CC}(\Pi)$, as the worst-case number of bits that are written on the board in any execution of Π. We say that Π *solves* a problem $P : \mathcal{X}^k \to \mathcal{Y}$ if for any input $X = (X_1, \ldots, X_k) \in \mathcal{X}^k$, the probability that Π's output on X is $P(X)$ is at least $2/3$. The *communication complexity* of a problem P, denoted $\text{CC}(P)$, is the minimum communication complexity of a protocol that solves P. We also study the *distributional communication complexity* of P, denoted $\text{CC}_\mu(P)$, where now the minimum is taken over protocols that only need to succeed with high probability over inputs drawn from the distribution μ.

Information theory. We require the following notions.

For a pair of random variables $\boldsymbol{X}, \boldsymbol{Y}$ with joint distribution μ, we denote by $\mathrm{I}_\mu(\boldsymbol{X};\boldsymbol{Y}) = \mathrm{H}(\boldsymbol{X}) - \mathrm{H}(\boldsymbol{X}|\boldsymbol{Y})$ the *mutual information* between $\boldsymbol{X}$ and $\boldsymbol{Y}$ (here H is the Shannon entropy). We omit the distribution μ when clear from the context.

For a pair of distributions p, q over the same domain, we denote by $\mathsf{D}(p \parallel q)$ the *KL divergence* between p and q, given by $\mathsf{D}(p \parallel q) = \mathrm{E}_{x\sim p}\left[\log\left(\frac{p(x)}{q(x)}\right)\right]$. Mutual information and KL divergence satisfy: $\mathrm{I}(\boldsymbol{X};\boldsymbol{Y}) = \mathrm{E}_{x\sim\mu(\boldsymbol{X})}\left[\mathsf{D}\left(\mu(\boldsymbol{Y}|\boldsymbol{X}=x) \parallel \mu(\boldsymbol{Y})\right)\right]$, where μ is the joint distribution of $\boldsymbol{X}, \boldsymbol{Y}$, $\mu(\boldsymbol{X})$ (resp. $\mu(\boldsymbol{Y})$) is the marginal distribution of $\boldsymbol{X}$ (resp. $\boldsymbol{Y}$), and $\mu(\boldsymbol{Y}|\boldsymbol{X}=x)$ is the distribution of $\boldsymbol{Y}$ conditioned on $\boldsymbol{X}=x$. Intuitively, the mutual information measures the differences between the distribution of $\boldsymbol{Y}$ when we know $\boldsymbol{X}$, and the prior distribution of $\boldsymbol{Y}$.

Information cost. We will use the following measure of the information revealed by an interactive protocol.

▶ **Definition 3.** The *information cost* of a (private coin) protocol Π over random inputs $\boldsymbol{X} = \boldsymbol{X}_1, \ldots, \boldsymbol{X}_k$ drawn according to a joint distribution μ, is defined as

$$\underset{\mu}{\mathrm{IC}}(\Pi) = \underset{\mu}{\mathrm{I}}(\boldsymbol{\Pi}; \boldsymbol{X}_1, \ldots, \boldsymbol{X}_k),$$

where $\boldsymbol{\Pi}$ is a random variable indicating the transcript of Π on inputs $\boldsymbol{X}_1, \ldots, \boldsymbol{X}_k$.

3 Compression for Multi-Party Protocols: a Proof Sketch

Suppose we are given a protocol Π, with communication $\mathrm{CC} = \mathrm{CC}(\Pi)$ and information cost $I = \mathrm{IC}_\mu(\Pi)$ on some input distribution μ. We want to construct another protocol, Π', which on a given input *generates a transcript of* Π, but with communication cost that depends only polylogarithmically on $\mathrm{CC}(\Pi)$, and mainly depends on the information cost I of Π and on the number of players, k. We follow the framework introduced in [2] for two-party protocols.

All compression schemes rely on the intuition that if $\mathrm{IC}_\mu(\Pi) = \mathrm{I}_\mu(\boldsymbol{\Pi}; \boldsymbol{X})$ is small, then someone who does not know the input $\boldsymbol{X}$ can sample "close to" the correct distribution $\boldsymbol{\Pi}$ even without knowing the input $\boldsymbol{X}$. For example, in the extreme case where the information cost is 0, transcript is *independent* of the inputs and we can sample it from its correct distribution without knowing the inputs. More generally, if the information cost is very small, $O(1)$ bits, then we can sample the transcript without looking at the inputs, and have the players look at their inputs and "correct the mistakes" afterwards without using a lot of communication (the manner in which we do this is described in Sections 3.3 and 3.4). When the protocol has high information cost, $\mathrm{IC}(\Pi) = \omega(1)$, we reduce to the case of constant information cost by chopping the transcript up into pieces that each reveals a constant amount of information: this way, we get $O(\mathrm{IC}(\Pi))$ "pieces" that each reveal $\Theta(1)$ information, and we can compress each "piece" separately.

It is not trivial to "chop up the transcript" into pieces with $\Theta(1)$ information cost, because we may not *know* a-priori how much information has been revealed at each point. For example, suppose that player 1 gets two bits of input, $a, b \in \{0,1\}$, which are uniform and independent. If $a = 0$, then player 1 sends a uniformly random bit, and in this case it reveals zero information about its input; if $a = 1$, player 1 sends the bit b, revealing one bit of information about its input. Only player 1 knows how much information it has revealed, and in general all the players could behave this way; therefore the players need to cooperate to cut up the transcript into pieces that each reveal $\Theta(1)$ information.

Next we describe more formally what we *mean* by "the information revealed" up to some point in the protocol, and how this notion relates to our ability to sample transcripts without looking at the input.

3.1 The divergence tree

We view the run of Π on an input $x = (x_1, \ldots, x_k)$ as a *binary tree* representing all possible transcripts of Π. Since Π is randomized, it induces a *distribution* π_x on the leafs of the tree; our goal is to sample a *leaf* of the tree (that is, a transcript of Π) from a distribution that is close to π_x, but using as little communication as possible. In the sequel we freely interchange *transcripts* with *nodes* of the transcript tree. For convenience we introduce the following short-hand notation: $\pi_{\leq r}, \pi_{<r}$ denote the distribution of the first r or the first $(r-1)$ bits of Π, respectively. To denote a distribution η conditioned on an event of the form $\boldsymbol{A} = a$, we write $\eta|a$, and for a specific value y, we write its probability under $\eta|a$ as $\eta(y|a)$. (When we condition on multiple values we write, e.g., $\eta|a, b$).

At each node v of the tree, there is some player $O(v)$ whose turn it is to speak when we reach node v. We call this player the *owner* of node v. The two children of node v correspond to the case where player $O(v)$ writes 0 and 1 on the board, respectively.

The owner $O(v)$ of v knows the correct distribution over children of v induced by π_x, because this distribution depends only on its input (the player determines which child we will go to by speaking). We denote by c_{vx} this "correct" distribution; formally, for each $b \in \{0,1\}$, if v is a node at depth r, then

$$c_{vx}(b) = \Pr\left[\boldsymbol{\Pi}_{\leq r+1} = v \cdot b \mid \boldsymbol{\Pi}_{\leq r} = v, \boldsymbol{X} = x\right].$$

(Note that actually this probability only depends on the input $x_{O(v)}$ of the player that owns v, because what a player decides to write on the board depends only on its input and what was written so far.)

The other players and the observer do not know c_{vx}, because they do not know x (only their own private input). But they know the *prior* distribution c_v on the children of node v, which is simply the probability over the protocol's randomness *and the input*, that player $O(v)$ will write 0 (resp. 1), given that we reached node v. Formally, for $b \in \{0,1\}$,

$$c_v(b) = \Pr\left[\boldsymbol{\Pi}_{\leq r+1} = v \cdot b \mid \boldsymbol{\Pi}_{\leq r} = v\right] = \sum_x \left(\Pr_\mu\left[\boldsymbol{X} = x \mid \boldsymbol{\Pi}_{\leq r} = v\right] \cdot c_{vx}(b)\right).$$

(Here again r is the depth of node v.)

Re-written in short-hand notation, we have

$$c_{vx}(b) = \pi_{\leq r+1}(vb|v,x), \qquad c_v(b) = \pi_{\leq r+1}(vb|v) = \sum_x \left(\mu(x|v) c_{vx}(b)\right).$$

3.2 Relating the information cost to the tree

Intuitively, if Π has low information cost, then the true distribution c_{vx} and the prior c_v should be "close" for most nodes v in the tree corresponding to input x, because the message each player decides to write on the board does not depend strongly on its input (otherwise it would reveal a lot of information about the input). And this is made formal by recalling that $\mathrm{IC}_\mu(\Pi) = \mathrm{I}_\mu(\boldsymbol{\Pi}; \boldsymbol{X})$, and using the chain rule and the relationship between mutual

information and divergence to see that

$$\begin{aligned}\underset{\mu}{\mathrm{IC}}(\Pi) &= \sum_{r=1}^{\mathrm{CC}(\Pi)} \underset{\mu}{\mathrm{I}}(\boldsymbol{\Pi}_r; \boldsymbol{X} \mid \boldsymbol{\Pi}_{<r}) = \sum_{r=1}^{\mathrm{CC}(\Pi)} \mathop{\mathrm{E}}_{v\sim\pi_{<r}} \left[\mathop{\mathrm{E}}_{x\sim\mu|v} \left[\mathsf{D}\left(\pi_r|v, x \parallel \pi_r|v\right)\right]\right] \\ &= \sum_{r=1}^{\mathrm{CC}(\Pi)} \mathop{\mathrm{E}}_{v\sim\pi_{<r}} \left[\mathop{\mathrm{E}}_{x\sim\mu|v} \left[\mathsf{D}\left(c_{vx} \parallel c_v\right)\right]\right]. \end{aligned} \tag{1}$$

So, what we "pay" at each round is the expected divergence between the true distribution c_{vx} and the prior c_v. Call this quantity the *divergence cost of* v on input x: $\mathrm{D}_x(v) = \mathsf{D}\left(c_{vx} \parallel c_v\right)$. We extend this to paths $p = v_0, \ldots, v_s$ in the natural way: the cost of the path is the sum of the costs of the nodes on the path, that is, $\mathrm{D}_x(p) = \sum_{i=0}^{s} \mathsf{D}\left(c_{v_i x} \parallel c_{v_i}\right)$. By re-arranging the order of the expectations in (1) we get that $\mathrm{IC}_\mu(\Pi) = \mathrm{E}_{x\sim\mu, t\sim\pi|x}\left[\mathrm{D}_x(t)\right]$, where here t is a complete transcript, viewed as path from the root of the tree to a leaf. Thus, protocols with low information cost have on average a low divergence cost on paths from the root to a leaf, so at most nodes on the path, the true distribution c_{vx} is close to the prior c_v in divergence. We rely on this characterization in our compression scheme.

A key technical ingredient in the compression algorithm is *rejection sampling*, which we review below.

3.3 Rejection sampling

Suppose we want to sample from some distribution p, but we only have access to samples generated from another distribution q. If we know an upper bound $M \geq \max_w p(w)/q(w)$ on the ratio between p and q, we can use *rejection sampling*, which works as follows:

1. Generate a candidate sample $w \sim q$.
2. *Accept* w with probability $p(w)/(M \cdot q(w))$, and otherwise *reject* w and try again.

It is not hard to show that the probability that for any w, the probability that w is generated by the procedure above is exactly $p(w)$: informally, we sample from q, but then "self-correct" by accepting the sample only with probability $p(w)/(M \cdot q(w))$, so the probability that the candidate is w and we accept it is $q(w) \cdot (p(w)/(Mq(w)) = p(w)/M$. Also, at each attempt, the probability that we accept the candidate is

$$\sum_w \left(q(w) \cdot \frac{p(w)}{Mq(w)} \right) = \frac{1}{M} \sum_w p(w) = \frac{1}{M},$$

so the distribution generated *given* that we accepted is exactly p. Moreover, the number of attempts required until we accept a candidate is $O(M)$ in expectation.

For our purposes we use rejection sampling as follows:

- The distribution q that we know how to sample from is the *prior* distribution π on leafs (or later on, internal nodes) of the tree, which all players know, because it does not depend on the input. (This is the distribution where at each node v we sample a child from the prior c_v.) We can sample from this distribution simply using the public randomness.
- The distribution p that we *want* to sample from is the true distribution π_x on leafs (obtained by taking c_{vx} at each node v), which is not known to any player.

We will show that the players can *approximate* the ratio $\pi_x(t)/\pi(t)$ for any leaf (or internal node) t. This allows us to first sample $t \sim \pi$ from the prior, and then reject it with *approximately* the right probability $\pi_x(t)/(M\pi(t))$, where M is an appropriately chosen

normalization factor. In this way we generate leafs from a distribution that is *close* to the true distribution $\pi_x(t)$.

(A major difference between our work and [2] is that in [2], where there are only two players, they do not need to explicitly compute the ratio $\pi_x(t)/\pi(t)$; they use a clever trick that rejects with the right probability. For a general number of players $k \geq 2$, we can no longer do this, and we must compute an approximation of the ratio and reject with that probability.)

To instantiate this outline, we must answer the following questions:

1. What is a good upper bound $M \geq \max_t \pi_x(t)/\pi(t)$? Actually, we will not be able to find a perfect bound, only a *high probability* bound which holds for most t.
2. How do we compute or approximate the ratio $\pi_x(t)/(M\pi(t))$?

3.4 Compressing protocols with constant information cost

To start with, suppose we have a protocol Π with $\mathrm{IC}_\mu(\Pi) = I = O(1)$, that is, a typical path from the root to a leaf of Π incurs only constant divergence cost. In this case we can compress Π to a protocol Π' with communication $2^{O(I)}$. This initially looks very bad, but since $I = O(1)$, in fact the communication cost of Π' is also $2^{O(1)} = O(1)$, regardless of the communication cost of the original protocol Π.

Recall that the divergence between two distributions p, q is defined as

$$\mathsf{D}\left(p \parallel q\right) = \mathop{\mathrm{E}}_{w\sim p}\left[\log \frac{p(w)}{q(w)}\right],$$

that is, $\mathsf{D}\left(p \parallel q\right)$ is the *expected log-ratio* between p and q when we sample from p. This means that if $\mathsf{D}\left(p \parallel q\right) = d$ and we sample $w \sim p$, then with good probability we will have $p(w)/q(w) \leq 2^{O(d)}$ [4]. Therefore we can use $2^{O(d)} = 2^{O(I)}$ as our (high-probability) upper bound M in the rejection sampling scheme, which leads to an expected communication cost of $2^{O(I)}$, times the cost of a single attempt of rejection sampling (that is, sampling a candidate using public randomness, and then deciding whether to accept it).

It remains to describe how we approximate the acceptance probability of a leaf t, which should be $\pi_x(t)/(M\pi(t))$. Here we use the following observation about k-party protocols, which generalizes the corresponding observation for two players from [2]: for each player i, let π_x^i be the distribution where at each node v we select a child with probability c_{vx} if player i owns node v, and with probability c_v otherwise. Each player i can compute $\pi_x^i(t)$ for any leaf t. And if we take the *product* of the $\pi_x^i(t)$'s, we get:

$$\prod_{i=1}^{k} \pi_x^i(t) = \prod_{r=0}^{\mathrm{CC}(\Pi)-1} \left(c_{t_r x}(t_{r+1}) \cdot c_{t_r}(t_{r+1})^{k-1}\right),$$

where t_r is the r-th node on the path from the root to t; this is because exactly one player, the owner of t_r, uses $c_{t_r x}$ to select a child at t_r, and the other players use c_{t_r}. Now let $f_i(t) = \pi_x^i(t)/\pi(t)$ for each $i = 1, \ldots, k$. Then the product of the f_i's is

$$\prod_{i=1}^{k} f_i(t) = \frac{\prod_{r=0}^{\mathrm{CC}(\Pi)-1} c_{t_r x}(t_{r+1}) \cdot c_{t_r}(t_{r+1})^{k-1}}{\prod_{r=0}^{\mathrm{CC}(\Pi)-1} c_{t_r}(t_{r+1})^{k}} = \prod_{r=0}^{\mathrm{CC}(\Pi)-1} \frac{c_{t_r x}(t_{r+1})}{c_{t_r}(t_{r+1})} = \frac{\pi_x(t)}{\pi(t)},$$

giving us exactly the ratio we want.

Thus, in order to implement the rejection sampling, we need to approximate the product $\prod_{i=1}^{k} f_i(t)$, and use it to get an approximate acceptance probability. We require *very* good precision: the approximation needs to be to within a factor $(1 \pm \epsilon)$, where $\epsilon = O(1/I)$ (for

now, I is constant, but later it will not be). This is not hard: we can estimate the product to within $(1 \pm \epsilon)$ by estimating $\sum_{i=1}^{k} \log f_i(t)$ to within $(1 \pm O(\epsilon))$, but first discarding from the sum terms that have very small absolute value, $|\log f_i(t)| < O(\epsilon/k)$. Because $2^x = 1 + O(x)$ for very small x, even if we ignore all the small terms, they only add up to $O(\epsilon)$ in absolute value, and we still get a $(1 \pm \epsilon)$-multiplicative approximation.

For the larger terms, $\log f_i$ such that $|\log f_i| \geq \epsilon/k$, we estimate each one of them to within $(1 \pm \epsilon)$ by dividing the range of possibilities for their absolute value, $[\epsilon/k, M]$, into intervals of exponentially-increasing width, where each interval is $(1 + \epsilon)$ the size of the preceding interval. We then have each player i tell us which interval their contribution $\log f_i(t)$ lies in, and its sign. Combining all the estimates of the individual contributions, we come up with a $(1 \pm \epsilon)$-approximation to the product $\prod_{i=1}^{k} f_i(t)$.

Using an approximation instead of the exact value of the acceptance probability means that the rejection sampling does not generate the correct distribution π_x. However, we prove that if we use a $(1 \pm \epsilon)$-approximation for the acceptance probability, then the distribution generated is $O(\epsilon)$-close to π_x in statistical distance, so that our compression scheme generates a distribution that is very close to the correct one. In the shared blackboard model, the cost of the approximation is $O(k \log 1/\epsilon)$ bits of communication.

3.5 Compression for protocols with large information cost

Following [2], compression for protocols with constant information cost can be extended to protocols with higher information cost as follows: we "chop up" each path in the protocol tree into segments with divergence cost $\Theta(1)$ each. This induces a set of *frontiers* in the tree, where each frontier intersects each path from the root at exactly one node. Instead of directly sampling a complete transcript, which corresponds to directly sampling leaf of the tree, we sample the transcript in segments: first we sample a node from the first frontier, then a node from the second frontier, and so on, until we reach a leaf. Because the divergence cost of each segment is constant, we can use rejection sampling as outlined above for constant divergence cost. And since each frontier "consumes" $\Omega(1)$ of the total divergence cost of the path, and we know that the total divergence cost is $O(I)$ with high probability, the total number of steps is $O(I)$.

Our overall compression scheme is as follows: we start from the root of the tree, and while we have not yet reached a leaf, we sample a node from the next frontier, using rejection sampling to sample very close to the correct distribution induced by π_x. The cost of each such step is $\tilde{O}(k)$, so the total cost of sampling a leaf of the tree is $\tilde{O}(I \cdot k)$.

Next we give a more precise definition of the frontiers, and show how we can sample a node from the next frontier.

3.5.1 Frontiers

The definition of a frontier is subtle (and differs from [2] significantly; having only two players makes life much easier). The main question is: how can we define the frontier in a way that is precise enough so that each path segment between two frontiers has divergence cost $\Theta(1)$, but on the other hand allows us the players to *find* the frontier using only $\tilde{O}(k)$ communication? In particular, we cannot afford to have the players send real numbers with high precision. So how can we identify the point where *together* the players' divergence cost has reached $\Theta(1)$?

Fix a parameter β, which is the target divergence cost we want each path segment between two frontiers to have. Denote by $\mathrm{D}_i(v,w)$ the *individual divergence cost* of player i, defined as the sum of the divergence costs on the path from v to u, but only at nodes owned by player i. Then by definition, $\mathrm{D}(v,w) = \sum_{i=1}^{k} \mathrm{D}_i(v,w)$. Moreover, each player can compute $\mathrm{D}_i(v,w)$, because it knows the divergence cost at nodes it owns: it knows both the true distribution and the prior, and can compute the divergence between them. Finally, let $\tilde{\mathrm{D}}_i(v,w) = \lfloor \mathrm{D}_i(v,w)/(\beta/k) \rfloor$ denote the player i's divergence cost $\mathrm{D}_i(v,w)$ divided by β/k and rounded down to the nearest integer.

Formally, we define the *frontier* of a node v in the tree to be the $\mathcal{F}_{vx}$ set of nodes w such that

1. For each strict prefix w' of the path from v to w we have $\sum_{i=1}^{k} (\beta/k)\tilde{\mathrm{D}}_i(v,w') < \beta$, but
2. For the full path we have $\sum_{i=1}^{k} (\beta/k)\tilde{\mathrm{D}}_i(v,w) \geq \beta$.

That is, the frontier is the first point on the path where the sum of the divergence costs of the players, each rounded down to the nearest β/k, first exceeds β.

Each player can compute its own contribution $\tilde{\mathrm{D}}_i(v,w)$ given v and w and announce a constant multiplicative approximation of it. This is good enough to ensure that for any frontier node $w \in \mathcal{F}_{vx}$ we have $\mathrm{D}(v,w) = \Theta(\beta)$.

(In [2], the frontier is defined as the point where *one* of the two players first reaches divergence cost $\mathrm{D}_i(v,w) \geq \beta$. Since there are only two players, this also means that the *total* divergence cost is $\Theta(\beta)$. This has the advantage that in order to check if a node is on the frontier, all we need to do is ask the two players whether they have individual divergence cost at most β or not, costing a single bit per player. In our case, if we tried to take this approach, the total divergence cost could be as high as $\Theta(k\beta)$, which we cannot afford, as our probability of accepting in the rejection sampling would then be only $2^{-\Theta(k\beta)}$. Thus, we must define the frontier by the *total* divergence cost directly, and this leads to technical complications.)

We define two distributions on the frontier $\mathcal{F}_{vx}$: the first, F_{vx}, is the "correct" distribution induced by π_x, where we sample a path from v to a leaf, $t \sim \pi_x|v$, and cut the path at the (unique) point where it intersects $\mathcal{F}_{vx}$. The second distribution, denoted F_v, is the *prior* distribution, induced by the prior π in the same way.

3.5.2 Sampling from the frontier

Suppose we are currently at node v in the tree, and we want to sample from the correct distribution F_{vx} on the frontier at node v. Let us recall the ingredients we need to sample from the frontier using rejection sampling:

1. We need to sample a frontier node $w \sim F_v$,
2. We need to compute or approximate the acceptance probability, $F_{vx}(w)/(MF_v(w))$.

We already saw how to approximate the acceptance probability in Section 3.4. Here it becomes important to set the precision parameter ϵ to $O(1/I)$, because we will be passing through $O(I)$ frontiers, and the error adds up; we can only afford an error of $O(1/I)$ per frontier to get constant error in the end.

It remains to describe how we can sample a frontier node from the prior, $w \sim F_v$. To do this, we first sample a leaf $t \sim \pi|v$, using public randomness (since $\pi|v$ is known to all players). Next, we find the point where the frontier $\mathcal{F}_{vx}$ intersects the path from v to t, and return this node. In order to find the cut-point of the frontier we use binary search (as in [2]), to find the first node w on the path from v to t where we have $\sum_{i=1}^{k} (\beta/k)\tilde{\mathrm{D}}_i(v,w) \geq \beta$.

The total cost of sampling from $w \sim F_v$ is bounded by $\log \mathrm{CC}$ times the cost of approximating $\sum_{i=1}^{k} (\beta/k)\tilde{\mathrm{D}}_i(v,w) \geq \beta$ to within a constant multiplicative factor ϵ, which is $O(k \log(k/\epsilon))$ bits.

4 Compression Lower Bound

In this section we present the $\texttt{AndTree}_h$ function, parameterized by h, with low information cost, $O(h \log k)$, but high communication cost, $\tilde{\Omega}(k \cdot h)$. This function serves to *separate* information from communication in the shared blackboard model, ruling out the existence of a compression scheme whose cost does not depend nearly-linearly on k.

4.1 The AndTree problem

The input to $\texttt{AndTree}_h$ is represented by a complete binary tree of depth h; at each node u of the tree, we embed an instance of AND, with each player i receiving a bit $\boldsymbol{X}_u^i$. We represent each node of the tree by the path from the root, with the root denoted λ, a left child appending zero to its parent, and a right child appending one to its parent.

For a tree T, let $\mathsf{leaf}(T)$ be the leaf obtained by starting at the root, turning left at each node u such that $\bigwedge_i \boldsymbol{X}_u^i = 0$, and turning right at each node u such that $\bigwedge_i \boldsymbol{X}_u^i = 1$. We define the output of the $\texttt{AndTree}_h$ problem to be the *parity* of this leaf: $\texttt{AndTree}_h(T) = \mathrm{parity}(\mathsf{leaf}(T))$.

4.2 The information cost of AndTree

A single instance of AND can be solved with $O(\log k)$ bits of information under any input distribution: simply have the players announce their inputs one-by-one, until we either find a player with input 0, in which case we halt and output 0, or all players have announced that their input is 1, in which case we output 1. This protocol has only $k+1$ possible transcripts: if all players got 1, the transcript is 1^k, and otherwise the transcript is $1^i 0$, where i is the index of the first player that got 0. Therefore the *entropy* of the transcript is $O(\log k)$, meaning that the information cost of the protocol is also $O(\log k)$.

To solve a full instance of $\texttt{AndTree}$, we start at the root, and use the protocol above to solve each node, moving down to the correct child. When we arrive at a leaf, we output its parity. The information cost is $O(h \log k)$ under *any* input distribution.

4.3 The communication complexity of AndTree

Next we must show that no randomized protocol with $o(k \cdot I)$ communication can solve $\texttt{AndTree}$ with small error in the worst-case. We show a slightly stronger lower bound: we give an input distribution μ, and show that no randomized protocol with $o(k \cdot I)$ communication can solve $\texttt{AndTree}$ with small *average* error when inputs are drawn from μ.

For the analysis, we divide the tree into R layers, each of depth $d = \Theta(\log k + \log h)$. We define the input distribution μ_R on an R-layer tree as follows: for each node u of the tree, we draw a random player $\boldsymbol{P}(u)$ uniformly and independently. The input of player $\boldsymbol{P}(u)$ is drawn uniformly, $\boldsymbol{B}_u \sim \mathsf{U}(\{0,1\})$; the other players receive 1. Thus, under μ_R, for each node u we have $\bigwedge_i \boldsymbol{X}_u^i = \boldsymbol{X}_u^{\boldsymbol{P}(u)} = \boldsymbol{B}_u$.

It is convenient to represent an r-layer tree as $T = (S, R_1, \ldots, R_{2^d})$, where S is the top layer (i.e., the first d levels of the tree), and $R_1, \ldots, R_{2^d}$ are the subtrees starting at the leafs of S.

▶ **Theorem 4.** *For sufficiently small constant ϵ, the randomized public-coin communication complexity of $\texttt{AndTree}_h$ is $\mathrm{CC}(\texttt{AndTree}_h, \epsilon) = \Omega(R \cdot k)$, where $R = \Theta(h/(\log k + \log h))$.*

We prove the theorem by induction on the number of layers: for each $r \le R$, given an protocol with communication C and error ϵ for the r-layer tree, we "peel off" a layer, and

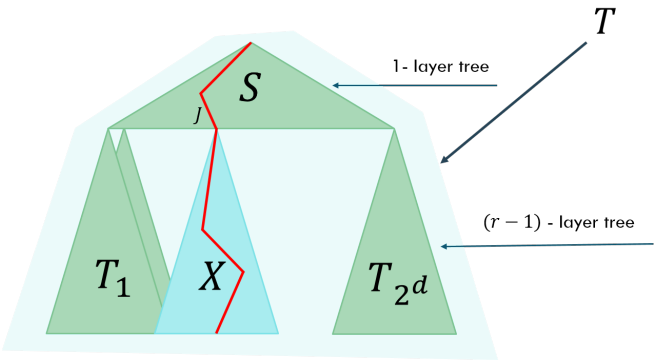

Figure 1 Embedding X in an r-layer tree

construct a protocol with communication $C - \Theta(k)$ and error $\epsilon + \Theta(1/R)$ for the $(r-1)$-layer tree. If we started out with communication less than $c \cdot R \cdot k$ for some sufficiently small constant c, then eventually we obtain a protocol with zero communication and constant error for the one-layer tree, which is impossible.

The induction step. Let $d = \beta(\log k + \log h)$ be the height of a layer, where β is a constant whose value will be fixed later.

Given a protocol Π for the r-layer tree $\texttt{AndTree}_{r \cdot d}$, with communication cost $\mathrm{CC}(\Pi) = C$ and distributional error ϵ on μ_r, we construct a new protocol, $\tilde{\Pi}$, for $(r-1)$-layer trees.

Embedding in an r-layer tree

The input to $\tilde{\Pi}$ is an $(r-1)$-layer tree $\boldsymbol{X}$, but the protocol Π that we are given takes as input r-*layer* trees. How can we use Π to solve our input, which has one fewer layer? The answer is that we *embed* our $(r-1)$-layer input in a larger r-layer tree $\boldsymbol{T}$, the rest of which we generate using public randomness. We design the embedding so that the answer on the input $\boldsymbol{X}$ can be extracted from the answer on $\boldsymbol{T}$.

More formally, we embed $\boldsymbol{X}$ in an r-layer tree by publicly generating a random "top layer", $\boldsymbol{S} \sim \mu_1$, and placing $\boldsymbol{X}$ as the $\boldsymbol{J}$-th subtree of $\boldsymbol{S}$, where $\boldsymbol{J}$ is the "correct" leaf of $\boldsymbol{S}$, obtained by solving the AND at each node and turning left when the answer is 0 and right when the answer is 1. Note that by definition, $\mathrm{parity}(\boldsymbol{J}) = \texttt{AndTree}(\boldsymbol{S})$.

The other $2^d - 1$ subtrees, placed under the leafs of $\boldsymbol{S}$ that are not the "correct" leaf $\boldsymbol{J}$, are generated publicly and independently from μ_{r-1}. Let $\boldsymbol{T}_1, \ldots, \boldsymbol{T}_{2^d}$ denote these subtrees. Then formally, what we said above is that we set $\boldsymbol{T}_{\boldsymbol{J}} = \boldsymbol{X}$, and we sample $(\boldsymbol{T}_1, \ldots, \boldsymbol{T}_{\boldsymbol{J}-1}, \boldsymbol{T}_{\boldsymbol{J}+1}, \ldots, \boldsymbol{T}_{2^d}) \sim (\mu_{r-1})^{2^d-1}$.

Let $\boldsymbol{T} = (\boldsymbol{S}, \boldsymbol{T}_1, \ldots, \boldsymbol{T}_{2^d})$ be the resulting r-layer tree.

▶ **Property 5.** *Observe that since* $\texttt{AndTree}_d(\boldsymbol{S}) = \mathrm{parity}(\boldsymbol{J})$*, and we set* $\boldsymbol{T}_{\boldsymbol{J}} = \boldsymbol{X}$*, we have* $\texttt{AndTree}_{(r-1)d}(\boldsymbol{X}) = \texttt{AndTree}_{(r-1)d}(\boldsymbol{T}_{\boldsymbol{J}}) = \texttt{AndTree}_{rd}(\boldsymbol{T}) \oplus \mathrm{parity}(\boldsymbol{J})$.

We can now solve our $(r-1)$-layer input $\boldsymbol{X}$ by calling Π on the r-layer tree $\boldsymbol{T}$ that we constructed, but this is not enough: for the induction step we need to construct a protocol with *less communication* than Π. Our goal is therefore to simulate the execution of Π on $\boldsymbol{T}$, but using less communication than Π requires. If we can do this, then we can solve $\texttt{AndTree}_{(r-1)d}$ using less communication.

Saving $\Theta(k)$ bits of communication

We split the transcript of the original protocol Π into two parts, $\boldsymbol{\Pi} = \boldsymbol{\Pi}_1\boldsymbol{\Pi}_2$, where the prefix $\boldsymbol{\Pi}_1$ is of length $|\boldsymbol{\Pi}_1| = \alpha k$ for a constant α whose value will be fixed later, and the suffix $\boldsymbol{\Pi}_2$ consists of the rest of the transcript.

In $\tilde{\Pi}$, instead of sampling $\boldsymbol{\Pi}_1$ "correctly" by having each player look at their input and send the messages indicated under Π, we *fix* a prefix m_1 of length αk, and have the players sample $\boldsymbol{\Pi}_2$ as though they had said m_1; that is, the transcript of $\tilde{\Pi}$ is the suffix $\Pi \sim \pi_2|\boldsymbol{\Pi}_1 = m_1, \boldsymbol{J} = j$. We need to show that there is a "good" choice for m_1 and j, under which the suffix has small error probability on our real input $\boldsymbol{X}$.

▶ **Definition 6.** We say that a pair (m_1, j) is *good*, where $m_1 \in \{0,1\}^{\alpha k}, j \in [2^d]$, if the following holds:

1. $\mathsf{D}\left(\mu_{r-1}(\boldsymbol{T}_j|\boldsymbol{\Pi}_1 = m_1, \boldsymbol{J} = j) \parallel \mu_{r-1}(\boldsymbol{T}_j)\right) \leq \frac{1}{100R^2}$, and,
2. $\Pr\left[\Pi \text{ errs} \mid \boldsymbol{\Pi}_1 = m_1, \boldsymbol{J} = j\right] \leq \left(1 + \frac{1}{R}\right) \cdot \epsilon$.

Note that since $\boldsymbol{T}_j$ is independent of $\boldsymbol{J}$, the first condition can also be written as:

$$\mathsf{D}\left(\mu_{r-1}(\boldsymbol{T}_j|\boldsymbol{\Pi}_1 = m_1, \boldsymbol{J} = j) \parallel \mu_{r-1}(\boldsymbol{T}_j|\boldsymbol{J} = j)\right) \leq \frac{1}{100R^2}.$$

▶ **Lemma 7.** *There exists a good pair (m_1, j).*

Before proving that there exists a good setting for (m_1, j), let us show that if (m_1, j) *is* good, then when we sample $\boldsymbol{\Pi}_2 \sim \pi_2|\boldsymbol{\Pi}_1 = m_1, \boldsymbol{J} = j$ we have small error on our $(r-1)$-layer input, $\boldsymbol{T}_j = \boldsymbol{X}$.

▶ **Lemma 8.** *If (m_1, j) is good, then* $\Pr_{\boldsymbol{X}\sim\mu_{r-1}}\left[\boldsymbol{\Pi} \text{ errs on } \boldsymbol{X} \mid \boldsymbol{J} = j, \boldsymbol{\Pi}_1 = m_1\right] \leq \left(1 + \frac{1}{R}\right) \cdot \epsilon + \frac{1}{10R}$.

Proof sketch. Let μ' be the distribution $\mu_{r-1}(\boldsymbol{T}_j|\boldsymbol{\Pi}_1 = m_1, \boldsymbol{J} = j)$ of the j-th subtree, given $\boldsymbol{\Pi}_1 = m_1$ and $\boldsymbol{J} = j$. Let $\boldsymbol{E} = \boldsymbol{E}(\boldsymbol{X})$ be an indicator for the event that the output produced by the suffix $\boldsymbol{\Pi}_2$ is incorrect on $\boldsymbol{X}$. Finally, let π_2 be the distribution of $\boldsymbol{\Pi}_2$ given $\boldsymbol{\Pi}_1 = m_1, \boldsymbol{J} = j$ when the input is $\boldsymbol{X} \sim \mu_{r-1}$, and let π_2' be the distribution of $\boldsymbol{\Pi}_2$ given $\boldsymbol{\Pi}_1 = m_1, \boldsymbol{J} = j$ when the input is $\boldsymbol{X} \sim \mu'$.

Re-stated in this notation, the lemma asserts that $\Pr_{\pi_2}[\boldsymbol{E}] \leq \left(1 + \frac{1}{R}\right) \cdot \epsilon + \frac{1}{10 \cdot R}$, and Condition 2 of Definition 6 says that $\Pr_{\pi_2'}[\boldsymbol{E}] \leq \left(1 + \frac{1}{R}\right) \cdot \epsilon$. Thus, to show the lemma, we show that $\pi_2(\boldsymbol{E})$ and $\pi_2'(\boldsymbol{E})$ are "close", and therefore the expectation of $\boldsymbol{E}$ cannot differ by much between them. We get the required "closeness" from the first condition of 6, which bounds the divergence between the two distributions. ◀

▶ **Corollary 9.** *Let $\tilde{\boldsymbol{\Pi}}$ be the protocol defined by taking a good pair (m_1, j), embedding the input in location $\boldsymbol{T}_j$, sampling the suffix $\boldsymbol{\Pi}_2$ from its distribution given $\boldsymbol{\Pi}_1 = m_1, \boldsymbol{J} = j$ (the missing parts of the input are sampled from public randomness according to m_1 and j) and returning ($\boldsymbol{\Pi}_2'$s output) $\oplus$ parity$(\boldsymbol{S})$. Then* $\Pr_{\boldsymbol{X}\sim\mu_{r-1}}\left[\tilde{\boldsymbol{\Pi}} \text{ errs on } \boldsymbol{X}\right] \leq \left(1 + \frac{1}{R}\right) \cdot \epsilon + \frac{1}{10R}$.

Now let us sketch the proof that with high probability there is a good pair (m_1, j).

Proof sketch of Lemma 7. We show that the probability over $\boldsymbol{M}_1$ and $\boldsymbol{J}$ that *either* the first or the second condition of Definition 6 fails to hold is smaller than 1, which means that there is a pair (m_1, j) satisfying both conditions.

First consider the first condition, which requires that m_1 does not reveal a lot of information about the subtree $\boldsymbol{T}_j$. Since m_1 is short, only αk bits, the total information it reveals about *all* the sub-trees $\boldsymbol{T}_1, \ldots, \boldsymbol{T}_{2^d}$ together is at most $O(\alpha k)$ with high probability. The sub-trees $\boldsymbol{T}_1, \ldots, \boldsymbol{T}_{2^d}$ are initially independent, and information has the *super-additivity property*:

if $\boldsymbol{A}_1, \ldots, \boldsymbol{A}_m$ are independent, then for any $\boldsymbol{B}$, $\mathrm{I}(\boldsymbol{B}; \boldsymbol{A}_1, \ldots, \boldsymbol{A}_m) \geq \sum_{i=1}^{m} \mathrm{I}(\boldsymbol{B}; \boldsymbol{A}_i)$. In our case this means that if we choose a *uniformly random* index $\boldsymbol{J} \in \{1, \ldots, 2^d\}$ which is independent of m_1, the information m_1 reveals about $\boldsymbol{T}_{\boldsymbol{J}}$ is at most $O(\alpha k)/2^d = O(\alpha k)/\operatorname{poly}(k)$, which is negligible.

Unfortunately, the choice of the index $\boldsymbol{J}$ where we embed our input is *not* independent of m_1: there is a hypothetical possibility that the players can discover where the input is embedded by computing the correct leaf of the top layer $\boldsymbol{S}$ (which is how we defined $\boldsymbol{J}$). If they can do this, then they can focus all their attention on the correct sub-tree $\boldsymbol{T}_{\boldsymbol{J}}$, and m_1 can reveal a lot of information about it. We need to show that since m_1 is short, the players *cannot* discover $\boldsymbol{J}$.

To capture the fact that the players cannot learn $\boldsymbol{J}$, except with small probability, we use the notion of *min-entropy*: for a random variable $\boldsymbol{A} \sim \eta$ over domain Ω, the min-entropy of $\boldsymbol{A}$ is defined as $\mathrm{H}_\infty(\boldsymbol{A}) = \min_{\omega \in \Omega} \log \frac{1}{\eta(\omega)}$. The min-entropy corresponds to our ability to *guess* the correct value of $\boldsymbol{A}$ (unlike Shannon entropy).

We use a lemma from [13] which generalizes the super-additivity of information to variables with high min-entropy, and asserts that

$$\begin{aligned} &\mathop{\mathrm{E}}_{\boldsymbol{J}|\boldsymbol{\Pi}_1 = m_1} \left[\mathsf{D}\left(\mu_{r-1}(\boldsymbol{T}_{\boldsymbol{J}}|\boldsymbol{\Pi}_1 = m_1, \boldsymbol{J}) \parallel \mu_{r-1}(\boldsymbol{T}_{\boldsymbol{J}}|\boldsymbol{J})\right)\right] \\ &\leq 2^{-\mathrm{H}_\infty(\boldsymbol{J}|m_1)} \cdot \mathsf{D}\left(\mu_{r-1}^{2^d}(\boldsymbol{T}_1, \ldots, \boldsymbol{T}_{2^d}|\boldsymbol{\Pi}_1 = m_1) \parallel \mu_{r-1}^{2^d}(\boldsymbol{T}_1, \ldots, \boldsymbol{T}_{2^d})\right). \end{aligned} \tag{2}$$

To use this lemma, we must show that with high probability the min-entropy $\mathrm{H}_\infty(\boldsymbol{J}|m_1)$ of the correct leaf given the message m_1 is high. This holds because m_1 consists of only αk bits of communication, which allows only an α-fraction of players to speak; for most nodes u in the top layer $\boldsymbol{S}$, the "influential" player $\boldsymbol{P}(u)$, whose input determines the correct child of u, does not get to say anything at all. Therefore the correct child at most nodes remains uniformly random even after seeing the message m_1, and the correct leaf $\boldsymbol{J}$ retains high min-entropy.

For the second condition of a good pair, we know that the overall error of Π is ϵ, and therefore, by Markov and the law of total expectation,

$$\Pr_{\boldsymbol{\Pi}_1, \mathcal{J}}\left[\Pr\left[\Pi \text{ errs} \mid \boldsymbol{J}, \boldsymbol{\Pi}_1\right] > \left(1 + \frac{1}{R}\right)\epsilon\right] < \frac{\epsilon}{\epsilon(1 + 1/R)} < 1 - \frac{1}{2R}.$$

A union bound over the probabilities that the first and second condition fail to hold yields the lemma. ◀

Proof of Theorem 4. Suppose Π is a protocol for $\texttt{AndTree}_h$ with error $1/100$ and communication C. Let $R = \lfloor h/d \rfloor - 1$. Using Corollary 9, we construct a series of protocols $\Pi_0, \ldots, \Pi_R$, where $\Pi_R = \Pi$ is the protocol we started with, and for each $r > 0$, the protocol Π_r solves $\texttt{AndTree}_{h-(R-r)d}$ with error at most ϵ_r, and its communication cost is $C - \alpha r k$. Applying Corollary 9 R times, we get that the final protocol Π_0, solves $\texttt{AndTree}_{h-Rd}$ with error

$$\epsilon_0 < \left(1 + \frac{1}{R}\right)^R \left(\epsilon_R + R \cdot \frac{1}{10R}\right) < e \cdot \left(\frac{1}{100} + \frac{1}{10}\right) < 1/3.$$

Since $h - Rd > 0$ by definition of R, the problem $\texttt{AndTree}_{h-Rd}$ cannot be solved with error $1/3$ with no communication (indeed, we know that the communication required is $\Omega(k)$ to solve even a single instance of AND), and therefore we must have $C > \alpha R k = \Omega\left(\frac{h}{\log k + \log h} k\right)$. ◀

We have now shown that for any sufficiently large h, the $\texttt{AndTree}_h$ problem can be solved using $O(h \log k)$ bits of information, but requires $\Omega(hk/(\log k + \log h))$ bits of communication. This shows that our compression scheme is optimal up to polylogarithmic factors in k and the input size.

References

1. Ziv Bar-Yossef, T. S. Jayram, Ravi Kumar, and D. Sivakumar. An information statistics approach to data stream and communication complexity. *J. Comput. Syst. Sci.*, 68(4):702–732, 2004.
2. Boaz Barak, Mark Braverman, Xi Chen, and Anup Rao. How to compress interactive communication. In *STOC*, pages 67–76, 2010.
3. Balthazar Bauer, Shay Moran, and Amir Yehudayoff. Internal compression of protocols to entropy. In *APPROX/RANDOM*, pages 481–496, 2015.
4. Mark Braverman. Interactive information complexity. In *STOC*, pages 505–524, 2012.
5. Mark Braverman and Rotem Oshman. On information complexity in the broadcast model. In *PODC*, pages 355–364, 2015.
6. Mark Braverman and Anup Rao. Information equals amortized communication. In *FOCS*, pages 748–757, 2011.
7. Joshua Brody, Harry Buhrman, Michal Koucký, Bruno Loff, Florian Speelman, and Nikolay K. Vereshchagin. Towards a reverse newman's theorem in interactive information complexity. In *CCC*, pages 24–33, 2013.
8. Amit Chakrabarti, Yaoyun Shi, Anthony Wirth, and Andrew Chi-Chih Yao. Informational complexity and the direct sum problem for simultaneous message complexity. In *FOCS*, pages 270–278, 2001.
9. Robert M Fano. *The transmission of information*. Massachusetts Institute of Technology, Research Laboratory of Electronics, 1949.
10. Prahladh Harsha, Rahul Jain, David A. McAllester, and Jaikumar Radhakrishnan. The communication complexity of correlation. *IEEE Transactions on Information Theory*, 56(1):438–449, 2010.
11. David A Huffman. A method for the construction of minimum redundancy codes. *proc. IRE*, 40(9):1098–1101, 1952.
12. Gillat Kol. Interactive compression for product distributions. In *STOC*, pages 987–998, 2016.
13. Gillat Kol and Ran Raz. Interactive channel capacity. In *STOC*, pages 715–724, 2013.
14. Sivaramakrishnan Natarajan Ramamoorthy and Anup Rao. How to compress asymmetric communication. In *CCC*, pages 102–123, 2015.
15. C. E. Shannon. A mathematical theory of communication. *The Bell Systems Technical Journal*, 27:July 379–423, October 623–656, 1948.
16. Alexander A. Sherstov. Compressing interactive communication under product distributions. In *FOCS*, pages 535–544, 2016.