Public vs. Private Randomness in Simultaneous Multi-Party Communication Complexity

Orr Fischer[*], Rotem Oshman[†] and Uri Zwick[‡]

July 4, 2016

Abstract

In simultaneous number-in-hand multi-party communication protocols, we have k players, who each receive a private input, and wish to compute a joint function of their inputs. The players simultaneously each send a single message to a referee, who then outputs the value of the function. The cost of the protocol is the total number of bits sent to the referee.

For two players, it is known that giving the players a public (shared) random string is much more useful than private randomness: public-coin protocols can be unboundedly better than deterministic protocols, while private-coin protocols can only give a quadratic improvement on deterministic protocols.

We extend the two-player gap to multiple players, and show that the private-coin communication complexity of a k-player function f is $\Omega(\sqrt{D(f)})$ for any $k \geq 2$. Perhaps surprisingly, this bound is tight: although one might expect the gap between private-coin and deterministic protocols to grow with the number of players, we show that the All-Equality function, where each player receives n bits of input and the players must determine if their inputs are all the same, can be solved by a private-coin protocol with $\tilde{O}(\sqrt{nk} + k)$ bits. Since All-Equality has deterministic complexity $\Theta(nk)$, this shows that sometimes the gap scales only as the square root of the number of players, and consequently the number of bits each player needs to send actually decreases as the number of players increases. We also consider the Exists-Equality function, where we ask whether there is a pair of players that received the same input, and prove a nearly-tight bound of $\tilde{\Theta}(k\sqrt{n})$ for it.

[*]Computer Science Department, Tel-Aviv University. Email: orrfischer@mail.tau.ac.il
[†]Computer Science Department, Tel-Aviv University. Email: roshman@mail.tau.ac.il
[‡]Computer Science Department, Tel-Aviv University. Email: zwick@tau.ac.il

1 Introduction

In his seminal '79 paper introducing the notion of two-party communication complexity [18], Yao also briefly considered communication between more than two players, and pointed out "one situation that deserves special attention": two players receive private inputs, and send randomized messages to a third player, who then produces the output. Yao asked what is the communication complexity of the Equality function (called "the identification function" in [18]) in this model: in the Equality function EQ_n, the two players receive vectors $\{0,1\}^n$, and the goal is to determine whether $x = y$.

Yao showed in [18] that EQ_n requires $\Omega(n)$ bits for determistic communication protocols, even if the players can communicate back-and-forth. Using a *shared* random string, the complexity reduces to $O(1)$, and using *private* randomness, but more than a single round, the complexity is $\Theta(\log n)$. In modern nomenclature, the model described above is called *the 2-player simultaneous model*, and the third player (who announces the output) is called the *referee*. Yao's question is then: what is the communication complexity of EQ_n using private randomness in the simultaneous model of communication complexity?

Some seventeen years later, Yao's question was answered: Newman and Sezegy showed in [16] that EQ_n requires $\Theta(\sqrt{n})$ bits to compute in the model above, if the players are allowed only private randomness. (Using shared randomness the complexity reduces to $O(1)$, even for simultaneous protocols.) Moreover, Babai and Kimmel showed in [3] that for *any* function f, if the deterministic simultaneous complexity of f is $D(f)$, then the private-coin simultaneous communication complexity of f is $\Omega(\sqrt{D(f)})$, so in this sense private randomness is of only limited use for simultaneous protocols.

In this paper we study multi-player simultaneous communication complexity[*], and ask: how useful are private random coins for more than two players? Intuitively, one might expect that as the number of players grows, the utility of private randomness should decrease. We first extend the $\Omega(\sqrt{D(f)})$ lower bound of [3] to the multi-player setting, and show that for any k-player function f, the private-coin simultaneous communication complexity of f is $\Omega(\sqrt{D(f)})$. We then show, perhaps contrary to expectation, that the extended lower bound is still tight in some cases.

To see why this may be surprising, consider the function $\text{ALLEQ}_{k,n}$, which generalizes EQ_n to k players: each player i receives a vector $x_i \in \{0,1\}^n$, and the goal is to determine whether all players received the same input. It is easy to see that the deterministic communication complexity of $\text{ALLEQ}_{k,n}$ is $\Omega(nk)$ (not just for simultanoues protocols), and each player must send n bits to the referee in the worst case. From the lower bound above, we obtain a lower bound of $\Omega(\sqrt{nk})$ for the private-coin simultaneous complexity of $\text{ALLEQ}_{k,n}$. It is easy to see that $\Omega(k)$ is also a lower bound, as each player must send at least one bit, so together we have a lower bound of $\Omega(\sqrt{nk} + k)$. If this lower bound is tight, then the average player only needs to send $O(\sqrt{n/k} + 1)$ bits to the referee in the worst-case, so in some sense we even *gain* from having more players, and indeed, if $k = \Omega(n)$, then the per-player cost of $\text{ALLEQ}_{k,n}$ with private coins is constant, just as it would be with shared coins.

Nevertheless, our lower bound *is* nearly tight, and we are able to give a simultaneous private-coin protocol for $\text{ALLEQ}_{k,n}$ where each players sends only $O(\sqrt{n/k} + \log(k))$ bits to the referee, for a total of $O(\sqrt{nk} + k \log \min\{k, n\})$ bits. This matches the lower bound of $\Omega(\sqrt{nk})$ when $k = O(n/\log^2 n)$. We also show that $\text{ALLEQ}_{k,n}$ requires $\Omega(k \log n)$ bits, so in fact our upper bound for ALLEQ is tight.

We then turn our attention to a harder class of k-player problems: those obtained by taking a 2-player function f and asking "do there exist two players on whose inputs f returns 1?". An example for this class is the function $\text{EXISTSEQ}_{k,n}$, which asks whether there exist two players that received the same input. We show that $\text{EXISTSEQ}_{k,n}$ requires $\tilde{\Theta}(k\sqrt{n})$ bits for private-coin simultaneous protocols, and moreover, any function in the class above has private-coin simultaneous complexity $\tilde{O}(kR(f))$, where $R(f)$ is the private-coin simultaneous complexity of f (with constant error).

[*]We consider the *number-in-hand* model, where each player receives a private input, rather than the perhaps more familiar *number-on-forehead* model, where each player can see the input of all the other players but not its own.

1.1 Related Work

As we mention above, two-player simultaneous communication complexity was first considered by Yao in [18], and has received considerable attention since. The Equality problem was studied in [3, 16, 7], and another optimal simultaneous protocol is given in [2], using error-correcting codes. In [12], a connection is established between simultaneous and one-round communication complexity and the VC-dimension. [8, 11] consider the question of simultaneously solving multiple copies of Equality and other functions, and in particular, [8] shows that solving m copies of Equality requires $\Omega(m\sqrt{n})$ bits for private-coin simultaneous 2-player protocols.

Multi-player communication complexity has also been extensively studied, but typically in the *number-on-forehead* model, where each player can see the inputs of all the other players but not its own. This model was introduced in [9]; sufficiently strong lower bounds on protocols in this model, even under restricted (but not simultaneous) communication patterns, would lead to new circuit lower bounds. Simultaneous communication complexity for number-on-forehead is considered in [4].

In contrast, in this paper we consider the *number-in-hand* model, where each player knows only its own input. This model is related to distributed computing and streaming (see, e.g., [17], which gives a lower bound for a promise version of Set Disjointness in our model).

An interesting "middle case" between the number-in-hand and number-on-forehead models is considered in [1, 5, 6]: there the input to the players is an undirected graph, and each player represents a node in the graph and receives the edges adjacent to this node as its input. This means that each edge is known to *two* players. This gives the players surprising power; for example, in [1] it is shown that graph connectivity can be decided in a total of $O(n \log^3 n)$ bits using public randomness. The power of private randomness in this model remains a fascinating open question and is part of the motivation for our work.

The functions ALLEQ and EXISTSEQ considered in this paper were also studed in, e.g., [10], but not in the context of simultaneous communication; the goal there is to quantify the communication cost of the network topology on communication complexity, in a setting where not all players can talk directly with each other.

2 Preliminaries

Notation. For a vector x of length n, we let x_{-i} denote the vector of length $n-1$ obtained by dropping the i-th coordinate of x (where $i \in [n]$).

Simultaneous protocols. Fix input domains $\mathcal{X}_1, \ldots, \mathcal{X}_k$ of sizes $m_1, \ldots, m_k$ (respectively). A *private-coin k-player simultaneous communication protocol* Π on $\mathcal{X}_1 \times \ldots \times \mathcal{X}_k$ is a tuple of functions $(\pi_1, \ldots, \pi_k, O)$, where each π_i maps the inputs $\mathcal{X}_i$ of player i to a distribution on a finite set of messages $\mathcal{M}_i \subseteq \{0,1\}^*$, and O is the referee's output function, mapping each tuple of messages in $\mathcal{M}_1 \times \ldots \times \mathcal{M}_k$ to a distribution on outputs $\{0,1\}$.

We say that Π *computes a function* $f : \mathcal{X}_1 \times \ldots \times \mathcal{X}_k \to \{0,1\}$ *with error* ϵ if for each $(x_1, \ldots, x_k) \in \mathcal{X}_1 \times \ldots \times \mathcal{X}_k$ we have:

$$\Pr_{\{m_i \sim \pi_i(x_i)\}_{i \in [k]}} [O(m_1, \ldots, m_k) \neq f(x_1, \ldots, x_k)] \leq \epsilon.$$

A *deterministic* protocol is defined as above, except that instead of distributions on messages, the protocol maps each player's input to a deterministic message, and the referee's output is also a deterministic function of the messages it receives from the players.

Communication complexity. The *communication complexity* of a protocol Π (randomized or deterministic), denoted by $\mathrm{CC}(\Pi)$, is defined as the maximum total number of bits sent by the players to the referee in any execution of the protocol on any input.[†]

For a function f, the *deterministic communication complexity of f* is defined as

$$D(f) = \min_{\Pi} \mathrm{CC}(\Pi),$$

where the minimum is taken over all deterministic protocols that compute f with no errors. The *private-coin ϵ-error communication complexity of f* is defined as

$$R_\epsilon(f) = \min_{\Pi:\Pi \text{ computes } f \text{ with error } \epsilon} \mathrm{CC}(\Pi).$$

Individual communication complexity of a player. We let $\mathrm{CC}_i(\Pi)$ denote the maximum number of bits sent by player i to the referee in any execution. For general communication protocols, it could be that the players never simultaneously reach their worst-case message sizes — that is, we could have $\mathrm{CC}(\Pi) < \sum_{i=1}^{k} \mathrm{CC}_i(\Pi)$. However, with *simultaneous* protocols this cannot happen:

Observation 1. *For any private-coin (or deterministic) simultaneous protocol Π we have $\mathrm{CC}(\Pi) = \sum_{i=1}^{k} \mathrm{CC}_i(\Pi)$.*

Proof. For each $i \in [k]$, let x_i be some input on which player i sends $\mathrm{CC}_i(\Pi)$ bits with non-zero probability. Then on joint input $(x_1, \ldots, x_k)$, there is a non-zero probability that each player i sends $\mathrm{CC}_i(\Pi)$ bits, for a total of $\sum_{i=1}^{k} \mathrm{CC}_i(\Pi)$ bits. Therefore $\mathrm{CC}(\Pi) \geq \sum_{i=1}^{k} \mathrm{CC}_i(\Pi)$. The inequality in the other direction is immediate, as there cannot be an execution of the protocol in which more than $\sum_{i=1}^{k} \mathrm{CC}_i(\Pi)$ bits are sent. $\square$

In the sequel we assume for simplicity that all players always send the same number of bits, that is, each player has a fixed message size. By the observation above, this does not change the communication complexity.

Maximal message complexity of a protocol. The *maximal message complexity* of a protocol Π is the maximum individual communication complexity over all players. The deterministic maximum message complexity is $D^\infty = \min_{\Pi} \max_i \mathrm{CC}_i(\Pi)$, and the *private-coin ϵ-error maximal message complexity of f* is defined as $R_\epsilon^\infty = \min_{\Pi \text{ computes } f \text{ with error } \epsilon} \max_i \mathrm{CC}_i(\Pi)$

Problem statements. The two main problems we consider in this paper are:

- $\textsc{AllEq}_{k,n}(x_1, \ldots, x_k) = 1$ iff $x_1 = \ldots = x_k$, where $x_1, \ldots, x_k \in \{0,1\}^n$;
- $\textsc{ExistsEq}_{k,n}(x_1, \ldots, x_k) = 1$ iff for some $i \neq j$ we have $x_i = x_j$, where $x_1, \ldots, x_k \in \{0,1\}^n$.

We often omit the subscript when the number of players and the input size are clear from the context.

3 Lower Bound

In this section we extend the lower bound from [3] to multiple players, and show that for any k-player function f and constant error probability $\epsilon \in (0, 1/2)$ we have $R_\epsilon(f) = \Omega(\sqrt{D(f)})$.

When proving two-party communication complexity lower bounds, it is helpful to view the function being computed as a matrix, where the rows are indexed by Alice's input, the columns are indexed by Bob's input, and each cell contains the value of the function on the corresponding pair of inputs.

[†]Another reasonable definition for randomized protocols is to take the maximum over all inputs of the *expected* total number of bits sent. For two players this is asymptotically equivalent to the definition above [13]. For $k > 2$ players, the expectation may be smaller than the maximum by a factor of $\log(k)$.

The natural extension to k players is a "k-dimensional matrix" (or tensor) where the i-th dimension is indexed by the inputs to the i-th player, and the cells again contain the values of the function on that input combination. For conciseness we refer to this representation as a "matrix" even for $k > 2$ players.

In [3] it is observed that the deterministic simultaneous communication complexity of a function is exactly the sum of the logarithms of the number of unique rows and the number of unique columns in the matrix (rounded up to an integer). We generalize the notion of "rows and columns" to multiple players as follows.

Definition 1 (Slice). *Fix a k-dimensional matrix $M \in \{0,1\}^{m_1 \times \ldots \times m_k}$. For a player i and an input (i.e., index) $x_i \in [m_i]$, we define* the (i, x_i)-th slice *of M to be the projection of M onto a $(k-1)$-dimensional matrix $M|_{(i,x_i)} \in \{0,1\}^{m_1 \times \ldots \times m_{i-1} \times m_{i+1} \times \ldots \times m_k}$ obtained by fixing player i's input to x_i. That is, for each $x \in \mathcal{X}_1 \times \ldots \times \mathcal{X}_k$ we have $M|_{(i,x_i)}(x_{-i}) = M(x)$.*

Note that for $k = 2$ and a 2-dimensional matrix M, the $(1, x)$-th slice of M is simply the row indexed by x, and the $(2, y)$-th slice is the column indexed by y.

We assume that the matrices we deal with have *no redundant slices*: there does not exist a pair $(i, x_i), (i, x'_i)$ (where $x_i \neq x'_i$) such that $M|_{(i,x_i)} = M|_{(i,x'_i)}$. If there are redundant slices, we simply remove them; they correspond to inputs to player i on which the function value is the same, for any combination of inputs to the other players. Such inputs are "different in name only" and we can eliminate the redundancy without changing the communication complexity of the function being computed.

Let $\dim_i(M)$ denote the length of M in the i-th direction: this is the number of possible inputs to player i, after redundant slices are removed (i.e., the number of unique slices for player i in M). We rely upon the following observation, which generalizes the corresponding observation for two players from [3]:

Observation 2. *Let $f : \mathcal{X}_1 \times \ldots \times \mathcal{X}_k \to \{0,1\}$ be a k-player function, and let M_f be the matrix representing f. Then in any deterministic protocol for f, each player i sends at least $\log \dim_i(M_f)$ bits in the worst case, and $D(f) = \sum_{i=1}^{k} \lceil \log \dim_i(M_f) \rceil$.*

Proof. Suppose for the sake of contradiction that there is a deterministic protocol Π for f where some player i that always sends fewer than $\lceil \log \dim_i(M_f) \rceil$ bits in Π. For this player there exist two slices (i.e., inputs to player i) $M|_{(i,x_i)}$ and $M|_{(i,x'_i)}$, with $x_i \neq x'_i$, on which the player sends the same message. Because we assumed that there are no redundant slices, there exists an input x_{-i} to the other players such that $M|_{(i,x_i)}(x_{-i}) \neq M|_{(i,x'_i)}(x_{-i})$. But all players send the same messages to the referee on inputs (x_i, x_{-i}) and (x'_i, x_{-i}), which means that on one of the two inputs the output of the referee is incorrect.

This shows that each player i must send at least $\lceil \log \dim_i(M_f) \rceil$ bits in the worst-case. This number of bits from each player is also sufficient to compute f, as the players can simply send the referee their input (after removing redundant slices, the number of remaining inputs is the number of unique slices). Therefore by Observation 1, $D(f) = \sum_{i=1}^{k} \lceil \log \dim_i(M_f) \rceil$. ☐

In [3], Babai and Kimmel prove the following for two players:

Lemma 3.1 ([3]). *For any 2-player private-coin protocol Π with constant error $\epsilon < 1/2$,*

$$\mathrm{CC}_1(\Pi) \cdot \mathrm{CC}_2(\Pi) \geq \Omega(\log \dim_1(M_f) + \log \dim_2(M_f)).$$

Using this property of 2-player protocols, we can show:

Lemma 3.2. *Let Π be a k-player private-coin protocol for $f : \mathcal{X}_1 \times \ldots \times \mathcal{X}_k \to \{0,1\}$ with constant error $\epsilon \in (0, 1/2)$. Then for each $i \in [k]$:*

$$\mathrm{CC}_i(\Pi) \cdot \left(\sum_{j \neq i} \mathrm{CC}_j(\Pi) \right) = \Omega(\log \dim_i(M_f)).$$

Proof. Fix a player $i \in [k]$. The k-player protocol Π induces a 2-player protocol Π', where Alice plays the role of player i, and Bob plays the role of all the other players. We have $\mathrm{CC}_1(\Pi') = \mathrm{CC}_i(\Pi)$ and $\mathrm{CC}_2(\Pi') = \sum_{j \neq i} \mathrm{CC}_j(\Pi)$ (recall that we assume the message size of each player is fixed).

The 2-player function computed by Π' is still f, but now we view it as a 2-player function, represented by a 2-dimensional matrix M'_f with rows indexed by $\mathcal{X}_i$ and columns indexed by $\mathcal{X}_1 \times \ldots \times \mathcal{X}_{i-1} \times \mathcal{X}_{i+1} \times \ldots \times \mathcal{X}_k$. Note that $\dim_1(M'_f) \geq \dim_i(M_f)$: if $M_f|_{(i,x_i)}$ and $M_f|_{(i,x'_i)}$ are slices of M_f that are not equal, then the corresponding rows of M'_f, indexed by x_i and x'_i, differ as well. Thus, by Lemma 3.1,

$$\mathrm{CC}_i(\Pi) \cdot \left(\sum_{j \neq i} \mathrm{CC}_j(\Pi) \right) = \mathrm{CC}_1(\Pi') \cdot \mathrm{CC}_2(\Pi') = \Omega(\log \dim_1(M'_f)) = \Omega(\log \dim_i(M_f)).$$

□

We can now show:

Theorem 3.3. *For any k-player function f and constant error $\epsilon < 1/2$ we have $R_\epsilon(f) = \Omega(\sqrt{D(f)})$.*

Proof. Let Π be an ϵ-error private-coin simultaneous protocol for f. By the lemma, for each $i \in [k]$ we have

$$\mathrm{CC}_i(\Pi) \cdot \left(\sum_{j=1}^{n} \mathrm{CC}_j(\Pi) \right) \geq \mathrm{CC}_i(\Pi) \cdot \left(\sum_{j \neq i} \mathrm{CC}_j(\Pi)) \right) = \Omega\left(\log \dim_i(M_f)\right).$$

Summing across all players, we obtain

$$\left(\sum_{i=1}^{n} \mathrm{CC}_i(\Pi) \right) \cdot \left(\sum_{j=1}^{n} \mathrm{CC}_j(\Pi) \right) = \Omega\left(\sum_{i=1}^{n} \log \dim_i(M_f) \right),$$

that is, by Observations 1 and 2,

$$\mathrm{CC}(\Pi)^2 = \Omega\left(D(f)\right).$$

The theorem follows. □

From the theorem above we see that the *average* player must send $\Omega(\sqrt{D(f)/k})$ bits. But what is the relationship between the *maximum* number of bits sent by any player in a private-coin protocol and a deterministic protocol for f? This question is mainly of interest for non-symmetric functions, since for symmetric functions all players must send the same number of bits in the worst-case.

Theorem 3.4. *For any k-player function f and constant error ϵ we have $R^\infty_\epsilon(f) = \Omega(\sqrt{D^\infty(f)/k})$.*

Proof. Recall that by Observation 2, $D^\infty(f) = \max_i \log \dim_i(M_f)$. Let i be a player maximizing $\log \dim_i(M_f)$. As we showed in the proof of Theorem 3.3, for this player we have in any private-coin simultaneous protocol Π:

$$\mathrm{CC}_i(\Pi) \cdot \left(\sum_{j=1}^{n} \mathrm{CC}_j(\Pi) \right) = \Omega\left(\log \dim_i(M_f)\right) = \Omega(D^\infty(f)).$$

Now let ℓ be the player with the maximum communication complexity in Π, that is, $\mathrm{CC}_j(\Pi) \leq \mathrm{CC}_\ell(\Pi)$ for each $j \in [k]$. We then have

$$\mathrm{CC}_i(\Pi) \cdot \left(\sum_{j=1}^{n} \mathrm{CC}_j(\Pi) \right) \leq \mathrm{CC}_\ell(\Pi) \cdot (k-1)\mathrm{CC}_\ell(\Pi) < k\mathrm{CC}^2_\ell(\Pi).$$

Combining the two, we obtain $\mathrm{CC}_\ell(\Pi) = \Omega\left(\sqrt{D^\infty(f)/k}\right)$, which proves the theorem. $\square$

Lower Bound of $\Omega(k \log n)$ for $\textsc{AllEq}_{k,n}$

We next show that in the specific case of $\textsc{AllEq}$, each player needs to send at least $\Omega(\log n)$ bits, yielding a lower bound of $\Omega(k \log n)$. This improves on the lower bound of Theorem 3.3 when $k = \Omega(n/\mathrm{polylog}(n))$, and will show that the protocol in the next section is optimal.

Theorem 3.5. *For any constant $\epsilon < 1/2$ we have $R_\epsilon(\textsc{AllEq}_{k,n}) = \Omega(k \log n)$.*

Proof. Fix a player $i \in [k]$. To show that player i must send $\Omega(\log n)$ bits, we reduce from $\textsc{Eq}_n$, but this time our reduction constructs a *one-way protocol*, where Alice, taking the role of player i, sends a message to Bob, representing all the other players and the referee; and Bob then outputs the answer. It is known that $\textsc{Eq}_n$ requires $\Omega(\log n)$ bits of communication for private-coind protocols — this is true even with unrestricted back-and-forth communication between the two players [13]. The lower bound follows.

Let Π be a simultaneous private-coin protocol for $\textsc{AllEq}_{k,n}$. We construct a one-way protocol for $\textsc{Eq}_n$ as follows: on input (x, y), Alice sends Bob the message that player i would send on input x in Π. Bob computes the messages each player $j \neq i$ would send on input y, and then computes the output of the referee; this is the output of the one-way protocol. Clearly, $\textsc{AllEq}_{k,n}(x, y, \ldots, y) = \textsc{Eq}_n(x, y)$, so the one-way protocol succeeds whenever Π succeeds.

$\square$

The lower bounds above use a series of 2-player reductions; they do not seem to exploit the full "hardness" of having k players with their own individual private randomness. This makes it more surprising that the lower bounds are tight, as we show in the next section.

4 Tight Upper bound for $\textsc{AllEq}$

In this section, we show that the lower bound proven in section 3 is tight for $\textsc{AllEq}_{k,n}$. This is done by showing a protocol with maximal message of size $O(\sqrt{\frac{n}{k}} + \log(\min(n,k)))$ bits per player, and $O(\sqrt{nk} + k\log(\min(n,k)))$ bits of communication overall.

Theorem 4.1. *There exists a private-coin one sided error randomized simultaneous protocol for* $\textsc{AllEq}_{k,n}$ *with maximal message of size* $O(\sqrt{\frac{n}{k}} + \log(\min(n,k))) = O(\sqrt{\frac{D^\infty(\textsc{AllEq}_{k,n})}{k}} + \log(\min(n,k)))$ *bits per player.*

Corollary 4.2. *There exists a private-coin one sided error randomized simultaneous protocol for* $\textsc{AllEq}_{n,k}$ *of cost* $O(\sqrt{nk} + k\log(\min(n,k))) = O(\sqrt{D(\textsc{AllEq}_{k,n})} + k\log(\min(n,k)))$.

We note that the *deterministic* communication complexity of $\textsc{AllEq}_{n,k}$ is $\Theta(nk)$, and hence also $D^\infty(\textsc{AllEq}_{k,n}) = \Theta(n)$. This follows immediately from Observation 2.

Our randomized private-coin protocol is as follows.

Error-correcting codes. In the first step of the protocol, each player encodes its input using a predetermined error correcting code, and uses the encoded string as the new input. We review the definition of an error correcting code. In the definition below, n and k are the standard notation for error correcting codes, which we keep for the sake of consistency with the literature in coding; they are unrelated to the parameters n, k of the communication complexity problem and will be used in this context in the following definition only.

Definition 2 ([14]). *$M \subseteq \{0,1\}^n$ is called an $[n,k,d]$*-code *if it contains 2^k elements (that is, $|M| = 2^k$) and $d_H(x,y) \geq d$ for every two distinct x, y, where d_H is the Hamming distance. For a $[n,k,d]$ code, let $\delta = \frac{d}{n}$ denote the* relative distance *of the code.*

An $[n,k,d]$-code maps each of 2^k inputs to a code word of n bits, such that any two distinct inputs map to code words that have large relative distance. We use a simple error-correcging code (see [14]), which was also used in [2]:

Lemma 4.3 ([14], Theorem 17.30[‡]). *For each $m \geq 1$ there is a $[3m, m, \frac{m}{2}]$-code.*

The relative distance of the code in lemma 4.3 is $\delta = \frac{(1/2)m}{3m} = \frac{1}{6}$.

When the players use the code from Lemma 4.3 to encode their inputs, each player's input grows by a constant factor (3), while the relative Hamming distance of any two differing inputs becomes at least δ. Let $N = 3n$ denote the length of the encoded inputs, and let $\bar{x}_i$ denote the encoding of player i's input x_i.

Partitioning into blocks.

After computing the encoding of their inputs, each player splits its encoded input into blocks of $L = \lceil \frac{N}{k} \rceil$ bits each, except possibly the last block, which may be shorter. For simplicity we assume here that all blocks have the same length, that is, L divides n. Let $b = N/L$ be the resulting number of blocks; we note that $b \leq \min(3n, k)$. Let $B_i(\ell) \in \{0,1\}^L$ denote the ℓ-th block of player i.

Because any two differing inputs have encodings that are far in Hamming distance, we can show that two players with different inputs will also disagree on many *blocks*:

Lemma 4.4. *For any two players i, j such that $x_i \neq x_j$, we have $|\{\ell \in [b] \mid B_i(\ell) \neq B_j(\ell)\}| \geq \delta b$.*

Proof. Assume for the sake of contradiction that $|\{\ell \in [b] | B_i(\ell) \neq B_j(\ell)\}| < \delta b$.

Let $\Delta = \{s \in [N] \mid \bar{x}_i(s) \neq \bar{x}_j(s)\}$ be the set of coordinates on which players i, j disagree. By the properties of the error correcting code, $|\Delta| \geq \delta N$.

Now partition Δ into disjoint sets $\Delta_1, \dots, \Delta_b$, where each Δ_ℓ contains the locations inside block ℓ on which the encoded inputs disagree. Each Δ_ℓ contains between 0 and N/b coordinates, as the size of each block is $L = N/b$. By our assumption, there are fewer than δb blocks that contain any differences, so the number of non-empty sets Δ_ℓ is smaller than δb. It follows that $|\Delta| < \delta b \cdot (N/b) = \delta N$, which contradicts the relative distance of the code. □

Comparing blocks.

Our goal now is to try to find two players that disagree on some block. We know that if there are two players with different inputs, then they will disagree on many different blocks, so choosing a random block will expose the difference with good probability. In order to compare the blocks, we use an optimal 2-player private-coin simultaneous protocol for EQ:

Theorem 4.5 ([3] Theorem 1.5). *There exists a private-coin one-sided error simultaneous protocol for the two player function EQ_m of cost $\Theta(\sqrt{m})$. If the inputs are equal, the protocol always outputs "Equal". If the inputs are not equal, then the protocol outputs "Equal" with probability $< 1/3$.*

Remark. *We refer here to the symmetric variant of the equality protocol in Remark 3.3 of [3], in which both Alice and Bob use the same algorithm to compute their outputs.*

We proceed as follows. Each player i chooses a block $\ell \in [b]$ at random. The player applies Alice's algorithm from [3]'s 2-player equality protocol on the chosen block $B_i(\ell)$, and sends the output to the referee, along with the index ℓ of the block. In this process each player sends $O(\sqrt{\frac{n}{k}} + \log(\min(n,k)))$ bits, because the length of a block is $L = O(n/k)$, and $b \leq \min(3n, k)$.

The referee receives the player's outputs $o_1, ..., o_k$, and for each pair that chose the same block index, it simulates [3]'s 2-player equality referee. If for all such pairs the output is 1 then the referee also outputs 1, otherwise it outputs 0. Let us denote by $Ref(o_1, \dots, o_k)$ the referee's output function.

[‡]The theorem in [14] gives a general construction for any distance up to $1/2$; here we use distance $1/6$.

Analysis of the error probability. Note that if all inputs are equal, then our protocol always outputs 1: the $\textsc{Eq}_L$ protocol from [3] has one-sided error, so in this case it will output 1 for any pair of blocks compared. On the other hand, if there exist two different inputs, we will only detect this if (a) the two corresponding players choose a block on which their encoded inputs differ, and (b) the $\textsc{Eq}_L$ protocol from [3] succeeds and outputs 0. We show that this does happen with good probability:

Lemma 4.6. *If* $\textsc{AllEq}(x_1, ..., x_k) = 0$*, then the protocol outputs* 0 *with probability at least* $\frac{2}{3}\delta(1 - e^{-\frac{1}{2}})$.

Proof. Since there are at least two distinct input strings, there exists an input string received by at most half the players. Let i be a player with such a string, and let $j'_1, ..., j'_{\frac{k}{2}}$ be $\frac{k}{2}$ players that disagree with player i's input.

Let A_t be the event that player j'_t chose the same block index as player i. Then

$$\Pr\left[Ref(o_1, ..., o_k) = 0\right] \geq \Pr\left[Ref(o_1, ..., o_k) = 0 \,\middle|\, \bigcup_{t=1}^{k/2} A_t\right] \cdot \Pr\left[\bigcup_{t=1}^{k/2} A_t\right].$$

We bound each of these two factors individually.

Since all A_t's are independent, and for a fixed t we have $Pr[A_t] = \frac{1}{b}$ then

$$\Pr\left[\bigcup_{t=1}^{k/2} A_t\right] = 1 - \left(1 - \frac{1}{b}\right)^{k/2} \geq 1 - \left(1 - \frac{1}{k}\right)^{k/2} \geq 1 - \left(e^{-1/k}\right)^{k/2} = 1 - e^{-1/2}.$$

Next, let us condition on the fact that some specific A_r occurred. Given that at least one of the A_t's occurred, let A_r be such an event, that is, player r chose the same block as player i.

Clearly, conditioning on A_r does not change the probability of each block being selected, because the blocks are chosen uniformly and independently: that is, for each $i, r \in [k]$ and $\ell \in [b]$,

$$\Pr\left[\text{player } i \text{ chose block } \ell \mid A_r\right] = \frac{1}{b}.$$

Therefore, by Lemma 4.4, given the event A_r, players i and r disagree on the block they sent with probability at least $(\delta b)/b = \delta$. Whenever i and r send a block they disagree on, the protocol from [3] outputs 0 with probability $2/3$. So overall,

$$\Pr\left[Ref(o_1, ..., o_k) = 0 \,\middle|\, \bigcup_{t=1}^{\frac{k}{2}} A_t\right] \geq \frac{2}{3}\delta.$$

Combining the two yields $Pr[Ref(o_1, ..., o_k) = 0] \geq \frac{2}{3}\delta(1 - e^{-\frac{1}{2}})$. ☐

Proof of Theorem 4.1. By lemma 4.6, if $\textsc{AllEq}(x_1, ..., x_k) = 0$ the algorithm errs with constant probability. If $\textsc{AllEq}(x_1, ..., x_k) = 1$ then since $\forall_{i,p,p'} B_p(i) = B'_p(i)$, and the fact that [3]'s protocol is a one-sided error protocol, the global protocol will always output 1, which is the correct value. Since this is a one-sided error protocol with constant error probability, this protocol can be amplified by repeating the protocol in parallel a constant number of times, so that the error probability becomes an arbitrarily small constant, and the communication is only increased by a constant factor. ☐

5 On ExistsEq

The upper bound of Section 4 reduces the AllEq problem to a collection of 2-player Eq problems, which can then be solved efficiently using known protocols (e.g., from [3]). This works because asking whether *all* inputs are equal, and finding any pair of inputs that are not equal is sufficient to conclude that the answer is "no". What is the situation for the ExistsEq problem, where we ask whether there *exists* a pair of inputs that are equal? Intuitively the approach above should not help, and indeed, the complexity of ExistsEq is higher:

Theorem 5.1. *If $k \leq 2^{n-1}$, then $R_\epsilon(\textsc{ExistsEq}_{k,n}) = \Omega(k\sqrt{n})$ for any constant $\epsilon < 1/2$.*

Proof. We show that each player i must send $\Omega(\sqrt{n})$ bits in the worst case, and the bound then follows by Observation 1. The proof is by reduction from 2-player $\textsc{Eq}_{n-1}$ (we assume that $n \geq 2$).

Let Π be a private-coin simultaneous protocol for $\textsc{ExistsEq}_{k,n}$ with error $\epsilon < 1/2$. Consider player 1 (the proof for the other players is similar). Assign to each player $i \in \{3, \ldots, k\}$ a unique label $b_i \in \{0,1\}^{n-1}$ (this is possible because $k \leq 2^{n-1}$.

We construct a 2-player simultaneous protocol Π' for $\textsc{Eq}_{n-1}$ with error $\epsilon < 1/2$ as follows: on inputs $(x, y) \in \{0,1\}^{n-1}$, Alice plays the role of player 1 in Π, feeding it the input $1x$ (that is, the n-bit vector obtained by prepending '1' to x); Bob plays the role of player 2 with input $1y$; and the referee in Π' plays the role of all the other players and the referee in Π, feeding each player i the input $0b_i$, where b_i is the unique label assigned to player i.

After receiving messages from Alice and Bob, and sampling the messages players $3, \ldots, k$ would send in Π when given the inputs described above, the referee computes the output of Π and that is also the output of Π'.

Because we prefixed the true inputs x, y with 1, and players $3, \ldots, k$ received inputs beginning with 0, we have $\textsc{ExistsEq}(1x, 1y, 0b_3, \ldots, 0b_k) = \textsc{Eq}(x, y)$. Therefore Π' succeeds whenever Π does, and in particular it has error at most ϵ. By the lower bound of [3], then, player 1 must send $\Omega(\sqrt{n})$ bits in the worst case. □

We note that if $k \geq 2^n$ then ExistsEq is trivial, as there must always exist two players with the same input. (The depencence of k on n in Theorem 5.1 can be improved to $k \leq (1 - o(1)) \cdot 2^n$ by assigning inputs to player $3, \ldots, k$ more cleverly.)

The lower bound above is tight up to $O(\log k)$, and indeed we can state something more general: for any 2-player function $f : \mathcal{X} \times \mathcal{Y} \to \{0,1\}$, let $\exists_k f$ be the k-player function that outputs 1 on input $(x_1, \ldots, x_k)$ iff for some $i, j \in [k]$ we have $f(x_i, x_j) = 1$.

Lemma 5.2. *For any 2-player function f and $k \geq 2$ we have $R_{k^2\epsilon}(\exists_k f) = O(k \cdot R_\epsilon(f))$.*

Proof. Let $\Pi = (\Pi_A, \Pi_B, O)$ be a 2-player private-coin simultaneous protocol for f with communication complexity $R_\epsilon(f)$.

We construct a protocol Π' for $\exists_k f$ as follows: on input $(x_1, \ldots, x_k)$, each player i samples two messages, $M_A^i \sim \Pi_A(x_i)$ and $M_B^i \sim \Pi_B(x_i)$, and sends them to the referee. The referee samples outputs $Z_{i,j} \sim O(M_A^i, M_B^j)$ for each $(i,j) \in [k]^2$ $(i \neq j)$ and outputs "1" iff for some pair $Z_{i,j} = 1$.

If $\exists_k f(x_1, \ldots, x_k) = 1$, then there exist i, j such that $f(x_i, x_j) = 1$, and for this pair of players we have $\Pr[Z_{i,j} = 0] \leq \epsilon$. Therefore the referee outputs "1" except with probability ϵ.

On the other hand, if $\exists_k f(x_1, \ldots, x_k) = 0$, then for every pair i, j of players we have $f(x_i, x_j) = 0$, so $\Pr[Z_{i,j} = 1] \leq \epsilon$. By union bound, in this case the probability that the referee outputs "1" is bounded by $\binom{k}{2}\epsilon < k^2\epsilon$. □

To handle the increased error of the protocol for $\exists_k f$, we can use a protocol for f that has error $O(1/k^2)$; this is achieved by taking a constant-error protocol for f, executing it $O(\log k)$ independent times, and taking the majority output [13]. We obtain the following:

Theorem 5.3. *For any 2-player function f, $k \geq 2$, and constant $\epsilon < 1/2$ we have $R_\epsilon(\exists_k f) = O(k \log k \cdot R_\epsilon(f))$.*

Corollary 5.4. *For* ExistsEq *we have $R_\epsilon(\textsc{ExistsEq}) = O(k \log k\sqrt{n})$, matching our lower bound up to a logarithmic factor.*

6 Conclusion

In this paper we extended the classical results of Babai and Kimmel [3] to the multi-player setting, and gave a tight bound for the gap between private-coin and deterministic communication complexity in the simultaneous setting. We showed that contrary to our initial expectations, the gap does not grow larger with the number of players, and indeed the per-player gap can shrink as the number of players grows. We also addressed a class of functions defined by taking a two-party function and asking whether there are two players whose inputs cause it to output 1.

Our work leaves open the interesting question of simultaneous lower bounds for the model considered in [1, 5, 6], where each player represents a node in a graph and is given the edges adjacent to that node. Our techniques do not apply to this scenario because of the sharing of edges between players. Indeed, the connectivity problem for this model (see [15]) cannot be addressed by reductions from two-player communication complexity, because it is easy for two players: we can simply have Alice and Bob compute spanning forests for their part of the input and send them to each other, at a total cost of $O(n \log n)$. Thus, further multi-party techniques need to be developed to address the hardness of connectivity.

References

[1] Kook Jin Ahn, Sudipto Guha, and Andrew McGregor. Graph sketches: Sparsification, spanners, and subgraphs. In *Proceedings of the 31st Symposium on Principles of Database Systems*, PODS '12, pages 5–14, 2012.

[2] A. Ambainis. Communication complexity in a 3-computer model. *Algorithmica*, 16(3):298–301, 1996.

[3] L. Babai and P. G. Kimmel. Randomized simultaneous messages: Solution of a problem of yao in communication complexity. In *Proceedings of the 12th Annual IEEE Conference on Computational Complexity*, CCC '97, pages 239–. IEEE Computer Society, 1997.

[4] László Babai, Anna Gál, Peter G. Kimmel, and Satyanarayana V. Lokam. Communication complexity of simultaneous messages. *SIAM J. Comput.*, 33(1):137–166, 2004.

[5] Florent Becker, Martín Matamala, Nicolas Nisse, Ivan Rapaport, Karol Suchan, and Ioan Todinca. Adding a referee to an interconnection network: What can(not) be computed in one round. In *25th IEEE International Symposium on Parallel and Distributed Processing, IPDPS 2011*, pages 508–514, 2011.

[6] Florent Becker, Pedro Montealegre, Ivan Rapaport, and Ioan Todinca. *Structural Information and Communication Complexity: 21st International Colloquium, SIROCCO 2014, Takayama, Japan, July 23-25, 2014. Proceedings*, chapter The Simultaneous Number-in-Hand Communication Model for Networks: Private Coins, Public Coins and Determinism, pages 83–95. 2014.

[7] J. Bourgain and A. Wigderson. Personal communication (see [3]).

[8] A. Chakrabarti, Yaoyun Shi, A. Wirth, and A. Yao. Informational complexity and the direct sum problem for simultaneous message complexity. In *Proceedings of the Fourty-Second Annual Symposium on Foundations of Computer Science*, FOCS '01, pages 270–278, 2001.

[9] Ashok K. Chandra, Merrick L. Furst, and Richard J. Lipton. Multi-party protocols. In *Proceedings of the Fifteenth Annual ACM Symposium on Theory of Computing*, STOC '83, pages 94–99, 1983.

[10] Arkadev Chattopadhyay, Jaikumar Radhakrishnan, and Atri Rudra. Topology matters in communication. In *55th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2014, Philadelphia, PA, USA, October 18-21, 2014*, pages 631–640, 2014.

[11] Rahul Jain and Hartmut Klauck. New results in the simultaneous message passing model via information theoretic techniques. In *Proceedings of the Twenty-Fourth Annual IEEE Conference on Computational Complexity, 2009*, CCC '09, pages 369 – 378, 2009.

[12] Ilan Kremer, Noam Nisan, and Dana Ron. On randomized one-round communication complexity. In *Proceedings of the Twenty-seventh Annual ACM Symposium on Theory of Computing*, STOC '95, pages 596–605, 1995.

[13] Eyal Kushilevitz and Noam Nisan. *Communication Complexity*. Cambridge University Press, 1997.

[14] F.J. MacWilliams and N.J.A. Sloane. *The Theory of Error Correcting Codes*. North-Halland, 1981.

[15] Andrew McGregor. Open problem 65. List of Open Problems in Sublinear Algorithms. `http://sublinear.info/index.php?title=Open_Problems:65`.

[16] Ilan Newman and Mario Szegedy. Public vs. private coin flips in one round communication games (extended abstract). In *Proceedings of the Twenty-eighth Annual ACM Symposium on Theory of Computing*, STOC '96, pages 561–570, 1996.

[17] Omri Weinstein and David P. Woodruff. *Automata, Languages, and Programming: 42nd International Colloquium, ICALP 2015, Kyoto, Japan, July 6-10, 2015, Proceedings, Part I*, chapter The Simultaneous Communication of Disjointness with Applications to Data Streams, pages 1082–1093. 2015.

[18] Andrew Chi-Chih Yao. Some complexity questions related to distributive computing(preliminary report). In *Proceedings of the Eleventh Annual ACM Symposium on Theory of Computing*, STOC '79, pages 209–213, 1979.