Multiparty Equality Function Computation in Networks with Point-to-Point Links*

Guanfeng Liang and Nitin Vaidya

Department of Electrical and Computer Engineering, and
Coordinated Science Laboratory
University of Illinois at Urbana-Champaign, USA
{gliang2,nhv}@illinois.edu

Abstract. In this paper, we study the problem of computing the multiparty equality (MEQ) function: $n \geq 2$ nodes, each of which is given an input value from $\{1, \cdots, K\}$, determine if their inputs are all identical, under the point-to-point communication model. The MEQ function equals to 1 if and only if all n inputs are identical, and 0 otherwise. The communication complexity of the MEQ problem is defined as the minimum number of bits communicated in the worst case. It is easy to show that $(n-1)\log_2 K$ bits is an upper bound, by constructing a simple algorithm with that cost. In this paper, we demonstrate that communication cost strictly lower than this upper bound can be achieved. We show this by constructing a *static* protocol that solves the MEQ problem for $n = 3, K = 6$, of which the communication cost is strictly lower than the above upper bound ($2\log_2 6$ bits). This result is then generalized for large values of n and K.

1 Introduction

In this paper, we study the problem of computing the following multiparty equality function (MEQ):

$$MEQ(x_1, \cdots, x_n) = \begin{cases} 0 \text{ if } x_1 = \cdots = x_n \\ 1 \text{ otherwise.} \end{cases} \quad (1)$$

The input vector $x = (x_1, \cdots, x_n)$ is distributed among $n \geq 2$ nodes, with only x_i known to node i, and each x_i chosen from the set $\{1, \cdots, K\}$, for some integer $K \geq 1$.

Communication Complexity: The notion of communication complexity (CC) was introduced by Yao in 1979 [12]. They investigated the problem of quantifying the number of bits that two separated parties need to communicate between

* This research is supported in part by Army Research Office grant W-911-NF-0710287 and National Science Foundation award 1059540. Any opinions, findings, and conclusions or recommendations expressed here are those of the authors and do not necessarily reflect the views of the funding agencies or the U.S. government.

themselves in order to compute a function whose inputs, namely X and Y, are distributed between them.

The communication cost of a protocol P, denoted as $C(P)$, is the number of bits exchanged for the worst case input pair. The communication complexity of a Boolean function $f : X \times Y \mapsto \{0, 1\}$, is minimum of the cost of the protocols for f.

Multiparty Function Computation: The notion of communication complexity can be easily generalized to a multiparty setting, i.e., when the number of parties > 2.

The communication complexity of a Boolean function $f : X_1 \times \cdots \times X_n \mapsto \{0, 1\}$, is minimum of the cost of the protocols for f.

There are more than one communication models for the multiparty problems. Two commonly used models are the "number on the forehead" model [4] and the "number in hand" model. Consider function $f : X_1 \times \cdots \times X_n \mapsto \{0, 1\}$, and input $(x_1, x_2, \cdots, x_n)$ where each $x_i \in X_i$. In the number on the forehead model, the i-th party can see all the x_j such that $j \neq i$; while in the number in hand model, the i-th party can only see x_i. As in the two-party case, the n parties have an agreed-upon protocol for communication, and all this communication is posted on a "public blackboard". In these two models, the communication may be considered as being *broadcast* using the public blackboard, i.e., when any party sends a message, all other parties receive the same message. Tight bounds often follow from considering two-way partitions of the set of parties.

In this paper, we consider a different point-to-point communication model, in which nodes communicate over *private* point-to-point links. This means that when a party transmits a message on a point-to-point link, only the party on the other end of the link receives the message. This model makes the problem significantly different from that with the broadcast communication model. We are interested in the communication complexity of the MEQ problem under the point-to-point communication model.

2 Related Work

The 2-party version of the MEQ problem (i.e., $n = 2$), which is usually referred to as the EQ problem, was first introduced by Yao in [12]. It is shown that the communication complexity of the EQ problem with *deterministic* algorithms is $\log K$ [6]. The complexity of the EQ problem can be reduced to $O(\log \log K)$ if randomized algorithms are allowed [6]. MEQ problem with $n \geq 3$ has been studied under the *number on the forehead* model and the *number in hand* model, both assuming a "public blackboard" for broadcast communications. The MEQ problem with $n \geq 3$ can be solved with cost of 2 bits [6] under the number on the forehead head model, while it requires $\Theta(\log K)$ bits under the number in hand model. On the other hand, the result changes significantly if we consider

the point-to-point communication model used in this paper (It is easy to show at least $\Omega(n \log K)$ bits are needed.).

The MEQ problem is related to the Set Disjointness problem and the consensus problem [8]. In the n-party Set Disjointness problem, we have n parties, and given subsets $S_1, \ldots, S_n \subseteq \{1, \ldots, K\}$, and the parties wish to determine if $S_1 \cap \cdots \cap S_n = \phi$ without communicating many bits. The disjointness problem is closely related to our MEQ problem. Consider the two-party set disjointness problem with subsets S_1 and S_2. Let x_1 and x_2 be the binary representations of S_1 and S_2, respectively. Then it is not hard to show that $x_1 = x_2$ is equivalent to $S_1 \cap \overline{S_2} = \phi$ and $\overline{S_1} \cap S_2 = \phi$. The multi-party set disjointness problem has been widely studied under the "number on the forehead" and broadcast communication model, e.g. [7, 11]. The set disjointness problem has also been studied under the "number in hand" model and point-to-point communication model (i.e., the same models we are using in this paper), with randomized algorithms. In [1], a lower bound of $\Omega(K/n^4)$ on its communication complexity is proved for randomized algorithms. The lower bound was then improved to $\Omega(K/n^2)$ in [2]. In [3], the authors established a further improved near-optimal lower bound of $\Omega(K/(n \log n))$. Nevertheless, these papers focus on the order of the communication complexity of randomized algorithms. On the other hand, in this paper, our goal is to characterize the *exact* communication complexity of *deterministic* algorithms.

In the Byzantine consensus problem, n parties, each of which is given an input x_i of $\log K$ bits, want to agree on a common output value x of $\log K$ bits under the point-to-point communication model, despite the fact that up to t of the parties may be *faulty* and deviate from the algorithm in arbitrary fashion [8]. The core of the consensus problem is to make sure that all fault-free parties' outputs are identical, which is essentially what the MEQ problem tries to solve. In our recent report [9], we established a lower bound on the communication complexity of the Byzantine consensus problem of n parties as a function of the communication complexity of the MEQ problem of $n - t$ parties. This motivates the MEQ problem under the point-to-point communication model. The consensus problem has also been studied under a slightly different fault-free model [5]. Authors of [5] investigated the fault-free consensus problem, which is essentially solving the MEQ problem with 1-bit inputs, i.e., $K = 2$, in tree topologies. We consider the problem under a more general setting with arbitrary K and do not assume any structure of the communication topology.

3 Models and Problem Definition

3.1 Communication Model

In this paper, we consider a point-to-point communication model. We assume a synchronous fully connected network of n nodes. We assume that all point-to-point communication channels/links are *private* such that when a node transmits, only the designated recipient can receive the message. The identity of the sender is known to the recipient.

3.2 Protocol

A protocol P is a schedule that consists of a sequence of steps. In each step l, as specified by the protocol, a pair of nodes are selected as the *transmitter* and *receiver*, denoted respectively as T_l and R_l. The transmitter T_l will send a message the receiver R_l. The message being sent is computed as a function $m_l(x_{T_l}, T_l^+(l))$, where x_{T_l} denotes T_l's input, and $T_l^+(l)$ denotes all the messages T_l has received so far. When it is clear from the context, we will use T_l^+ to denote $T_l^+(l)$ to simplify the presentation.

In this paper, we design protocols that are ***static*** : the triple $\langle T_l, R_l, m_l(\cdot)\rangle$ are pre-determined by the protocol and are independent of the inputs. In other words, in step l, no matter what the inputs are, the transmitter, receiver, and the function according to which the transmitter compute the message are the same. Since the schedule is fixed, a static protocol can be represented as a sequence of $L(P)$ steps: $\{\alpha_1, \alpha_2, \cdots, \alpha_{L(P)}\}$, where $\alpha_l = \langle T_l, R_l, m_l(x_{T_l}, T_l^+)\rangle$ in the l-th step. $L(P)$ is called the length of the protocol P, and P always terminates after the $L(P)$-th step. Denote $S_l(P)$ as the cardinality of $m_l()$, i.e., the number of possible channel symbols needed in step l of a static protocol P, considering all possible inputs. Then the communication cost of a static protocol P is determined by

$$C(P) = \sum_{l=1}^{L(P)} \log_2 S_l(P), \tag{2}$$

If only binary symbols are allowed, $S_l(P) = 2$ for all l, and $C(P)$ becomes $L(P)$.

3.3 Problem Definitions

We define two versions of the MEQ problem.

MEQ-AD (Anyone Detects): We consider protocols in which every node i decides on a one-bit output $EQ_i \in \{0, 1\}$. A node i is said to have detected a *mismatch* (or inequality of inputs) if it sets $EQ_i = 1$. A protocol P is said to solve the MEQ-AD problem if and only if **at least one** node detects a mismatch when the inputs to the n nodes are not identical. More formally, the following property must be satisfied when P terminates:

$$EQ_1 = \cdots = EQ_n = 0 \Leftrightarrow MEQ(x_1, \cdots, x_n) = 0. \tag{3}$$

MEQ-CD (Centralized Detect): The second class of protocols we consider are the ones in which one particular node is assigned to decide on an output. Without loss of generality, we can assume that node n has to compute the output. Then a protocol P is said to solve the MEQ-CD problem if and only if, when P terminates, node n computes output EQ_n such that

$$EQ_n = MEQ(x_1, \cdots, x_n). \tag{4}$$

Communication Complexity: Denote $\Gamma_{AD}(n,K)$ and $\Gamma_{CD}(n,K)$ as the sets of all protocols that solve the MEQ-AD and MEQ-CD problems with n nodes, respectively. We are interested in finding the communication complexity of the two versions of the MEQ problem, which is defined as the infimum of the communication cost of protocols in $\Gamma_{AD}(n,K)$ and $\Gamma_{AD}(n,K)$, i.e.,

$$C_{AD}(n,K) = \inf_{P\in\Gamma_{AD}(n,K)} C(P), \text{ and } C_{CD}(n,K) = \inf_{P\in\Gamma_{CD}(n,K)} C(P).$$

Communication Complexity with General Protocols: In general, a protocol that solves the MEQ problem may not necessarily be static. The schedule of transmissions might be determined dynamically on-the-fly, depending on the inputs. So the transmitter and receiver in a particular step l can be different with different inputs. Since the set of all static protocols is a subset of all general protocols, the communication complexities of the two versions of the MEQ problem are bounded from above by the cost of static protocols. The purpose of this paper is to show that there exist instances of the MEQ problem whose communication complexity is lower than the intuitive upper bound we are going to present in the next section. For this purpose, it suffices to show that, even if we constrain ourselves to static protocols, some MEQ problems can still be solved with cost lower than the upper bound. In sections 6 to 7, such examples of static protocols are presented.

4 Upper Bound of the Complexity

An upper bound of the communication complexity of both versions of the MEQ problem is $(n-1)\log_2 K$, for all positive integer $n \geq 2$ and $K \geq 1$. This can be proved by a trivial construction: in step i, node i sends x_i to node n, for all $i < n$. The decisions are computed according to

$$EQ_i = \begin{cases} MEQ(x_1,\cdots,x_n) & ,i = n; \\ 0 & ,i < n. \end{cases} \tag{5}$$

It is obvious that this protocol solves both the MEQ-AD and MEQ-CD problems with communication cost $(n-1)\log_2 K$, which implies $C_{AD(CD)}(n,K) \leq (n-1)\log_2 K$. In particular, when $K = 2^k$, we have $C_{AD(CD)}(n,2^k) \leq (n-1)k$.

For the two-party equality problem ($n = 2$), this bound is tight [6], for arbitrary K. The bound is also tight when $K = 2$ (binary inputs). $(n-1)\log_2 2 = n-1$ bits are necessary when $K = 2$, since any protocol with communication cost $< n-1$ will have at least one node not communicating with any other node at all, making it impossible to solve the MEQ problem. However, in the following sections, we are going to show that the $(n-1)\log K$ bound is not always tight, by presenting a static protocol that solves instances of the MEQ problem with communication cost lower than $(n-1)\log_2 K$.

5 Equivalent MEQ-AD Protocols

In the rest of this paper, except for section 8, we will focus on static protocols that solve the MEQ-AD problem.

It is not hard to see that a static protocol P can be interpreted as a directed multi-graph $G(V, E(P))$, where the set of vertices $V = \{1, \cdots, n\}$ represents the n nodes, and the set of directed edges $E(P) = \{(T_1, R_1), \cdots, (T_{L(P)}, R_{L(P)})\}$ represents the transmission schedule in each step. From now on, we will use the terms protocol and graph interchangeably, as well as the terms transmission and link. Fig.1(a) gives an example of the graph representation of a protocol for $n = 4$. In Fig.1(a), the numbers next to the directed links indicate the corresponding step numbers.

Two protocols P and P' in are said to be **equivalent** if their costs are equal, i.e., $C(P) = C(P')$. The following lemma says that we can flip the direction of any edge in $E(P)$ and obtain a protocol P' that is equivalent to P.

Lemma 1. *Given any static protocol P for MEQ-AD of length $L(P)$, and any positive integer $l \leq L(P)$, there exists a equivalent static protocol P' of the same length, such that $E(P)$ and $E(P')$ are identical, except that in the l-th step, the transmitter and receiver are swapped, i.e.,*

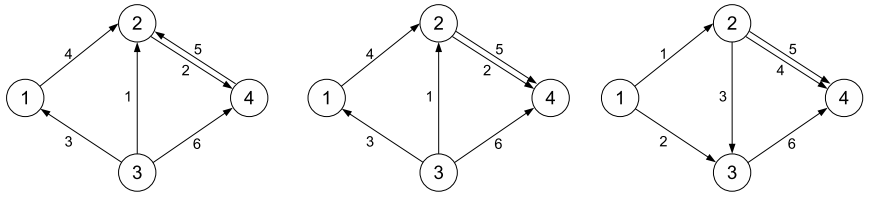
$$\begin{aligned} E(P') &= E(P)\backslash\{(T_l, R_l)\} \cup \{(R_l, T_l)\} \\ &= \{(T_1, R_1), \cdots, (T_{l-1}, R_{l-1}), (R_l, T_l), (T_{l+1}, R_{l+1}), \cdots, (T_{L(P)}, R_{L(P)})\}. \end{aligned}$$

Proof. Given the integer l and a protocol $P = \{\alpha_1, \cdots, \alpha_{l-1}, \alpha_l, \alpha_{l+1}, \cdots, \alpha_{L(P)}\}$ with $\alpha_l = (T_l, R_l, m_l(x_{T_l}, T_l^+))$, we construct $P' = \{\alpha'_1, \cdots, \alpha'_{l-1}, \alpha'_l, \alpha'_{l+1}, \cdots, \alpha'_{L(P)}\}$ by modifying P as follows:

- $\alpha'_j = \alpha_j$ for $1 \leq j \leq l-1$.
- $\alpha'_l = (R_l, T_l, m'_l(x_{R_l}, R_l^+))$. Here $m'_l(x_{R_l}, R_l^+) = m_l(x_{T_l}, T_l^+)|_{x_1=\cdots=x_n=x_{R_l}}$ is the symbol that party R_l *expects* to receive in step l of protocol P, assuming all parties have the same input as x_{R_l}.
- $\alpha'_j = (T_j, R_j, m'_j(x_{T_j}, T_j^+))$ for $j > l$.
  - If $T_j = R_l$, $m'_j(x_{T_j}, T_j^+) = m_j(x_{T_j}, T_j^+)|_{m_l(x_{T_l}, T_l^+)=m'_l(x_{R_l}, R_l^+)}$ is the symbol that party R_l sends in step j, **pretending** that it has received $m'_l(x_{R_l}, R_l^+)$ in step l of P.
  - If $T_j \neq R_l$, $m'_j(x_{T_j}, T_j^+) = m_j(x_{T_j}, T_j^+)$.

- To compute the output, T_l first computes EQ_{T_l} in the same way as in P. Then T_l sets $EQ_{T_l} = 1$ if $m'_l(x_{R_l}, R_l^+) \neq m_l(x_{T_l}, T_l^+)$, else no change. That is, T_l sets EQ_l to 1 if the symbol it receives from R_l in step l of P' differs from the symbol T_l would have sent to R_l in step l of P. The other nodes compute their outputs in the same way as in P.

To show that P and P' are equivalent, consider the two cases:



(a) Graph representation of P (b) Equivalent protocol of P with Step 5 flipped (c) iid equivalent protocol of P

Fig. 1: Example of graph representation of a protocol P and its equivalent protocols. The numbers next to the links indicate the corresponding step number.

- $m'_l(x_{R_l}, R_l^+) = m_l(x_{T_l}, T_l^+)$: It is not hard to see that in this case, the execution of every step is identical in both P and P', except for step l. So for all $i \neq T_l$, EQ_i is identical in both protocols. Since $m'_l(x_{R_l}, R_l^+) = m_l(x_{T_l}, T_l^+)$, EQ_{T_l} remains unchanged, so it is also identical in both protocols.
- $m'_l(x_{R_l}, R_l^+) \neq m_l(x_{T_l}, T_l^+)$: Observe that these two functions can be different only if the n inputs are not all identical. So it is correct to set $EQ_{T_l} = 1$.

□

In Fig.1(b), the graph for an equivalent protocol obtained by flipping the link corresponding to the 5-th step of the 4-node example in Fig.1(a) is presented.

Let us denote all the symbols a node i receives from and sends to the other nodes throughout the execution of protocol P as i^+ and i^-, respectively. It is obvious that i^- can be written as a function $M_i(x_i, i^+)$, which is the union of $m_l(x_i, i^+(l))$ over all steps l in which node i is the transmitter. If a protocol P satisfies $M_i(x_i, i^+) = M_i(x_i)$ for all i, we say P is **individual-input-determined (iid)**. The following theorem shows that there is always an iid equivalent for every protocol.

Theorem 1. *For every static protocol P for MEQ-AD, there always exists an iid equivalent static protocol P^*, which corresponds to an acyclic graph.*

Proof. According to Lemma 1, we can flip the direction of any edge in $E(P)$ and obtain a new protocol which is equivalent to P. It is to be noted that we can keep flipping different edges in the graph, which implies that we can flip any subset of $E(P)$ and obtain a new protocol equivalent to P.

In particular, we consider a protocol equivalent to P, whose corresponding graph is acyclic, and for all $(i, j) \in E(P)$, the property $i < j$ is satisfied. In this protocol, every node i has no incoming links from any node with index greater than i. This implies that the messages transmitted by node i are independent of the inputs to nodes with larger indices. Thus we can re-order the transmissions of this protocol such that node 1 transmits on all of its out-going links first, then node 2 transmits on all of its out-going links, ..., node $n-1$ transmits to n at the end. Name the new protocol Q. Obviously Q is equivalent to P.

Since we can always find a protocol Q equivalent to P as described above, all we need to do now is to find P^*. If Q itself is iid, then $P^* = Q$ and we are

done. If not, we obtain P^* in the following way (using function M'), which is similar to how we obtain the equivalent protocol P' in Lemma 1:

- For node 1, since it receives nothing from the other nodes, $M_1(x_1, 1^+) = M_1(x_1)$ is trivially true.
- For node $1 < i < n$, we modify Q as follows: node i computes its out-going message as a function $M_i'(x_i) = M_i(x_i, i^+|_{x_1=\cdots=x_n=x_i})$, where $i^+|_{x_1=\cdots=x_n=x_i}$ are incoming messages node i expects to receive, assuming that all parties have the same input x_i. At the end of the protocol, node i checks if $i^+|_{x_1=\cdots=x_n=x_i}$ equals to the actual incoming symbols i^+. If they match, nothing is changed. If they do not match, the inputs cannot be identical, and node i can set $EQ_i = 1$. (The correctness of this step may be easier to see by induction: apply this modification one node at a time, starting from node 1 to node $n-1$.) □

Theorem 1 shows that, to find the least cost of static protocols, it is sufficient to investigate only the static protocols that are iid and the corresponding communication graph is acyclic. From now on, such protocols are called iid static protocols for MEQ-AD. Fig.1(c) shows an iid static protocol that is equivalent to the one shown in Fig.1(a).

NOTE: The above technique of inverting the direction of transmissions can also be applied to general non-static MEQ-AD protocols. So Theorem 1 can be extended to cover all general protocols that solve the MEQ-AD problem. This immediately implies that among all MEQ-AD protocols (static and not static), there always exist an optimal protocol that is static. So **for the MEQ-AD problem, it is sufficient to only consider static protocols**.

6 MEQ-AD(3,6)

Let us first consider MEQ-AD(3,6), i.e., the case where 3 nodes (say A, B and C) are trying to solve the MEQ-AD problem when each node is given input from one out of six values, namely $\{1, 2, 3, 4, 5, 6\}$. According to Theorem 1, for any protocol that solves this MEQ-AD problem, there exists an equivalent iid partially ordered protocol in which node A has no incoming link, node B only transmits to node C, and node C has no out-going link. We construct one such protocol that solves MEQ-AD(3,6) and requires only 3 channel symbols, namely $\{1, 2, 3\}$, per link. Denote the channel symbol being sent over link ij as s_{ij}, the schedule of the proposed protocol is: (1) Node A sends $s_{AB}(x_A)$ to node B; (2) Node A sends $s_{AC}(x_A)$ to node C; and (3) Node B sends $s_{BC}(x_B)$ to node C. Table 1 shows how s_{ij} is computed as a function of x_i.

Now consider the outputs. Node A simply sets $EQ_A = 0$. For nodes B and C, they just compare the channel symbol received from each incoming link with the *expected* symbol computed with its own input value, and detect a mismatch if the received and expected symbols are not identical. For example, node B receives $s_{AB}(x_A)$ from node A. Then it detects a mismatch if the $s_{AB}(x_A) \neq s_{AB}(x_B)$.

x	1	2	3	4	5	6
s_{AB}	1	1	2	2	3	3
s_{AC}	1	2	2	3	3	1
s_{BC}	1	2	3	1	2	3

Table 1: A protocol for MEQ-AD(3,6)

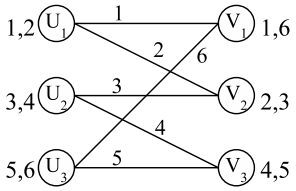


Fig. 2: The bipartite graph corresponding to the MEQ-AD(3,6) protocol in Table 1.

It can be easily verified that if the three input values are not all identical, at least one of nodes B and C will detect a mismatch. Hence the MEQ-AD(3,6) problem is solved with the proposed protocol. The communication cost of this protocol is

$$3\log_2 3 = \log_2 27 \approx 0.92 \times 2\log_2 6. \tag{6}$$

Notice that in this case, the upper bound from Section 4 equals to $(3-1)\log_2 6 = 2\log_2 6$. So we have found a static MEQ-AD protocol whose communication cost is lower than the upper bound. In fact, this protocol is optimal in the sense that it can be shown to achieve the minimum communication cost among all static protocols We prove the optimality of this protocol using an edge coloring argument.

6.1 Edge Coloring Representation of MEQ-AD(3,K)

From Sections 5, we have shown that it is sufficient to study 3-node systems where messages are transmitted only on links AB, AC and BC. Let us denote $|s_{AB}|$, $|s_{AC}|$ and $|s_{BC}|$ as the number of different symbols being transmitted on links AB, AC and BC, respectively.

Theorem 2. *The existence of a MEQ-AD(3,K) static protocol P with cost $C(P)$ is equivalent to the existence of a simple bipartite graph $G(U,V,E)$ and a distance-2 edge coloring scheme W, such that $|U|\times|V|\times|W| = 2^{C(P)}$, given $|E| = K$, $|U|\times|V| \geq K$, $|U|\times|W| \geq K$ and $|V|\times|W| \geq K$. Here U and V are disjoint sets of vertices, E is the set of edges, $|U| = |s_{AB}|$ and $|V| = |s_{AC}|$ are the sizes of sets U and V, and $|W| = |s_{BC}|$ is the number of colors used in W.*

The detailed proof can be found in our technical report [10]. According to Theorem 2, we can conclude that the problem of finding a least cost static protocol for $MEQ-AD(3,K)$ is equivalent to the problem of finding the minimum of $|U|\times|V|\times|W|$ for the bipartite graphs and distance-2 coloring schemes that satisfy the above constraints.

Using Theorem 2, to show that $C_{AD}(3,6) = \log_2 27$, we only need to show that for every combination of $|U|\times|V|\times|W| < 27$, there exists no bipartite graph $G(U,V,E)$ and a distance-2 coloring scheme W that satisfy the conditions as described in Theorem 2. It is not hard to see that there are only two combinations (up to permutation) that satisfy all conditions and have product

less than 27: (2, 3, 3) and (2, 3, 4). Notice that in both cases, $|E| = |U| \times |V|$, where every pair of edges is within distance 2 of each other, which means that the corresponding graph $G(U, V, E)$ can only be distance-2 edge colored with at least $|E| = 6 > 4 > 3$ colors. So neither (2, 3, 3) nor (2, 3, 4) satisfies the aforementioned conditions. Hence, together with the protocol presented before, we can conclude that $C_{AD}(3, 6) = \log_2 27$. The bipartite graph corresponding to Table 1 is illustrated in Fig. 2. Near the nodes U_i (or V_i) we show the set of value x's such that $s_{AB}(x) = i$ (or $s_{AC}(x) = i$). The number near each edges is the input value corresponding to that edge.

7 MEQ-AD(3,2^k)

Now we construct a protocol when the number of possible input values $K = 2^k, k \geq 1$ and only binary symbols can be transmitted in each step, using the MEQ-AD(3,6) protocol we just introduced in the previous sections as a building block.

First, we map the 2^k input values into 2^k different vectors in the vector space $\{1, 2, 3, 4, 5, 6\}^h$, where $h = \lceil \log_6 2^k \rceil = \lceil k \log_6 2 \rceil$. Then h instances of the MEQ-AD(3,6) protocol are performed in parallel to compare the h dimensions of the vector. Since 3 channels symbols are required for each instance of the MEQ-AD(3,6) protocol, we need to transmit a vector from $\{1, 2, 3\}^h$ on each of the links AB, AC and BC. One way to do so is to encode the 3^h possible vectors from $\{1, 2, 3\}^h$ into $b = \lceil \log_2 3^h \rceil = \lceil h \log_2 3 \rceil$ bits, and transmit the b bits through the links. Since the h instances of MEQ-AD(3,6) protocols solve the MEQ-AD(3,6) problem for each dimension, altogether they solve the MEQ-AD(3,2^k) problem. The communication cost this protocol can be computed as [10]

$$C(P) = 3\lceil h \log_2 3 \rceil < (0.92 \times 2k) + 7.755. \tag{7}$$

From Eq.7, we can see that when k is large enough, the communication cost of this protocol is upper bounded by 0.92 times of the upper bound $2 \log_2 2^k = 2k$ from Section 4. The way in which the above protocol is constructed can be generalized to obtain a MEQ-AD(3,K) protocol P with similar cost

$$C(P) < (0.92 \times 2 \log_2 K) + \Delta \tag{8}$$

for arbitrary value of K, where Δ is some positive constant.

8 About MEQ-CD

In this section, we will show that $C_{CD}(n, K)$ *roughly* equals to $C_{AD}(n, K)$:

$$C_{AD}(n, K) \leq C_{CD}(n, K) \leq C_{AD}(n, K) + n - 1. \tag{9}$$

We have shown the first inequality in Section 3.3. The second inequality can be proved by the following simple construction: Consider any protocol P for MEQ-AD, construct a protocol P' by having node i send EQ_i to node n by the end

of P, for all $i < n$. Node n collects the $n-1$ decisions from all other nodes and computes the final decision

$$EQ'_n = \max\{EQ_1, \cdots, EQ_n\}. \tag{10}$$

It is easy to see that, $EQ'_n = MEQ(x_1, \cdots, x_n)$. So $P' \in \Gamma_{CD}(n, K)$. Since $C(P') = C(P) + n - 1$, the second inequality is proved. From Eq.8 it then follows that for large enough K, the MEQ-CD(3,K) problem can also be solved with communication strictly less than $2\log_2 K$ bits. The performance can be improved somewhat by exploiting communication that may be already taking place between node n and the other nodes. For example, to solve MEQ-CD(3,6), instead of having nodes A and B sending 1 extra bit to node C at the end of the MEQ-AD(3,6) protocol in Section 6, we only need to add one possible value to s_{BC}, namely $s_{BC} \in \{1, 2, 3, 4\}$, where $s_{BC} = 4$ means that node B has detected a mismatch. The cost of this protocol is $2\log_2 3 + \log_2 4 = 2\log_2 3 + 2 < 3\log_2 3 + 2$. The same approach can also be applied to the MEQ-AD(3,2^k) protocol from Section 7 by making $|s_{BC}| = \lceil \log_2(3^h + 1) \rceil$, and obtain an MEQ-CD(3,2^k) protocol with cost of $2\lceil h \log_2 3 \rceil + \lceil \log_2(3^h + 1) \rceil$ bits, which is almost the same as $3\lceil h \log_2 3 \rceil$ for large h.

9 MEQ Problem with Larger n

Our construction in sections 6 and 7 can be generalized to networks of larger sizes. For brevity, just consider the case when $n = 3^m$. The nodes are organized in $m-1$ layers of "triangle"s. At the bottom ($(m-1)$-th) layer, there are 3^{m-1} triangles, each of which is formed with 3 nodes running the MEQ-AD(3,K) protocol presented in section 7. Then the i-th layer ($i < m-1$) consists 3^i triangles, each of which is formed with 3 "smaller" triangles from the $(i+1)$-th layer running the MEQ-AD(3,K) protocol. So the top layer consists of one triangle. For $K = 2^k$, the cost of this protocol is approximately

$$\frac{n-1}{2}(0.92 \times 2k + 7.755) \approx 0.92(n-1)k, \tag{11}$$

for large k. Notice that $(n-1)k$ is the upper bound from Section 4. So the improvement of a constant factor of 0.92 can also be achieved for larger networks.

10 Conclusion

In this paper, we study the communication complexity problem of the multiparty equality function, under the point-to-point communication model. The point-to-point communication model changes the problem significantly compared with previously used broadcast communication models. We focus on static protocols in which the schedule of transmissions is independent of the inputs. We then introduce techniques to significantly reduce the space of protocols to be studied.

We then study the MEQ-AD(3,6) problem and introduce an optimal static protocol that achieves the minimum communication cost among all static protocols that solve the problem. This protocol is then used as building blocks for construction of efficient protocols for more general MEQ-AD problems. The problem of finding the communication complexity of the MEQ problem for arbitrary values of n and K is still open.

Acknowledgments We thank the referees for their insightful comments and asking interesting questions. In particular, the referees pointed out that the MEQ problem is related to the set disjointness problem, and also asked the question of generalizing our results to networks larger than 3 nodes. Sections 2 and 9 incorporate these comments.

References

1. Noga Alon, Yossi Matias, and Mario Szegedy. The space complexity of approximating the frequency moments. In *STOC*, 1996.
2. Ziv Bar-yossef, T. S. Jayram, Ravi Kumar, and D. Sivakumar. An information statistics approach to data stream and communication complexity. In *IEEE FOCS*, pages 209–218, 2002.
3. Amit Chakrabarti, Subhash Khot, and Xiaodong Sun. Near-optimal lower bounds on the multi-party communication complexity of set disjointness. In *IEEE CCC*, 2003.
4. Ashok K. Chandra, I Merrick, L. Furst, and Richard J. Lipton. Multi-party protocols. In *In Proceedings of the Fifteenth Annual ACM Symposium on Theory of Computing*, pages 94–99, 1983.
5. Yefim Dinitz, Shlomo Moran, and Sergio Rajsbaum. Exact communication costs for consensus and leader in a tree. *J. of Discrete Algorithms*, 1:167–183, April 2003.
6. Eyal Kushilevitz and Noam Nisan. *Communication Complexity*. Cambridge University Press, 2006.
7. Eyal Kushilevitz and Enav Weinreb. The communication complexity of set-disjointness with small sets and 0-1 intersection. In *IEEE FOCS*, 2009.
8. Leslie Lamport, Robert Shostak, and Marshall Pease. The byzantine generals problem. *ACM Trans. on Programming Languages and Systems*, 1982.
9. Guanfeng Liang and Nitin Vaidya. Complexity of multi-valued byzantine agreement. *Technical Report, CSL, UIUC (http://arxiv.org/abs/1006.2422)*, June 2010.
10. Guanfeng Liang and Nitin Vaidya. Multiparty equality function computation in networks with point-to-point links. *Technical Report, CSL, UIUC*, 2010.
11. Mihai Pătraşcu and Ryan Williams. On the possibility of faster sat algorithms. In *Proceedings of the Twenty-First Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '10, pages 1065–1075, Philadelphia, PA, USA, 2010. Society for Industrial and Applied Mathematics.
12. Andrew Chi-Chih Yao. Some complexity questions related to distributive computing(preliminary report). In *STOC '79: Proceedings of the eleventh annual ACM symposium on Theory of computing*, pages 209–213, New York, NY, USA, 1979. ACM.