Multiparty Equality and Byzantine Broadcast using Random Linear Codes in Point-to-Point Networks *

Guanfeng Liang and Nitin Vaidya
Department of Electrical and Computer Engineering, and
Coordinated Science Laboratory
University of Illinois at Urbana-Champaign
gliang2@illinois.edu, nhv@illinois.edu

June 5, 2011

*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.

1 Introduction

In this paper, we present a competitive algorithm for multiparty equality computation in a point-to-point network. We also use this algorithm to obtain a competitive algorithm for Byzantine broadcast. The design of Byzantine broadcast is facilitated by the use of a simple random coding strategy in solving the equality problem.

Network model: We consider a point-to-point network, modeled by a simple graph $G(V, E)$. $V = \{1, 2, \cdots, n\}$ is the set of n nodes. The system is assumed to be synchronous. The n nodes are connected by a set of directed links E, with $l_{i,j}$ denoting the link from node i to node j. Each link $l_{i,j}$ has capacity $d_{i,j}$. $t\,d_{i,j}$ bits may be reliably transmitted on link $l_{i,j}$ in duration t (using a suitable time unit). We assume that $d_{i,j}$ is integer for all i, j.

2 Multiparty Equality (MEQ)

The objective of the multiparty equality problem is to determine whether all the n inputs are identical. We consider the following version of this problem, which we will refer to as MEQ hereafter.

- Each node i is given an input x_i of length L bits.
- Each node sets an outout bit to be 0 or 1. The output bit is initialized at each node as $\perp$.
- If all the inputs are equal, then the 1-bit output of all the nodes is 0. Otherwise, at least one node produces output equal to 1.

The execution time of a protocol for solving MEQ is the duration of time required from the initiation of the protocol until all the nodes have set their output to 0 or 1. For a given protocol, the execution time may depend on the input values at the different nodes. For a MEQ protocol P for a graph G, define $t(G, L, P)$ as its execution time, in the worst case, to solve the MEQ problem with L-bit inputs. Throughput of protocol P is then defined as

$$\frac{L}{t(G, L, P)}$$

Capacity of the MEQ problem is defined as follows, where the supremum is taken over all protocols that solve the MEQ problem for L bit inputs.

$$C_{MEQ}(G) \triangleq \sup_{P \text{ solves } MEQ} \lim_{L \to \infty} \frac{L}{t(G, L, P)}. \tag{1}$$

In this report, we present an algorithm that solves the multiparty equality (MEQ) problem [5] in point-to-point networks with throughput at least 1/2 the capacity. Our approach is based on a simple random linear coding strategy. In particular, in our randimg coding strategy, each node (party) computes some random linear functions of its own local input (interpreted as symbols from a suitable field), and transmits the values of the random functions to the other nodes on its outgoing links. Each node then computes its output using the values received from the other nodes, and its own input.

3 Preliminaries

3.1 Some MinCut Results

In our work, we consider a class of protocols wherein the number of bits transmitted on a certain link in each "round" or step of the algorithm is constant (depends only on L), independent of the actual input values provided to the nodes. In our recent work [5], we have shown that, given a static protocol P for the MEQ problem that transmits α bits on a certain directed link $l_{i,j}$, and β bits on a directed link $l_{j,i}$, it is possible to design a static MEQ protocol that transmits $\alpha + \beta$ bits on link $l_{i,j}$ and 0 bits on link $l_{j,i}$, and vice-versa. In other words, it is possible to "invert" the direction of communication by a certain static MEQ protocol, and still obtain another correct static MEQ protocol.[1] This, in turn, implies that it is sufficient to ignore the directions of the links in the network, and consider a undirected version of the point-to-point network. The undirected version of graph G replaces directed links $l_{i,j}$ and $l_{j,i}$ by a single undirected link with capacity $d_{i,j} + d_{j,i}$. The undirected cuts below correspond to the undirected graph thus obtained starting from G.

Let W be the minimum of the undirected cuts of the point-to-point network G (that is mincut in the undirected version of graph G). Thus,

$$W = MINCUT_{undirected}(G) \triangleq \min_{s \subset V, s \neq \phi} CUT_{undirected}(s, V - s). \tag{2}$$

It can be shown that the capacity of the MEQ problem in directed graph network G is upper bounded by W, i.e.,

$$C_{MEQ}(G) \leq W. \tag{3}$$

The appendix sketches the proof of the above upper bound.

For throughput and capacity, we will use the units of *symbols/unit time*, where symbols size and time unit are defined suitably. Capacity of each link (in units of symbols/unit time) is assumed to be an integer.

In our discussion below, we will find it convenient to interpret the point-to-point network as a **directed multigraph** with unit capacity edges. Thus, the communication link of capacity $d_{i,j}$ from node i to node j will now be modeled using $d_{i,j}$ unit capacity directed edges from node i to node j. Similarly, by ignoring the directions in this multigraph, we obtain an **undirected multigraph** representation of the network. The mincut in this undirected multigraph will be W as defined above. It is known [4] that in such an undirected multigraph with mincut W, there exist at least $W/2 \leq R \leq W$ edge disjoint undirected unit-capacity spanning trees (recall that the edges in the multigraph are all unit capacity edges). In Section 4, we present a random linear code-based protocol that solves the MEQ problem at rate R symbols/time unit (where the symbol size and time unit are chosen suitably).

3.2 Example

We now illustrate the four representations of the point-to-point network, listed below, which we use in our discussion:

- G: Simple directed graph

[1] In fact, similar inversion can be performed in a more general class of MEQ protocols as well.

- G^*: Directed multigraph with unit capacity edges
- $\overline{G}$: Simple undirected graph
- $\overline{G^*}$: Undirected multigraph with unit capacity edges

Figure 1(a) shows a directed point-to-point network consisting of four nodes. The numbers near the various links are the link capacities, in suitable units. We assume that the link capacities are integers. Figure 1(b) shows an undirected graph representation of the network in figure 1(a). Note that the capacity of the undirected link between nodes A and B is the sum of the capacities of the links AB and BA in figure 1(a). Figures 1(c) and 1(d) show directed multigraph, and undirected multigraph representation, respectively. In our multigraph representations, each edge has unit capacity. For instance, there are three edges between nodes A and B in the undirected multigraph (figure 1(d)), since the undirected edge AB in figure 1(b) has capacity 3. Figure 1(e) shows an unit-capacity edge disjoint spanning tree in the directed multigraph, with the root of the tree being node A. Figure 1(f) shows two unit-capacity edge disjoint spanning trees for the undirected multigraph. The edges in each spanning tree are drawn differently.

$MINCUT_{undirected}(G)$, which is equal to $MINCUT(\overline{G})$ (Figure 1(b)) is 2 (between node C and nodes {A,B,D}), whereas $MINCUT_{directed}(G)$ (Figure 1(a)) is 0 (from node C to nodes {A,B,D}).

3.3 Invertibility of a Matrix with a Random Diagonal

Let A be an arbitrary fixed m-by-m matrix. Consider a random m-by-m diagonal matrix C with m diagonal elements $x_1, \cdots, x_m$.

$$C = \begin{pmatrix} x_1 & 0 & 0 & \cdots & 0 \\ 0 & x_2 & 0 & \cdots & 0 \\ 0 & 0 & x_3 & \cdots & 0 \\ \vdots & & & \ddots & \vdots \\ 0 & \cdots & \cdots & 0 & x_m \end{pmatrix} \tag{4}$$

For some integer t, $1 \leq t \leq m$, the diagonal elements of C are partitioned into t sets $s_i = \{x_{s_{i,1}}, \cdots, x_{s_{i,|s_i|}}\}$, $i = 1, \cdots t$, such that $x_{s_{i,1}} = \cdots = x_{s_{i,|s_i|}} = X_i$ for all $i \leq t$. X_i's are random variables. $X_1, \cdots, X_t$ are selected independently and uniformly randomly from $GF(2^p)$. Then we have:

Theorem 1 *The probability that the m-by-m matrix $C - A$ is invertible is lower bounded by:*

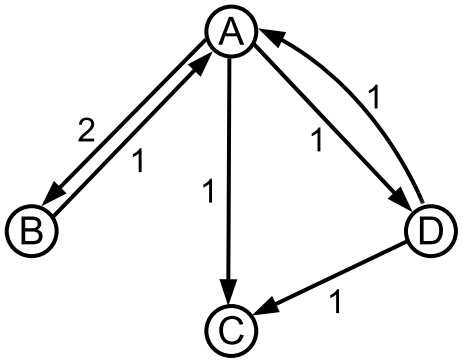
$$\Pr\{(C - A) \textit{ is invertible}\} \geq 1 - \frac{m}{2^p}. \tag{5}$$

Proof: Consider the determinant of matrix $C - A$.

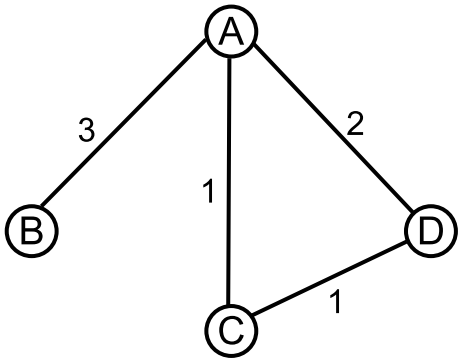
$$\det(C - A) = \det \begin{pmatrix} (x_1 - a_{1,1}) & -a_{1,2} & \cdots & -a_{1,m} \\ -a_{2,1} & (x_2 - a_{2,2}) & \cdots & -a_{2,m} \\ \vdots & & \ddots & \vdots \\ -a_{m,1} & \cdots & -a_{m,m-1} & (x_m - a_{m,m}) \end{pmatrix} \tag{6}$$

$$= (x_1 - a_{1,1})(x_2 - a_{2,2}) \cdots (x_m - a_{m,m}) + \text{other terms} \tag{7}$$

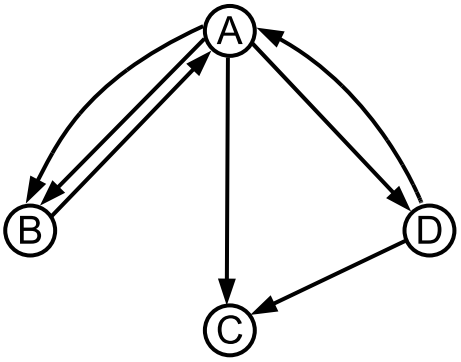
$$= \Pi_{i=1}^{t} X_i^{|s_i|} + Q_{m-} \tag{8}$$



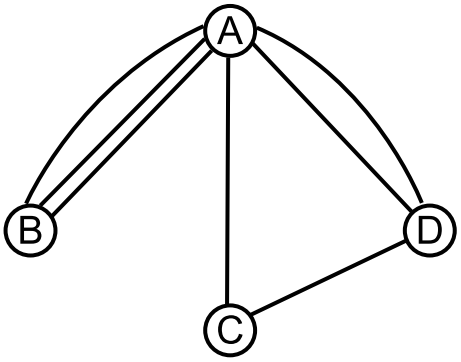
(a) Directed Simple Graph G



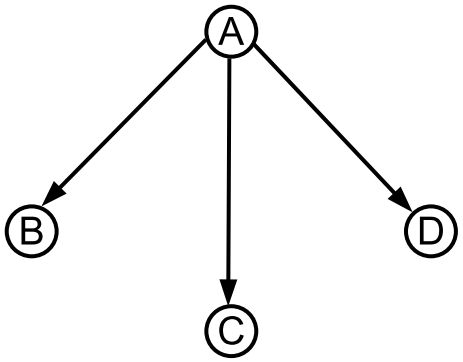
(b) Undirected Simple Graph $\overline{G}$



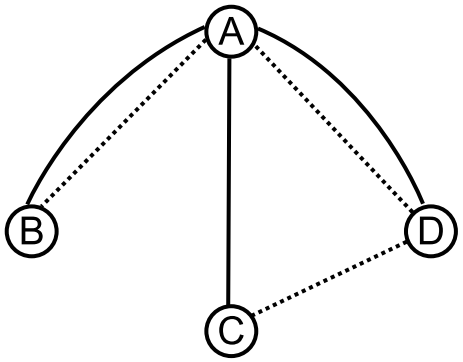
(c) Directed Multigraph G^*, every directed link has capacity 1



(d) Undirected Multigraph $\overline{G^*}$, every undirected link has capacity 1



(e) Directed spanning tree in G^* rooted at node A



(f) Undirected spanning trees in $\overline{G^*}$

Figure 1: Graph representations of the network

The last step above is obtained by replacing x_i's in $C - A$ by X_j's as appropriate. The first term above, $\Pi_{i=1}^{t} X_i^{|s_i|}$, is a degree-$m$ polynomial of $X_1, \cdots, X_t$. Q_{m-} is a polynomial of degree at most $m-1$ and it represents the remaining terms in $\det(C-A)$. $\det(C-A)$ cannot be identically zero since it contains only one degree m term. Then by the Schwartz-Zippel Theorem, the probability that $\det(C-A) = 0$ is $\leq m/2^p$. Since $C - A$ is invertible if and only if $\det(C-A) \neq 0$, we conclude that

$$\Pr\{(C-A) \text{ is invertible}\} \geq 1 - \frac{m}{2^p} \tag{9}$$

□

4 MEQ Protocol with Random Linear Code

In this discussion, we consider the undirected multigraph representation of the network, with each edge being a unit-capacity edge. For an undirected multigraph $\overline{G^*}$ with (undirected) mincut W, as noted previously, $R \geq W/2$ undirected unit-capacity edge disjoint spanning trees exist. To each unit-capacity edge in each undirected spanning tree, we now assign a direction according to the corresponding link directions in the underlying directed network G. We want to point out that, after the directions are assigned to the links in a undirected spanning tree, it may not necessarily become a directed spanning tree. These directed unit capacity edges will be used in the algorithm below. Note that between each pair of nodes there may be multiple such unit capacity edges.

We represent input x_i at each node i as a column vector of R symbols from $GF(2^p)$: $[x_i(1), \cdots, x_i(R)]^T$. So the size of each local input is pR bits. Every node i computes coded symbols as random linear combinations of the R local input symbols. For example, a coded symbol can be written as

$$y = c(1)x_i(1) + c(2)x_i(2) + \cdots + c(R)x_i(R) = c^T x_i. \tag{10}$$

where c above represents the coefficient vector. The coefficients $c(1), \cdots, c(R)$ are chosen randomly from $GF(2^p)$. On each of its outgoing unit capacity edges (obtained from the spanning trees), node i sends 1 randomly coded symbol obtained as a linear function of its R-symbol input x_i. The coefficients, although randomly chosen, can be viewed as a part of the algorithm specification, and assumed to be known to the nodes. Upon receiving a coded symbol y, node i just checks if $y = c^T x_i$. If any one of the received coded symbol fails this check, node i sets its output bit to 1 (since a mismatch of inputs has been detected), otherwise the output is set to 0. In other words, on receiving a coded symbol from node j, node i will not detect a mismatch if

$$c(1)x_i(1) + \cdots + c(R)x_i(R) = c(1)x_j(1) + \cdots + c(R)x_j(R), \tag{11}$$

which is equivalent to

$$c(1)(x_i(1) - x_j(1)) + \cdots + c(R)(x_i(R) - x_j(R)) = 0. \tag{12}$$

4.1 Representing a spanning tree with a $(n-1)$-by-$(n-1)$ matrix

The coded symbols are transmitted according to the R unit-capacity spanning trees. Let us label the trees as $T_1, \cdots, T_k, \cdots, T_R$. Each tree T_k can be represented by a $(n-1)$-by-$(n-1)$ matrix A_k that has the following structure:

- Each row corresponds to exactly one edge in tree T_k.

- If directed edge $(1, i)$ $(i > 1)$ is in tree T_k, then there is a row in A_k in which the $(i-1)$-th element is 1, and the rest of the entries are 0.
- If directed edge $(i, 1)$ $(i > 1)$ is in tree T_k, then there is a row in A_k in which the $(i-1)$-th element is -1, and the rest of the entries are 0.
- If directed edge (i, j) where $1 < i < j$, is in tree T_k, then there is a row in A_k in which the $(i-1)$-th entry is -1, the $(j-1)$-th entry is 1, and the rest of the entries are 0.
- If directed edge (i, j) where $1 < j < i$, is in tree T_k, then there is a row in A_k in which the $(i-1)$-th entry is 1, the $(j-1)$-th entry is -1, and the rest of the entries are 0.

Observe that, for edges incident on nodes 1, the corresponding rows have exactly one non-zero entry. Also, the row corresponding to an edge that is incident on node i has a non-zero entry in column $i-1$. Example of A_k with $T_k = \{(1,2),(2,3),(2,5),(5,4)\}$ is as follows:

$$A_k = \begin{pmatrix} 1 & 0 & 0 & 0 \\ -1 & 1 & 0 & 0 \\ -1 & 0 & 0 & 1 \\ 0 & 0 & 1 & -1 \end{pmatrix} \tag{13}$$

Since there must be at least one edge in T_k that is incident on node 1, there must be at least one row of A_k that has only one non-zero element. Also, since every node is covered by at least one edge in T_k, every column of A_k has at least one non-zero element(s). Since there is at most one edge between every pair of nodes in T_k, no two rows are non-zero in identical columns. Therefore, by row manipulation, we can transform matrix A_k into another matrix in which every row and every column has exactly one non-zero element. It then follows that A_k must be invertible.

4.1.1 Matrix representation of the MEQ solution

For any positive integer $r \leq R$, let us denote the difference in the r-th input symbols at x_1 and x_i as $e(i, r)$, defined as $e(i, r) = x_1(r) - x_i(r)$. We define a "difference-vector" of the n inputs, corresponding to the r-th input symbol, as

$$e_r = \begin{pmatrix} e(2,r) \\ e(3,r) \\ \vdots \\ e(n,r) \end{pmatrix}. \tag{14}$$

Note that the differences are defined with respect to the input at node 1. Let $c_{k,r,l}$ be the r-th coefficient used to compute the coded symbol transmitted on the edge corresponding to the l-th row of A_k. That is, if this transmission occurs from node i, then the symbol transmitted is computed as

$$c_{k,1,l}x_i(1) + c_{k,2,l}x_i(2) + \cdots + c_{k,R,l}x_i(R)$$

Define $C_{k,r} = diag(c_{k,r,1}, \cdots, c_{k,r,n-1})$ be the diagonal matrix whose diagonal is $[c_{k,r,1}, \cdots, c_{k,r,n-1}]$. Then the $(n-1)$ equalities in the form of Equation 12 that hold for the $(n-1)$ symbols transmitted on the $(n-1)$ edges in T_k can be represented using the equation below.

$$C_{k,1}A_ke_1 + C_{k,2}A_ke_2 + \cdots + C_{k,R}A_ke_R = 0. \tag{15}$$

Define $(n-1)k \times (n-1)k$ matrix M_k as

$$M_k = \begin{pmatrix} C_{1,1}A_1 & C_{1,2}A_1 & \cdots & C_{1,k}A_1 \\ C_{2,1}A_2 & C_{2,2}A_2 & \cdots & C_{2,k}A_2 \\ \vdots & & & \\ C_{k,1}A_k & C_{k,2}A_k & \cdots & C_{k,k}A_k \end{pmatrix} \tag{16}$$

Then instances of Equation 15 corresponding to all the R spanning trees can be collectively represented as follows.

$$M_R \times \begin{pmatrix} e_1 \\ e_2 \\ \vdots \\ e_R \end{pmatrix} = 0. \tag{17}$$

When M_R is invertible (non-singular), for the above equality to hold, $e_1, \cdots, e_R$ must all equal 0, which is equivalent to $x_1 = \cdots = x_n$. So the random linear coding protocol solves the MEQ problem at rate R correctly if M_R is invertible. So all we need to show is that when the diagonal elements of each matrix $C_{k,r}$ are chosen at random, the matrix M_R is invertible with high probability.

Theorem 2 *Matrix M_R is invertible with probability at least*

$$1 - \frac{(n-1)R}{2^p}. \tag{18}$$

Proof: We prove that M_k is invertible with high probability for $1 \leq k \leq R$. The proof is by induction, with $k = 1$ being the base case.

Base Case: $k = 1$

For $k = 1$,

$$M_1 = C_{1,1}A_1 \tag{19}$$

A_1 is a $(n-1) \times (n-1)$ invertible matrix, and thus has rank $(n-1)$. Since $C_{1,1}$ is a $(n-1) \times (n-1)$ diagonal matrix, its rank is also $(n-1)$ provided that all its $n-1$ diagonal elements are non-zero. Thus, the rank of M_1 is $(n-1)$ as well provided that all the diagonal elements of $C_{1,1}$ are non-zero. Since each of the diagonal elements is chosen uniformly randomly from $GF(2^p)$, this event occurs with probability

$$\left(1 - \frac{1}{2^p}\right)^{n-1} \geq 1 - \frac{n-1}{2^p}$$

Induction Step: k **to** $k+1$

Now assume that the $(n-1)k \times (n-1)k$ matrix M_k $(1 \leq k < R)$ is invertible with probability $\geq \left(1 - \frac{n-1}{2^p}\right)^k$. The $(n-1)(k+1) \times (n-1)(k+1)$ matrix M_{k+1} can be written as

$$M_{k+1} = \begin{pmatrix} M_k & D_k \\ F_k & C_{k+1,k+1}A_{k+1} \end{pmatrix}, \tag{20}$$

where

$$D_k = \begin{pmatrix} C_{1,k+1}A_1 \\ C_{2,k+1}A_2 \\ \vdots \\ C_{k,k+1}A_k \end{pmatrix} \tag{21}$$

is a $(n-1)k \times (n-1)$ matrix, and

$$F_k = \left(C_{k+1,1}A_{k+1}, C_{k+1,2}A_{k+1}, \cdots, C_{k+1,k}A_{k+1}\right) \tag{22}$$

is a $(n-1) \times (n-1)k$ matrix. $C_{k+1,k+1}A_{k+1}$ is a $(n-1) \times (n-1)$ matrix. Recall that elements of all these matrices are from $GF(2^p)$.

Matrix M'_{k+1} below has the same rank as M_{k+1} because the $n(k+1) \times n(k+1)$ matrix below formed using identity matrices $I_{(n-1)k}$ and $I_{(n-1)}$ has full rank.

$$\begin{aligned} M'_{k+1} &= \begin{pmatrix} I_{(n-1)k} & 0 \\ -F_k M_k^{-1} & I_{(n-1)} \end{pmatrix} M_{k+1} && (23) \\ &= \begin{pmatrix} I_{(n-1)k} & 0 \\ -F_k M_k^{-1} & I_{(n-1)} \end{pmatrix} \begin{pmatrix} M_k & D_k \\ F_k & C_{k+1,k+1}A_{k+1} \end{pmatrix} && (24) \\ &= \begin{pmatrix} M_k & D_k \\ 0 & C_{k+1,k+1}A_{k+1} - F_k M_k^{-1} D_k \end{pmatrix}. && (25) \end{aligned}$$

The matrix below has full rank because A_{k+1} is invertible.

$$\begin{pmatrix} I_{(n-1)k} & 0 \\ 0 & A_{k+1}^{-1} \end{pmatrix}$$

Then it follows that matrix M''_{k+1} below has the same rank as M'_{k+1}, and hence the same rank as M_{k+1}.

$$\begin{aligned} M''_{k+1} &= M'_{k+1} \begin{pmatrix} I_{(n-1)k} & 0 \\ 0 & A_{k+1}^{-1} \end{pmatrix} && (26) \\ &= \begin{pmatrix} M_k & D_k \\ 0 & C_{k+1,k+1}A_{k+1} - F_k M_k^{-1} D_k \end{pmatrix} \begin{pmatrix} I_{(n-1)k} & 0 \\ 0 & A_{k+1}^{-1} \end{pmatrix} && (27) \\ &= \begin{pmatrix} M_k & D_k A_{k+1}^{-1} \\ 0 & C_{k+1,k+1} - F_k M_k^{-1} D_k A_{k+1}^{-1} \end{pmatrix} && (28) \end{aligned}$$

Now we show that the $(n-1) \times (n-1)$ matrix $C_{k+1,k+1} - F_k M_k^{-1} D_k A_{k+1}^{-1}$ above is invertible with high probability. Note that the values of the diagonal elements of the diagonal matrix $C_{k+1,k+1}$ are chosen independent of $F_k M_k^{-1} D_k A_{k+1}^{-1}$. So by Theorem 1, we can conclude that the probability of $C_{k+1,k+1} - F_k M_k^{-1} D_k A_{k+1}^{-1}$ being invertible is lower bounded by $1 - (n-1)/2^p$.

Since M_k is invertible with probability $\geq \left(1 - \frac{n-1}{2^p}\right)^k$, and $C_{k+1,k+1} - F_k M_k^{-1} D_k A_{k+1}^{-1}$ is invertible with probability $\geq \left(1 - \frac{n-1}{2^p}\right)$, it follows that M''_{k+1} – and, therefore, M_{k+1} – is invertible with probability

$$\geq \left(1 - \frac{n-1}{2^p}\right)^k \left(1 - \frac{n-1}{2^p}\right) = \left(1 - \frac{n-1}{2^p}\right)^{k+1}$$

This implies that M_R is invertible with probability

$$\geq \left(1 - \frac{n-1}{2^p}\right)^R \geq \left(1 - \frac{(n-1)R}{2^p}\right)$$

For specific n and R, the above lower bound can be made arbitrarily close to 1 by picking a large enough value of p. □

4.1.2 Variation on the Random Coding Strategy

Consider a variation on our random coding strategy wherein, for a tree T_k, suppose that we choose some of the coding coefficients identically, instead of choosing them independently (e.g, $c_{k,r,l} = c_{k,r,m}$ for some l, m). In particular, with this modified strategy, the coefficients are chosen such that the diagonal elements of each matrix $C_{k,r}$ are partitioned such that the elements in each partition are identical, whereas the values in different partitions are chosen independently. The coefficient in matrix $C_{k,r}$ for different k, r are chosen independent of each other. As a special case of this strategy, for a certain k, r, we may have $c_{k,r,l} = c_{k,r,m} \quad \forall l, m$. It turns out that Theorem 1 holds even when the diagonal elements of matrix $C_{k,r}$ are partitioned as above. Therefore, even if $c_{k,r,l} = c_{k,r,m}$ for some (or all) l, m pairs, $C_{k,r}$ is still invertible with probability at least $1 - (n-1)/2^p$. Thus, Theorem 2 holds for the modified random coding strategy as well. This fact will be useful in our construction of a Byzantine broadcast algorithm below.

4.1.3 Sub-Optimality of the Random Linear Coding Protocol

As we mentioned in the beginning of this section, in a point-to-point network $G(V, E)$ with minimum undirected cut W, at least $R \geq W/2$ unit-capacity spanning trees can be found. So the random linear coding protocol we just described solves the MEQ problem at rate $R \geq W/2$ with high probability. Since the capacity of the MEQ problem $C_{MEQ} \leq W$, it follows that

$$R \geq C_{MEQ}/2. \tag{29}$$

It is not difficult to construct networks in which $R < C_{MEQ}$. This implies that, although the proposed scheme achieve throughput at least $\frac{1}{2}$ of the capacity, it does not achieve capacity in general.

5 Byzantine Broadcast with Random Linear Codes

Using the result from the previous section, we sketch a Byzantine broadcast algorithm using random linear codes, which achieves at least 1/2 of the capacity of Byzantine broadcast in point-to-point networks. For the broadcast, one of the nodes, say node 1, acts as the source, and wants to broadcasts a value to all the other nodes. At the completion of the algorithm, all the fault-free nodes in the network receive (or "agree on") an identical value. In addition, the agreed value is identical to the source node's value, provided that the source node itself is fault-free. Throughput and capacity of broadcast in point-to-point networks can be defined analogous to similar quantities for multicast and unicast. For more details, please refer [6].

The core of the broadcast algorithm designed using the above solution for equality is as follows:

- Step 1: Node 1 uses a traditional store-and-forward approach to broadcast a value, containing R symbols, to all the other nodes in the network.

  During the broadcast performed by node 1, it is possible that some nodes in the network may tamper the packets. The remaining algorithm is designed to detect such tampering, using our solution to equality.

- Step 2: In the network of n nodes, up to $t < n/3$ nodes may be faulty, possibly including the source node 1. After receiving the R-symbol value in step 1, each subset of $(n - t)$ nodes computes the MEQ function of the received R-symbol values. So there are n-choose-$(n - t)$ instances of MEQ to be performed.

  Due to the manner in which we use the random functions for performing MEQ, all the n-choose-$(n-t)$ instances of MEQ can be performed simultaneously. In particular, if k instances of MEQ use a certain unit capacity edge, it suffices to send one packet on that edge (that is, we do not need to send k packets on that edge). In fact, the packets sent during step 1 are also useful in performing the equality checks, and if a packet sent on a certain edge in step 1, additional packets needs not be sent on that edge in step 2.

- Step 3: Each node broadcasts a single bit to all the other nodes using a traditional Byzantine Broadcast algorithm (such as [9, 3, 2]). Bit broadcast by a node N is 0 if its output in **all** instances of MEQ that it belongs to in Step 2 is 0; else the bit is 1. In essence, at the end of step 3, all the nodes will learn if any of the nodes in the network has detected a mismatch of values received during step 1.

  If such a mismatch is announced, it implies that (i) some node(s), possibly including node 1, tampered packets during step 1, or (ii) there was no tampering during step 1, and yet a faulty node is announcing a mismatch during step 3. In either case, if a node announces a mismatch in step 3, indeed some node has misbehaved during steps 1 through 3.

- Step 4: If misbehavior is detected as above, then additional steps are performed to learn (partial) information regarding the identity of the misbehaving node.

The rest of the algorithm is similar to the "dispute control" [1] structure, which was also used in our our capacity-achieving Byzantine broadcast algorithm for 4-node point-to-point networks [6].

Now let us consider the throughput achieved by the above algorithm in comparison to the capacity of Byzantine broadcast. Denote $\{G_i : i = 1, \cdots, \binom{n}{n-t}\}$ the set of all $\binom{n}{n-t}$ subgraphs of G, each of which contains $n - t$ nodes and the links between them. For each G_i, define $\overline{W}_i = MINCUT_{undirected}(G_i)$. In Appendix B, we show that the capacity of Byzantine broadcast in a directed point-to-point network $G(V, E)$ with $t < n/3$ failures is upper bounded by

$$C_{BB} \leq \overline{W} \triangleq \min_{i=1,\cdots,\binom{n}{n-t}} \overline{W}_i. \tag{30}$$

Also, for each subgraph G_i, denote by $\overline{K}_i$ the maximum number of edge disjoint unit-capacity spanning trees in its undirected multigraph representation of $\overline{G_i^*}$. As we have discussed in Section 3.1, $\overline{W}_i/2 \leq \overline{K}_i \leq \overline{W}_i$. Now let

$$\overline{K} \triangleq \min_{i=1,\cdots,\binom{n}{n-t}} \overline{K}_i, \tag{31}$$

which is the maximum number of unit-capacity spanning trees that every subgraph G_i can have. Observer that for two arbitrary sequences of values $\{x_1, ..., x_N\}$ and $\{y_1, ..., y_N\}$ such that $x_i \leq y_i$ for all i, $\min\{x_i\} \leq \min\{y_i\}$ is always true. So given that $\overline{W}_i/2 \leq \overline{K}_i \leq \overline{W}_i$ for all G_i, we can conclude that $\overline{W}/2 \leq \overline{K} \leq \overline{W}$.

Let us now view G as a directed multigraph with unit capacity edges. Let $\overrightarrow{B}$ be the maximum number of directed edge-disjoint spanning trees in directed graph G rooted at the source node. Then $\overrightarrow{B}$ is the broadcast capacity in the absence of any failures. Clearly, $\overrightarrow{B}$ is an upper bound on the capacity of Byzantine broadcast (when failures may occur). As a result, we have

$$C_{BB} \leq \min(\overline{W}, \overrightarrow{B}). \tag{32}$$

Now select $R = \min(\overline{K}, \overrightarrow{B})$. From the facts that $C_{BB} \leq \min(\overline{W}, \overrightarrow{B})$ and $\overline{W}/2 \leq \overline{K} \leq \overline{W}$, it follows that

$$R \geq C_{BB}/2. \tag{33}$$

Our Byzantine broadcast algorithm achieves throughput $R = \min(\overline{K}, \overrightarrow{B})$. The algorithm uses the "dispute control" [1] structure, also used in our capacity-achieving 4-node Byzantine broadcast algorithm in [6]. The information to be broadcast by the source (node 1) is divided into generations of identical size, specifically R symbols. In each generation, source node 1 broadcasts the R packets (or symbols) to the rest of the network, along R directed unit-capacity edge disjoint spanning trees rooted at node 1 (similar to the example in Figure 1(f)). Since $R \leq \overrightarrow{B}$, R such spanning trees necessarily exist.

Source node 1 sets its input x_1 equal to the R packets it has just broadcast, and each node i ($\neq 1$) sets x_i equal to the R packets received during the above broadcast.

The MEQ operation is performed on these x_i values. In particular, each node i transmits randomly coded packets generated from x_i on those outgoing unit-capacity directed edges that were not used during the above broadcast. Each node i then checks whether all the received packets are consistent with x_i. It should not be difficult to see that doing this solves the MEQ problem among every subset of $n - t$ (fault-free) nodes in G, since Theorem 2 holds even when some of the diagonal elements of the random matrix M are identical. It then follows that either all fault-free nodes will have identical x_i's, or misbehavior by some faulty node is detected. After the misbehavior is detected, extra diagnostic operations are perform and then the algorithm proceeds to the next generation of data. These operations are similar to those in [6] and [8]. For brevity, we omit the details here.

5.1 Byzantine Consensus using Byzantine Broadcast

The Byzantine broadcast algorithm we describe above can be used to construct *Byzantine consensus* algorithm that achieves the same throughput. Throughput and capacity of consensus in point-to-point networks can be defined analogous to similar quantities for broadcast. For more details, please refer [7].

In our Byzantine consensus algorithm, the input information at each node is divided into generations, each of R symbols. An set $P_{match}(g)$ is maintained as the largest set of nodes that appear to have identical inputs up to the beginning of the g-th generation. Initially, $P_{match}(1) = V$, i.e., all nodes are in P_{match} when the algorithm starts. In every generation g, let N^* be the node in $P_{match}(g)$ with the smallest index. Then Steps 1 to 3 are performed with N^* being the source.

In addition to step 3, every node in $i \in P_{match}(g)$ ($\neq N^*$) compares the symbols broadcast by N^* in step 1 to its own input of the current generation, denoted as $x_i(g)$. Node i then broadcasts a single bit to all the other nodes using a traditional Byzantine Broadcast algorithm, indicating whether $x_i(g)$ equals to what broadcast by N^* or not. If some node $i \in P_{match}(g)$ claims that $x_i(g)$ is different from what N^* broadcast, or misbehavior is detected, Step 4 is performed. In particular, in order to update P_{match}, every node in $P_{match}(g)$ is required to broadcast their input of the g-th generation using a traditional Byzantine broadcast algorithm. Then we update $P_{match}(g+1)$ as the largest subset of $P_{match}(g)$ that broadcast the same input. If $|P_{match}(g+1)| \geq n - t$, continue to the next generation. Otherwise, $|P_{match}(g+1)| < n - t$, which implies that the inputs of fault-free nodes are not identical. Then the algorithm terminates with a default output.

References

[1] Zuzana Beerliova-Trubiniova and Martin Hirt. Efficient multi-party computation with dispute control. In *TCC*, 2006.

[2] Piotr Berman, Juan A. Garay, and Kenneth J. Perry. Bit optimal distributed consensus. *Computer science: research and applications*, 1992.

[3] Brian A. Coan and Jennifer L. Welch. Modular construction of a byzantine agreement protocol with optimal message bit complexity. *Inf. Comput.*, 97(1):61–85, 1992.

[4] Zongpeng Li and Baochun Li. Network coding in undirected networks. In *Conference on Information Sciences and Systems*, 2004.

[5] Guanfeng Liang and Nitin Vaidya. Multiparty equality function computation in networks with point-to-point links. In *SIROCCO*, 2010.

[6] Guanfeng Liang and Nitin Vaidya. Capacity of byzantine agreement with finite link capacity. In *INFOCOM 2011. 30th IEEE International Conference on Computer Communications.*, 2011.

[7] Guanfeng Liang and Nitin Vaidya. Capacity of byzantine consensus with capacity limited point-to-point links. *Technical Report, CSL, UIUC*, March 2011.

[8] Guanfeng Liang and Nitin Vaidya. Error-free multi-valued consensus with byzantine failures. In *ACM PODC*, 2011.

[9] M. Pease, R. Shostak, and L. Lamport. Reaching agreement in the presence of faults. *JOURNAL OF THE ACM*, 1980.

A Proof of Upper Bound on Capacity of MEQ

The upper bound is presented in Equation (3).

Suppose the cut between a certain set s and $V - s$ is the minimum undirected cut in graph G, i.e., $CUT_{undirected}(s, V - s) = W$. Now consider a constrained version of the MEQ problem wherein all nodes in s are have an identical input, denoted as x, and all nodes in $V - s$ have an identical input, denoted as y. If we ignore the cost of "internal" communication among the nodes in set s, and similarly, ignore the cost of internal communication among the nodes in $V - s$, then solving

the MEQ problem with such constrained distribution of inputs is equivalent to solving the 2-party equality problem: Alice and Bob each is given input x and y, respectively, need to check if $x = y$ by communicating with each other. It has been proved that at least L bits must be communicated between Alice and Bob, in the worst case, to solve 2-party equality for L-bit values. It then follows that even if the inputs at the nodes in G are constrained as described above, the execution time $t(G, L, P)$ must satisfy $W\, t(G, L, P) \geq L$ for any L and P. This implies that $C_{MEQ}(G) \leq W$.

B Proof of an Upper Bound on C_{BB}

The upper bound is presented in Equation (30).

Given any subgraph G_i of G with size $n - t$, let us rename the nodes in G_i as $g_1, g_2, \cdots, g_{n-t}$, and the nodes not in G_i as $f_1, f_2, \cdots, f_t$.

We first discuss the case when the source of the Byzantine broadcast problem is not in G_i. Without loss of generality, we can assume that f_1 is the source.

Given any algorithm, namely A, that solves Byzantine broadcast in network G, with node f_1 as the source, with at most t failures, and at some rate R, in the following, we construct a protocol P that solves MEQ in G_i at rate R as follows. For the MEQ problem, let us assume that x_j is the input value at node g_j. Thus, the goal is to determine whether x_j is identical at all $g_j \in G_i$.

1. Every node $g_j \in G_i$ creates a local virtual network as follows:
   (a) It creates one virtual node $g_{k,j}$ for each $g_k \in G_i$, $k \neq j$. Similarly, it creates one virtual node $f_{l,j}$ for each $f_l \notin G_i$. Node g_j also includes itself in the local virtual network.
   (b) Every pair of virtual nodes are connected with a pair of links of the same capacity as the ones that connects the corresponding pair of actual nodes in the original network G. In other words, link $(g_{k,j}, g_{l,j})$ has the same capacity as link (g_k, g_l). Similarly, link $(g_{k,j}, f_{l,j})$ has the same capacity as link (g_k, f_l).
   (c) Node g_j connects itself with each of the virtual nodes $f_{l,j}$ such that link $(g_j, f_{l,j})$ has the same capacity as link (g_j, f_l), and link $(f_{l,j}, g_j)$ has the same capacity as link (f_l, g_j).
   (d) Node g_j connects itself with each of the virtual nodes $g_{k,j}$ with one link $(g_j, g_{k,j})$ that has the same capacity as link (g_j, g_k). There is no link from virtual node $g_{k,j}$ to node g_j.
   (e) Every virtual node is assigned with the same code that the corresponding actual node should run in algorithm A. In other words, virtual node $f_{l,j}$ is assigned the execution code that node f_l should run in A. Virtual node $g_{k,j}$ is assigned the execution code that node g_k should run in algorithm A, except that it drops the messages that should be sent to node g_j (since there is no link from virtual node $g_{k,j}$ to node g_j).
   (f) Node g_j execute correctly as specified by algorithm A. When algorithm A specifies that g_j should send a message to an actual node $g_k \in G_i$, g_j sends the message to both the actual node g_k and the virtual node $g_{k,j}$. When it should send a message to node $f_l \notin G_i$ according to algorithm A, g_j sends the message only to the virtual node $f_{l,j}$. When it receives an message from virtual node $f_{l,j}$, it pretends that the message is received from the actual node f_l. Since there is no link from virtual node $g_{k,j}$ to node g_j, g_j will not receive messages from $g_{k,j}$.

2. Recall that node f_1 is the source nodes for the broadcast being performed by algorithm A. Thus, node f_1 has an input value that it will broadcast using algorithm A. Each node $g_j \in G_i$ sets the initial input at node $f_{1,j}$ equal to x_j (recall that x_j is the input for the MEQ operation at node g_j). Thus, algorithm A for node $f_{1,j}$ will have input x_j. Then, all the nodes in the network perform their part of algorithm A, with each node g_j simulating the behavior of the corresponding virtual nodes.

3. As we will see later, algorithm A will eventually terminate (at all nodes, including the virtual nodes). When algorithm A terminates, every node $g_j \in G_i$ obtains an output value x'_j. Each node g_j sets its output for the MEQ problem to 0 if $x_j = x'_j$, and to 1 otherwise.

Figure 2 illustrates the construction of an MEQ protocol P for 3 nodes, g_1, g_2 and g_3, with a Byzantine broadcast algorithm A for 4 nodes, and at most 1 failure. The gray areas indicate the virtual networks created by nodes g_1, g_2 and g_3.

Observe that, from the perspective of nodes $g_1, \cdots, g_{n-t}$, the above execution is admissible when using algorithm A, when node f_l is faulty ($1 \leq l \leq t$) and behaves like $f_{l,j}$ to node g_j. Thus, the execution above will appear to be that of a network in which t nodes $f_1, \cdots, f_t$ are (possibly) faulty. Since algorithm A solves Byzantine broadcast with up to t failures, the algorithm will terminate in the above execution as well, and each node $g_j \in G_i$ will obtain an identical output value $x'_j = x$ for some value x. The exact value of x will depend on the inputs x_j.

Now consider two cases:

- $x_1 = x_2 = \cdots = x_{n-t} = z$ (input to the MEQ problem at each node in G_i is equal to z): In this case, observe that all the simulated sources nodes $f_{l,j}$ will have identical input, say, equal to z. It should be easy to see that the behavior of nodes in G_i will then be identical to the behavior of the actual network wherein node f_1 has input z, with all the nodes behaving correctly. Then the output value x from A must equal to z for all $g_j \in G_i$, and hence all the nodes in G_i will set their outputs for the MEQ problem to 0 correctly (since $x_j = z$).

- $\exists g_j, g_k \in G_i$ *s.t.* $x_j \neq x_k$ (the input for the MEQ problem at the nodes in G_j is not identical): In this case as well, as noted above, the output x at all the nodes in G_i is identical by the definition of algorithm A. However, since not all x_j's are equal, it follows that, x must be different from x_j at some node $g_j \in G_i$. This node g_j will set its output for the MEQ problem to 1, and inequality of the inputs will be correctly detected.

Now we can conclude that, given any algorithm A that solves Byzantine broadcast in G at some rate R, we can construct a protocol P that solves the MEQ problem in G_i at the same rate R, when the source is not in G_i. According to Equation 3, the MEQ problem in G_i cannot be solve at rate higher than W_i. This implies that $C_{BB}(G) \leq W_i$.

The discussion when the source, namely g_1, is in G_i is almost the same. We can construct the virtual network for each g_j in the same way as described above, with the following modifications: (1) node g_1 sets x_1, its input for the MEQ problem, as the value that it will broadcast using algorithm A; and (2) node $g_j \neq g_1$ sets x_j as the initial input at node $g_{1,j}$, i.e., the virtual node corresponding to node g_1. Then we can prove that $C_{BB}(G) \leq W_i$ when the source is in G_i using the same argument above. Then Equation 30 immediately follows when all G_i's are considered.

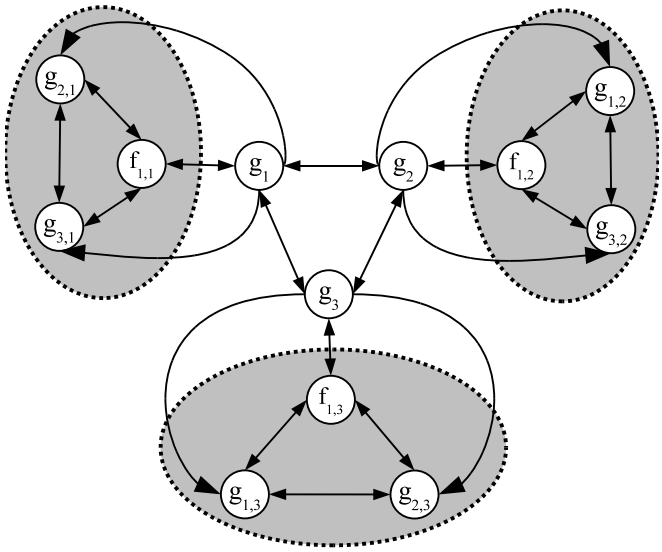


Figure 2: Solving MEQ in 3 nodes with Byzantine broadcast algorithm for 4 nodes.