Byzantine Broadcast in Point-to-Point Networks using Local Linear Coding [*]

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

Abstract

The goal of Byzantine Broadcast (BB) is to allow a set of fault-free nodes to agree on information that a source node wants to broadcast to them, in the presence of Byzantine faulty nodes. We consider design of efficient algorithms for BB in point-to-point networks where the rate of transmission over each communication link is limited by its "link capacity". Given an algorithm A to solve BB in a network G, let us denote by $t(G, L, A)$ the worst-case execution time of A without violating link capacity constraints in G, when L is the size of the input at the source node. Then, we define the *capacity* of BB in network G as the supremum of $L/t(G, L, A)$ over all L and all possible BB algorithms A.

We prove upper bounds on the capacity of Byzantine broadcast over arbitrary point-to-point networks. An algorithm is then given that solves BB at a rate of at least 1/2 or 1/3 of the capacity, depending on different conditions the underlying network satisfies. This Byzantine Broadcast algorithm tolerates up to f faulty nodes as long as the the total number of nodes is at least $3f + 1$ and the connectivity is at least $2f + 1$.

To the best of our knowledge, ours is the first algorithm that achieves a constant fraction of capacity of BB in general point-to-point networks.

[*]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

The problem of Byzantine Broadcast – also known as the Byzantine Generals problem [14] – was introduced by Pease, Shostak and Lamport in their 1980 paper [23]. Since the first paper on this topic, Byzantine Broadcast has been the subject of intense research activity, due to its many potential practical applications, including replicated fault-tolerant state machines [5], and fault-tolerant distributed file storage [25]. Informally, Byzantine Broadcast (BB) can be described as follows (we will define the problem more formally later). There is a source node that needs to broadcast a message (also called its *input*) to all the other nodes such that even if some of the nodes are *Byzantine faulty*, all the fault-free nodes will still be able to agree on an identical message; the agreed message is identical to the source's input if the source is fault-free.

We consider the problem of maximizing the *throughput* of Byzantine Broadcast (BB) in networks of point-to-point links, wherein each directed communication link is subject to a "capacity" constraint. Informally speaking, *throughput* of BB is the number of bits of Byzantine Broadcast that can be achieved per unit time (on average), under the worst-case behavior by the faulty nodes. Despite the large body of work on BB [9, 6, 3, 13, 2, 22], performance of BB in general point-to-point work has not been investigated previously. In reality, link capacities may differ significantly. Existing algorithms often do not perform well under such realistic conditions. In fact, one can easily construct example networks in which previously proposed algorithms achieve throughput that is arbitrarily worse than the optimal. In our prior work, we have developed an algorithm that optimizes the throughput in 4-node point-to-point networks [18], and also an optimal algorithm when total communication overhead is the cost metric [19]. In contrast, this paper presents a BB algorithm for arbitrary point-to-point networks. The paper makes two main contributions:

1. We prove upper bounds on the capacity of BB in point-to-point networks wherein each directed communication link is subject to a capacity constraint.
2. We present the first BB algorithm that achieves a constant fraction (1/2 or 1/3) of the capacity in *arbitrary* point-to-point networks.

2 Preliminaries

2.1 The Byzantine Broadcast (BB) Problem

Byzantine Broadcast is an example of a class of problems known as *Byzantine Agreement*. The BB problem considers a network of n nodes, named $1, 2, \cdots, n$, with one node designated as the *sender* or *source*, and the other nodes designated as the *peers*. In our discussion, we will assume that node 1 is the source node. Source node 1 is given an input value x containing L bits, and the goal here is for the source to broadcast its input to all the other nodes. The fault-free nodes must "agree on" the input value being broadcast by the source, despite the possibility that some of the nodes may be faulty (possibly including the source). In particular, the following conditions must be satisfied when the input value at the source node is x:

- **Termination:** Every fault-free node i must eventually decide on an output value; let us denote the output value of fault-free node i as x'_i.
- **Agreement:** All fault-free nodes must agree on an identical output value, i.e., there exists x' such that $x'_i = x'$ for each fault-free node i.
- **Validity:** If the source node is fault-free, then the agreed value must be identical to the input value of the source, i.e., $x' = x$.

Adversary Model: The faulty nodes are controlled by an adversary that has a complete knowledge of the network topology, the algorithm, and the information the source is trying to send. No secret is hidden from the adversary. The adversary can take over up to f nodes at any point during execution of the algorithm, where $f < n/3$. These nodes are said to be *faulty*. The faulty nodes can engage in any kind of deviations from the algorithm, including sending false messages, collusion, and crash failures.

2.2 Network Model

We assume a synchronous point-to-point network modeled as a directed simple graph $G(V, E)$, where the set of vertices $V = \{1, 2, \cdots, n\}$ represents the nodes in the point-to-point network, and the set of edges E represents the links in the network. With a slight abuse of terminology, we will use the terms *edge* and *link* interchangeably. We assume that $n \geq 3f + 1$ and that the network connectivity is at least $2f + 1$ (these two conditions are necessary for existence of a correct BB algorithm [9]).

In the given network, links may not exist between all node pairs. Each directed link is associated with a *fixed* link capacity, which specifies the maximum amount of information that can be transmitted on that link per unit time. Specifically, over a directed link (i, j) with capacity z bits/unit time, we assume that up to $z\tau$ bits can be reliably sent from node i to node j over time duration τ (for any non-negative τ). This is a deterministic model of *channel capacity* that has been widely used in the network coding literature [15, 4, 11, 12]. All link capacities are assumed to be integers. Rational link capacities can be turned into integers by choosing a suitable time unit. Irrational link capacities can be approximated by integers with arbitrary accuracy by choosing a suitably time unit. Propagation delays on the links are assumed to be zero (relaxing this assumption does not impact the correctness of results shown for large input sizes). We also assume that each node correctly knows the identity of the nodes at the other end of its links.

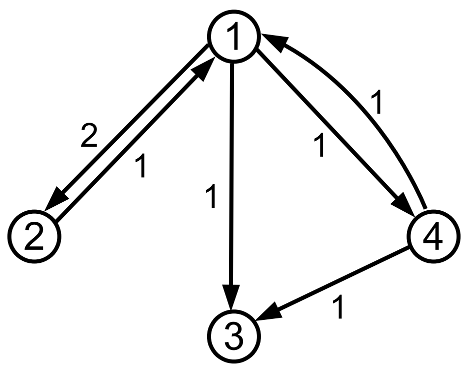
2.3 Multigraph Representations and MinCuts

In our discussion below, we will find it convenient to interpret a point-to-point network G as a **directed multigraph** with unit capacity edges. Thus, an edge $e = (i, j)$ of capacity z will now be modeled using z directed unit-capacity edges from node i to node j. Additionally, by ignoring the directions of edges in this multigraph, we obtain an **undirected multigraph** representation of the network, denoted as $\overline{G}$. Figures 1(a) to 1(c) illustrate examples of different representations of a graph. Define $MINCUT(G, i, j)$ as the *directed* mincut in directed graph G *from* node i to node j. Similarly, for the undirected representation, define $MINCUT(\overline{G}, i, j)$ as the *undirected* mincut in undirected graph $\overline{G}$ *between* node i and node j. Note that, while we always have $MINCUT(\overline{G}, i, j) = MINCUT(\overline{G}, j, i)$, in general $MINCUT(G, i, j)$ may not be equal to $MINCUT(G, j, i)$. We also define the following notations for later use:

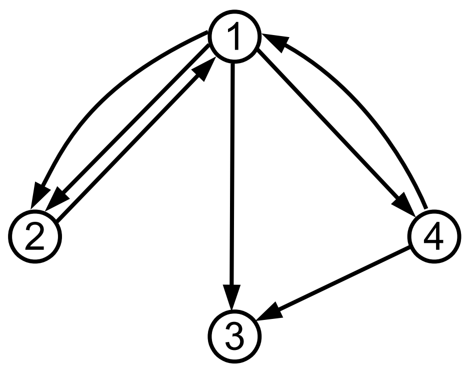
- $D(G, i) = \min_{j \in V, j \neq i} MINCUT(G, i, j)$: the minimum directed cut from node i to any of the other nodes in G. It is well-known that $D(G, i)$ is the broadcast capacity from node i in G [7] – *broadcast capacity* from node i characterizes the maximum rate at which node i can deliver information to all the other nodes in the network (in the absence of any failures).

- $U(G) = \min_{i, j \in V, i \neq j} MINCUT(\overline{G}, i, j)$: the minimum undirected cut between any pair of nodes in the undirected multigraph representation $\overline{G}$ of graph G.

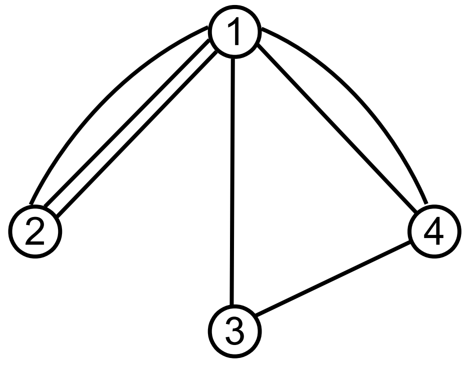
In the example of Figure 1, $MINCUT(G, 1, 4) = 1$, $MINCUT(\overline{G}, 1, 4) = 3$, $D(G, 1) = 1$, and $U(G) = 2$.



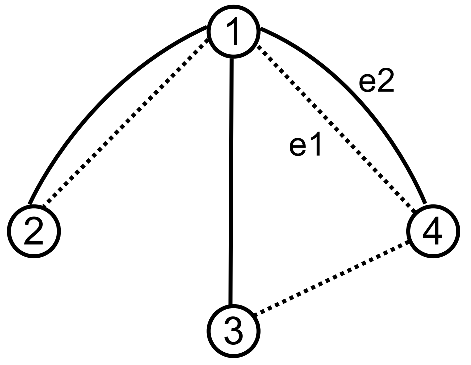
(a) Directed Simple Graph G



(b) Directed Multigraph representation of G, every directed edge has capacity 1



(c) Undirected Multigraph $\overline{G}$, every undirected edge has capacity 1



(d) Undirected unit-capacity spanning trees in $\overline{G}$

Figure 1: Graph representations of the network

2.4 Capacity of Byzantine Broadcast

We first define the throughput of a given algorithm in a given network. For any algorithm A that solves BB with an L-bit input at the source in network G, $t(G, L, P)$ is defined as the duration of time required, in the worst case, from the initiation of the algorithm until its termination, without violating the capacity constraints of the links in G. Throughput of algorithm A is then defined as $\frac{L}{t(G,L,P)}$. We define capacity of BB as follows.

Capacity C_{BB} of Byzantine Broadcast in graph G is defined as the supremum of the throughput of all algorithms that solve the BB problem. That is,

$$C_{BB}(G) \triangleq \sup_{A \text{ solves BB in } G} \frac{L}{t(G, L, A)}. \tag{1}$$

3 Algorithm Overview

The goal of the proposed BB algorithm is to perform Byzantine Broadcast of an L-bit input. The algorithm is designed to perform efficiently for large L. We first briefly describe the salient features of the BB algorithm.

Execution in Multiple Generations: To improve throughput, BB of an L-bit value is performed "in parts". In particular, for a certain integer B, the L-bit value is divided into L/B parts, each consisting of B bits. For convenience, we assume that L/B is an integer. A sub-algorithm (called Algorithm 1 below) is used to perform BB on each of these B-bit parts. We refer to each execution of the Algorithm 1 as a "generation". Algorithm 1 for each *generation* has the following structure:

Algorithm 1 Structure of the BB algorithm for generation g

1. `Broadcast with failure detection:` This phase of the algorithm allows the source node 1 to broadcast B bits of generation g. This phase also performs failure detection, while satisfying the following conditions: (i) each fault-free nodes agrees on whether a failure has been detected or not; (ii) if a failure is not detected, then the the broadcast values received by the fault-free nodes in generation g satisfy the *agreement* and *validity* conditions stated in Section 2.1, and (iii) if a failure is detected, then at least one faulty node has adversely affected the execution of generation g.

2. `Fault Diagnosis:` Whenever a failure is detected above, additional steps are taken to learn (possibly partial) information regarding the identity of the faulty node(s). This technique is also known as "dispute control" [1], and has been used in our previous work [18, 19]. The `Fault Diagnosis` phase identifies a pair of adjacent nodes, of which at least one node is guaranteed to be faulty. The directed links between these two nodes are not used in the future generations of Algorithm 1 (this is how the algorithm adapts to past failure detections). It can be shown that [18, 19], if a certain node is identified as potentially faulty in at least $f + 1$ generations, then that node must necessarily be faulty (and, therefore, excluded from the execution of future generations). Thus, the total number of generations during which faulty behavior is detected is limited by $f(f + 1)$.

Long-Term Throughput: Due to the bounded number of generations in which the faulty nodes can misbehave, it turns out that the throughput of the Byzantine Broadcast algorithm (which tolerates up to f Byzantine faults) can be made arbitrarily close to the throughput at which we can perform `Broadcast with failure detection` of Algorithm 1 above. For lack of space, we omit the proof of this claim. The proof follows from our prior work on throughput of Byzantine Broadcast [18].

Due to the above reason, we will now focus our attention on the problem of performing *Broadcast with Failure Detection* and propose an algorithm for this. Our algorithm will make use of a new solution to the *Multi-Party Equality* (MEQ) problem as a sub-algorithm. In Section 4 we discuss our algorithm for MEQ, and later describe how that can be used to solve the *broadcast with failure detection* problem.

4 The Multiparty Equality (MEQ) Problem

Let us consider the MEQ problem for the m nodes in a network $\mathcal{G}(\mathcal{V},\mathcal{E})$. Thus, $\mathcal{V}$ is the set of nodes in the network, and $\mathcal{E}$ is the set of directed edges in $\mathcal{G}$. As we will see later, our algorithm for BB in network G will apply the MEQ algorithm to various sub-graphs of G.

For the MEQ problem, each of the nodes in $\mathcal{V}$ starts with an input, and the objective of MEQ is for the nodes to determine whether they all share the same input. The MEQ problem does **not** consider faulty behavior. Formally, requirements of our version of MEQ are as follows:

- Each node i is given an input x_i consisting of B bits.
- Each node sets an output bit to be 0 or 1.
- If $x_1 = \cdots x_m$, then all the nodes set output to 0; otherwise, at least one node outputs 1.

We can define *throughput*, and *capacity* C_{MEQ}, of MEQ on network $\mathcal{G}$ analogous to throughput and capacity C_{BB} of Byzantine Broadcast. For lack of space, we do not repeat these definitions here.

Theorem 1 *In point-to-point network $\mathcal{G}(\mathcal{V},\mathcal{E})$, the capacity of the MEQ problem is upper bounded by the minimum undirected cut, that is, $C_{MEQ}(\mathcal{G}) \le U(\mathcal{G})$.*

Theorem 1 is a natural extension of the well-known result on the communication complexity of the two-party equality problem [26]. The proof is included in Appendix A. In the rest of this section, we discuss our algorithm that solves MEQ with throughput at least $U(\mathcal{G})/2$ using random linear coding. While the MEQ problem may potentially be solved using other algorithms, the structure of our MEQ algorithm is designed to help solve the BB problem, as seen later.

We refer to the strategy used in the algorithms below as "local coding" to contrast it with traditional network coding strategies [11, 12]. Specifically, in our algorithms, as will be seen later, the coding scheme does not combine packets from different sources.

4.1 Solving the MEQ Problem along Spanning Trees

We first present an algorithm that solves MEQ using a set of edge-disjoint spanning trees in $\overline{\mathcal{G}}$. Given the undirected multigraph $\overline{\mathcal{G}}$, define R as the "spanning tree packing number" of $\overline{\mathcal{G}}$, which is the maximum number of edge-disjoint undirected unit-capacity spanning trees that can be "packed" in $\overline{\mathcal{G}}$ [21]. Figure 1(d) shows two spanning trees packed in the graph in Figure 1(c). It is well-known [16] that

$$U(\mathcal{G})/2 \le R \le U(\mathcal{G}).$$

The unit-capacity edges included in the R disjoint spanning trees of $\overline{\mathcal{G}}$ will be used in Algorithm 2 below for solving MEQ in $\mathcal{G}$. The direction in which each such edge is used is the same as the direction of the corresponding edge in the directed multigraph representation of $\mathcal{G}$. For instance, in Figure 1(d), edge $e1$ is used to in the direction from node 1 to node 4, because the corresponding unit-capacity directed edge in Figure 1(b) is in that direction. Similarly, edge $e2$ in Figure 1(d) is used to send packets from node 4 to node 1, because the corresponding edge in Figure 1(b) is in

Algorithm 2 MEQ along R Spanning Trees

1. For each unit-capacity edge $e = (i, j)$ that is in one of the R trees in $\overline{\mathcal{G}}$, chooses a row vector $c_e = [\, c_e(1), c_e(2), \cdots, c_e(R)\,]$ (called a coefficient vector) containing R symbols, each chosen uniformly randomly from $GF(2^{B/R})$. Node i transmits to node j symbol $y_e = c_e x_i^T$, which is a random linear combination of the symbols in x_i. y_e is said to be a "coded symbol".
2. At each node j, for each unit-capacity link $e = (i, j)$ that belongs to one of the R trees, node j checks whether y_e equals $c_e x_j^T$. The check is said to fail if $y_e \neq c_e x_j^T$. (Recall that y_e is $c_e x_i^T$.)
3. At each node j, if any of the received symbols fails the check in step 2, then node j sets its output bit to 1; otherwise node j sets its the output to 0.

that direction. In the algorithm below, we will denote the links (or edges) by the corresponding node pair listed in the order in which the edge can be used. For instance, $e1 = (1, 4)$ and $e2 = (4, 1)$.

We assume that B/R is an integer, and represent the B-bit input x_i at each node i as a row vector of R symbols from Galois Field $GF(2^{B/R})$: $[x_i(1), \cdots, x_i(R)]$.

Suppose that $\mathcal{G}$ is identical to graph G in Figure 1. In this case, assuming that the two undirected spanning trees showed in Figure 1(d) are used, node 1 sends one random linear combination of its input symbols to node 4 using the directed unit-capacity edge from node 1 to node 4, as specified by the spanning tree showed in dotted lines. Similarly node 4 sends one random linear combination of its R/B input symbols to node 1, using the edge specified by the spanning tree showed in solid line. The coefficients in steps 1 and 2, although randomly chosen, can be viewed as a part of the algorithm specification, and assumed to be known to all the nodes (i.e., the coefficients are chosen at algorithm design time, not at runtime).

4.2 Correctness of Algorithm 2

Recall that m is the number of nodes in $\mathcal{G}$. Let us define $d_i(r) = x_i(r) - x_m(r)$ as the difference between x_i and x_m at the r-th symbol. Step 2 of Algorithm 2 determines, for link $e = (i, j)$, whether $c_e x_i^T = c_e x_j^T$, which is equivalent to checking whether $c_e(1)(d_i(1) - d_j(1)) + \cdots + c_e(R)(d_i(R) - d_j(R)) = 0$. Such a check is performed for each of the $(m-1)$ edges in each of the R-trees, resulting in $(m-1)R$ checks. These $(m-1)R$ checks together can be represented in a matrix form as

$$M_R\, d^T = 0, \tag{2}$$

where M_R is a $(m-1)R$-by-$(m-1)R$ square matrix defined by the elements of c_e vectors for all links e in the R spanning trees, and $d = [d_1(1), \cdots, d_{m-1}(1), d_1(2), \cdots, d_{m-1}(2), \cdots, d_1(R), \cdots, d_{m-1}(R)]$.

In Algorithm 2, all m output bits are set to 0 if and only if $M_R d^T = 0$. According to the definition of the MEQ problem, all the output bits should be 0 if and only if $x_1 = \cdots = x_m$. Also, $x_1 = \cdots = x_m$ if and only if $d = 0$. Finally, if M_R is invertible, then the only solution of $M_R\, d^T = 0$ is $d = 0$. It then follows that Algorithm 2 is correct if M_R is invertible.

Theorem 2 *When the coefficients in step 1 of Algorithm 2 are chosen independently and uniformly at random from $GF(2^{B/R})$, matrix M_R is invertible with probability at least $1 - \frac{(m-1)R}{2^{B/R}}$. Thus, for large enough B, Algorithm 2 is correct with a non-zero probability.*

Proof Sketch: Let us label the R undirected spanning trees as $T_1, \cdots, T_R$. We can order the rows of M_R such that rows $(k-1)(m-1)+1$ to $k(m-1)$ correspond to the $m-1$ edges on spanning tree T_k. For each spanning tree T_k, assign a unique index from 1 to $m-1$ to its $m-1$ edges. Denote by $c_{k,l}(r)$ the randomly chosen coefficient for the r-th input symbol used for generating the coded symbol sent on the l-th edge of tree T_k.

Define $C_{k,r} = diag(c_{k,1}(r), \cdots, c_{k,m-1}(r))$ as the diagonal matrix with the diagonal elements filled with the r-th coefficient used on the $m-1$ edges on tree T_k. Also, define $(m-1)$-by-$(m-1)$ matrix A_k to represent the $m-1$ edges on tree T_k as follows: (1) if the l-th directed edge on T_k is pointing from node i to node j, then the l-th row of A_k has the i-th element as -1 and the j-th element as 1, the remaining entries in the l-th row being 0; (2) if $i = m$ then j-th element of the l-th row is set to 1, the remaining elements of that row being 0; (3) if $j = m$ then i-th element of the l-th row is set to -1, the remaining elements of that row being 0. As an example, the spanning tree marked by dotted lines in Figure 1(d) contains three edges ($m = 4$). Suppose that we index the edges from 1 to 3 in the following order: (1,2), (1,4), (4,3). The resulting A matrix for this tree is $\begin{pmatrix} -1 & 1 & 0 \\ -1 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix}$.

Now M_R can be written in the following form:

$$M_R = \begin{pmatrix} C_{1,1}A_1 & C_{1,2}A_1 & \cdots & C_{1,R}A_1 \\ C_{2,1}A_2 & C_{2,2}A_2 & \cdots & C_{2,R}A_2 \\ \vdots & & \ddots & \vdots \\ C_{R,1}A_R & C_{R,2}A_R & \cdots & C_{R,R}A_R \end{pmatrix}. \tag{3}$$

Define $M_k = \begin{pmatrix} C_{1,1}A_1 & \cdots & C_{1,k}A_1 \\ \vdots & \ddots & \vdots \\ C_{k,1}A_k & \cdots & C_{k,k}A_k \end{pmatrix}$ for $1 \leq k \leq R$. Note that M_{k1} is a sub-matrix of M_{k2} when $k1 < k2$. We can show by induction that, with probability at least $\left(1 - \frac{m-1}{2^{B/R}}\right)^k$, determinant of M_k is non-zero. Then the theorem follows by setting $k = R$. The detailed proof is in Appendix B. □

Theorem 2 implies that, for sufficiently large B, there exist a set of coefficient vectors that make M_R invertible. Then Algorithm 2 deterministically solves the MEQ problem in $\mathcal{G}$ with this set of coefficient vectors.

Observe that Algorithm 2 requires only one round of communication, the length of a round being equal to the transmission time of B/R bits (one symbol from $GF(2^{B/R})$) along a unit-capacity edge. Without loss of generality, assume that the unit of capacity is 1 bit/time unit. Then it takes B/R time units for Algorithm 2 to terminate, and its throughput is $\frac{B}{B/R} = R$ bits/unit time. Recall that $R \geq U(\mathcal{G})/2$ and $C_{MEQ}(\mathcal{G}) \leq U(\mathcal{G})$. Thus, the throughput of Algorithm 2 is at least $C_{MEQ}(\mathcal{G})/2$.

4.3 Solving the MEQ Problem without Knowing the Spanning Trees

In Algorithm 2, we have assumed that the R undirected unit-capacity spanning trees are known a priori. It turns out that knowing the actual trees is not necessary – knowing the value R suffices. Here we present Algorithm 3 that solves the MEQ problem with parameter $\rho \leq R$, which achieves throughput ρ without knowing any spanning tree. For this algorithm, the input value at each node i is represented as a row vector of ρ symbols from $GF(2^{B/\rho})$, similar as in Algorithm 2.

Since $\rho \leq R$, there exists a set of ρ undirected spanning trees in $\overline{\mathcal{G}}$, which represents a subgraph of the network $\mathcal{G}$, all the tree edges will also carry randomly coded symbols (without explicit knowledge of the trees). As a result, Theorem 2 continues to hold for Algorithm 3 by substituting R with any $\rho \leq R$.

5 Byzantine Broadcast (BB) with Local Linear Coding

In this section, we present an algorithm for Byzantine Broadcast in $G(V, E)$, with node 1 as the source, tolerating up to f Byzantine faults, when the network connectivity is at least $2f + 1$, and $n \geq 3f + 1$. Connectivity $\geq 2f + 1$ implies that for every pair of nodes i and j in G, after removing

Algorithm 3 MEQ with parameter ρ

```
1. For every unit-capacity edge $e = (i, j)$ in $\mathcal{G}$, chooses a coefficient vector $c_e = [\, c_e(1), c_e(2), \cdots, c_e(\rho)\,]$ containing $\rho$ symbols, each chosen uniformly randomly from $GF(2^{B/\rho})$. Node $i$ transmits to node $j$ symbol $y_e = c_e x_i^T$, which is a random linear combination of the symbols in $x_i$. $y_e$ is said to be a "coded symbol".
2. At each node $j$, for each unit-capacity link $e = (i, j)$ in $\mathcal{G}$, node $j$ checks whether $y_e$ equals $c_e x_j^T$. The check is said to fail if $y_e \neq c_e x_j^T$. (Recall that $y_e$ is $c_e x_i^T$.)
3. At each node $j$, if any of the received symbols fails the check in step 2, then node $j$ sets its output bit to 1; otherwise node $j$ sets its the output to 0.
```

any subset of $2f$ nodes, there is still a positive directed mincut from i to j.

5.1 Upper Bounds of the Capacity $C_{BB}(G)$ of BB

We first prove upper bounds on the capacity of the Byzantine Broadcast in network $G(V, E)$ as functions of cut sets in the following two classes of its subgraphs:

1. $\binom{n}{n-f}$ subgraphs of $G(V, E)$, each containing $n - f$ nodes. These subgraphs are named $G_1, \cdots, G_{\binom{n}{n-f}}$. For each G_k, let R_k be the spanning tree packing number of $\overline{G}_k$. Define
$$R^* = \min_{G_k} R_k.$$
Then, at least R^* undirected spanning trees can be packed in each $\overline{G}_k$.
2. A subgraph in the second class is obtained as follows: We will say that edges in $F \subset E$ are "explainable" if there exists a set $W \subset V$ such that (i) W contains at most f nodes, and (ii) each edge in F is incident on at least one node in W. Set W is then said to "explain set F". Consider each *explainable* set of edges F_i. Suppose that $W_1, \cdots, W_K$ are all the subsets of V that *explain* edge set F_i. Subgraph H_i is obtained by removing edges in F_i from E, and nodes in $\bigcap_{k=1}^{K} W_k$ from V. [1] In general, H_i above may or may not contain the source node 1. We only need to focus on those H_i's that do contain node 1. We rename the subgraphs H_i's that contain source node 1 as $G'_1, G'_2, \cdots, G'_k, \cdots$.

Theorem 3 *In a point-to-point network $G(V, E)$, the capacity of the BB (C_{BB}) with node 1 as the source satisfies the following upper bounds, using subgraphs G_k and G'_k defined above.*

1. *C_{BB} is upper bounded by the capacity of MEQ in any subgraph G_k, that is, $C_{BB}(G) \leq \min_{G_k} C_{MEQ}(G_k)$;*
2. *C_{BB} is upper bounded by the capacity of broadcast from node 1 in any subgraph G'_k in the absence of failures, that is, $C_{BB}(G) \leq \min_{G'_k} D(G'_k, 1)$.*

Please see Section 2.3 for definition of broadcast capacity D used above, and U used later below. The proof of Theorem 3 is included in Appendix E. The intuition behind the first condition above is that, given any BB algorithm A in network G with throughput R, algorithm A can also be used to solve the MEQ problem in any subgraph G_k with throughput R; hence $R \leq C_{MEQ}(G_k)$.

The intuition for the second condition is as follows: Suppose that subgraph G'_k was constructed using an explainable edge set F_i (see above). Due to the manner in which subgraph G'_k is defined, for any node j in G'_k, there must exist a set of nodes W_j that explains F_i such that $j \notin W_j$. Consider a scenario in which the nodes in W_j are faulty, and all other nodes are fault-free. Also suppose

[1] It is possible that H_i for different i may be identical. This does not affect the correctness of our algorithm.

that these faulty nodes misbehave by refusing to communicate with neighbors over the links in F_i (essentially pretending that the corresponding neighbors are faulty), but behave correctly otherwise. Despite such misbehavior, as Appendix E elaborates, node j must still be able to learn the value that source node 1 (possibly faulty) is broadcasting. So $C_{BB}(G)$ cannot exceed the broadcast capacity from node 1 to the nodes in G'_k, that is, $D(G'_k, 1)$.

Let $U^* = \min_{G_k} U(G_k)$ and $D^* = \min_{G'_k} D(G'_k, 1)$. We have the following from Theorems 1 and 3:

Corollary 1 *The capacity of the BB problem in arbitrary network G satisfies:*

$$C_{BB}(G) \leq \min(U^*, D^*).$$

5.2 BB with Local Linear Coding

Using Algorithm 3 presented in the previous section, we construct a BB algorithm that achieves at least 1/2 or 1/3 of the upper bound in Corollary 1, hence at least 1/2 or 1/3 of $C_{BB}(G)$, depending on the relationship between certain cuts of graph G.

Suppose that the source node (node 1) wants to broadcast an L-bit value x using our BB algorithm. This L-bit input value x is divided into L/B parts of size B bits each, as discussed in Sections:overview. These parts are denoted as $x(1), \cdots, x(L/B)$. Algorithm for L-bit BB consists of L/B sequential executions (generations) of Algorithm 4 presented below – note that Algorithm 4 is a more detailed version of Algorithm 1 in Section 3. For the g-th generation ($1 \leq g \leq L/B$), the source node 1 uses $x(g)$ as its input in Algorithm 4. Each generation of the algorithm results in all nodes deciding on the g-th part (namely $x'_i(g)$) of its final decision value x'_i.

For the subsequent discussion in this section, it will be helpful if the reader has read Algorithm 4 first. The proof of the correctness of Algorithm 3 includes proving the following three claims. We will argue the correctness of the claims later.

- **Claim 1:** In the absence of misbehavior by nodes in G', broadcast in G' in step 1 will deliver data from node 1 to other peers in G' at rate D^*. If some nodes in G' misbehave, then broadcast may be received incorrectly.
- **Claim 2:** Performing Algorithm 3 with parameter R^* over graph $\mathcal{G} = G'$ in step 2 guarantees that

  (a) If no failure is detected, then all fault-free nodes must have identical x_i's, and hence BB for the current generation is correct; or

  (b) If failure is detected, some faulty node must have misbehaved. In this case fault diagnosis is performed.
- **Claim 3:** During fault diagnosis in step 3, at least one edge and/or one node will be identified as faulty and will be removed from G'. All fault-free nodes and the links between them remain in G' throughout the whole algorithm.

Now we argue the correctness of the claims.

Broadcast (claim 1): It can be shown that G' belongs to the second class of subgraphs of G defined earlier in this section, and all the fault-free nodes are always included in G'. Recall that $D^* = \min_{G'_k} D(G'_k, 1) \leq D(G', 1)$. So the broadcast from node 1 in G' at rate D^* is achievable with simple store-and-forward routing [7], if no node in G' misbehaves. Data received by a fault-free peer i during this broadcast is named x_i.

Algorithm 4 BB Algorithm (generation g)

Let G' be the subgraph of G obtained by removing from G those edges and nodes that have been identified as faulty during the `Fault Diagnosis` phase. Graph G' evolves with time, and only the nodes in G' perform the algorithm below. It can be shown that all the fault-free nodes, and the links between them, are always in G'.

1. `Broadcast:` Source node 1 uses a traditional store-and-forward approach to broadcast the g-th part of its input, $x(g)$, to the nodes in subgraph G', at rate D^*. This broadcast is not fault-tolerant. Source node 1 sets $x_1 = x(g)$, and each peer i ($i \neq 1$) sets x_i equal to the B bits received during the above broadcast (if nothing is received, then x_i is set to some default value).

2. `Failure Detection:` Perform Algorithm 3 with parameter R^* over graph $\mathcal{G} = G'$ and x_i's at the nodes in G' as the inputs (we will justify the use of G' and R^* in the discussion below).

   (a) Each node in G' obtains an output bit from the MEQ algorithm execution. Each node in G' broadcasts its output bit using a traditional 1-bit Byzantine Broadcast algorithm, denoted as `Broadcast_Binary` (e.g., algorithm in [6, 3] may be used).

   Failure is detected if the output bit broadcast by any of the nodes in G' is 1.

   (b) If failure is not detected, every fault-free node i in G' decides on output $x'_i(g) = x_i$, and proceeds to the next generation. Otherwise, the fault-free nodes perform `Fault Diagnosis`.

3. `Fault Diagnosis:` Performed only when failure is detected during `Failure Detection`. Information about the identity of the faulty node(s) is updated using "dispute control" mechanism [1]. During `Fault Diagnosis`, node 1 once again broadcasts x_1 using algorithm `Broadcast_Binary`. By the end of `Fault Diagnosis`, additional nodes and/or links in G' may be identified as faulty, and removed from G' used in the subsequent generations.

   (a) If node 1 is found faulty in the `Fault Diagnosis` step, terminate the algorithm with a default output.

   (b) Otherwise, every fault-free node i sets $x'_i(g)$ equal to x_1 broadcast by node 1 during `Fault Diagnosis`, and proceeds to the next generation.

Failure Detection (claim 2): For failure detection, our objective is to solve the MEQ problem in every subgraph G_k that is also a subgraph of G' (recall that G_k is in the first class of subgraphs of G defined earlier), and to determine whether *absence of equality* is detected during *at least one* of the MEQ executions. A naive approach for this is to run the MEQ algorithm independently for each of these subgraphs G_k (recall that, each G_k considered here is also a subgraph of G'). If in any of these executions, a node in one of the subgraphs G_k sets its output to 1, then an *absence of equality* would be detected. However, we only need to know whether equality check failed in one of these subgraphs; identify of the subgraph is not required to be known. Due to this, along with the the local communication (only between adjacent nodes) used in MEQ Algorithm 3, we can achieve the goal more efficiently. In particular, we can apply Algorithm 3 just once to graph G' itself, with parameter ρ for the algorithm chosen as to not exceed the number of disjoint undirected spanning trees in any of the subgraphs G_k that are included in G'. In particular, we choose $\rho = R^*$.

By a simple extension of Theorem 2, it can be shown that Algorithm 3 with parameter R^* can detect, with a non-zero probability, the absence of equality in all G_k that are subgraphs of G'.

This claim follows from the theorem below. Let us define matrix $M^{(k)}$ used in the theorem first. Each subgraph G_k contains $n - f$ nodes. For each subgraph G_k included in G', let us pick any set of R^* undirected spanning trees of $\overline{G_k}$ and construct a square matrix, say $M^{(k)}$, of size $(n - f - 1)R^*$, similar to M_R in the previous section. An edge being reused in spanning trees of multiple subgraphs $\overline{G_k}$'s is equivalent to the corresponding $M^{(k)}$'s having one row in common (up to permutation of node indices). The theorem below (proof included in Appendix F) states that, for sufficiently large B, all $M^{(k)}$ matrices can be made invertible simultaneously:

Theorem 4 *When performing Algorithm 3 with parameter* $\rho = R^*$ *in* $\mathcal{G} = G'$, *all matrices* $M^{(k)}$ *are invertible simultaneously with probability at least* $1 - \binom{n}{n-f}(n - f - 1)R^*/2^{B/R^*}$.

It can be shown that G' always contains all fault-free nodes as well as the links between then. So the union of all such subgraphs G_k includes all the fault-free nodes and the links between them. As noted above, if the inputs (at the fault-free nodes) to Algorithm 3 are unequal, then the inequality will be detected by Algorithm 3. Alternatively, it is also possible that a faulty node in G' may misbehave leading to the conclusion the inputs are unequal. It follows that if the output bits broadcast after `Failure Detection` step are all 0, then all fault-free nodes i will have identical x_i value, and hence BB of this generation is correct. Otherwise, some node must have misbehave sometime during the current generation (including possibly node 1).

Fault Diagnosis (claim 3): The `Fault Diagnosis` step using "dispute control" [1] is then performed: this step results in either (a) a specific node as being identified as the culprit during generation g, or (b) the identity of the misbehaving node is narrowed down to a pair of neighboring nodes (that is connected in G'), such that at least one of the nodes is certain to be faulty. The link between this pair of nodes is then deemed "faulty" and not used in future generations. Also, due to the use of "dispute control", fault diagnosis will be performed for at most $f(f + 1)$ times throughout the whole execution. Please see Appendix G for more details.

5.3 Throughput of Algorithm 4

In Appendix H, we show that the throughput of Algorithm 4 can be made arbitrarily close to $\frac{D^*R^*}{D^*+R^*}$ for large enough B and $L = \Omega(B)$. Then we further prove the following theorem:

Theorem 5 *For sufficiently large B and* $L = \Omega(B)$, *throughput of Algorithm 4 satisfies:*

$$\text{Throughput} \geq \frac{C_{BB}(G)}{2} \qquad \text{if } D^* \leq R^*; \tag{4}$$

$$\text{Throughput} \geq \frac{C_{BB}(G)}{3} \qquad \text{if } D^* > R^*. \tag{5}$$

6 Conclusion

We prove upper bounds on the capacity of Byzantine Broadcast in general point-to-point networks. A local linear coding based BB algorithm that achieves throughput at least 1/2 or 1/3 of the capacity C_{BB} in general point-to-point networks is introduced. To the best of our knowledge this is the first result for the BB problem that achieves a constant factor of capacity for general point-to-point networks.

References

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

[2] Z. Beerliova-Trubiniova and M. Hirt. Perfectly-secure mpc with linear communication complexity. In *TCC*, 2008.

[3] P. Berman, J. A. Garay, and K. J. Perry. Bit optimal distributed consensus. *Computer science: research and applications*, 1992.

[4] N. Cai and R. W. Yeung. Network error correction, part ii: Lower bounds. *Communications in Information and Systems*, 2006.

[5] M. Castro and B. Liskov. Practical byzantine fault tolerance. In *OSDI*, pages 173–186, Berkeley, CA, USA, 1999.

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

[7] J. Edmonds. *Edge-Disjoint Branchings*. Combinatorial Algorithms, Academic Press, 1973.

[8] M. J. Fischer. The consensus problem in unreliable distributed systems (a brief survey). In *Proceedings of the 1983 International FCT-Conference on Fundamentals of Computation Theory*, pages 127–140, London, UK, 1983. Springer-Verlag.

[9] M. J. Fischer, N. A. Lynch, and M. Merritt. Easy impossibility proofs for distributed consensus problems. *Distrib. Comput.*, 1986.

[10] T. Ho, R. Koetter, M. Medard, D. Karger, and M. Effros. The benefits of coding over routing in a randomized setting. In *Information Theory, 2003. Proceedings. IEEE International Symposium on*, page 442, june-4 july 2003.

[11] T. Ho, B. Leong, R. Koetter, M. Medard, M. Effros, and D. Karger. Byzantine modification detection in multicast networks using randomized network coding, 2004.

[12] S. Jaggi, M. Langberg, S. Katti, T. Ho, D. Katabi, and M. Medard. Resilient network coding in the presence of byzantine adversaries. In *INFOCOM'07*, 2007.

[13] V. King and J. Saia. Breaking the $o(n^2)$ bit barrier: scalable byzantine agreement with an adaptive adversary. In *Proceeding of the 29th ACM SIGACT-SIGOPS symposium on Principles of distributed computing*, PODC '10, pages 420–429, New York, NY, USA, 2010. ACM.

[14] L. Lamport, R. Shostak, and M. Pease. The byzantine generals problem. *ACM Trans. on Programming Languages and Systems*, 1982.

[15] S.-Y. Li, R. Yeung, and N. Cai. Linear network coding. *Information Theory, IEEE Transactions on*, 49(2):371–381, Feb. 2003.

[16] Z. Li and B. Li. Network coding in undirected networks. In *Conference on Information Sciences and Systems*, 2004.

[17] G. Liang and N. Vaidya. Multiparty equality function computation in networks with point-to-point links. In *SIROCCO*, 2010.

[18] G. Liang and N. Vaidya. Capacity of byzantine agreement with finite link capacity. In *INFOCOM 2011*, 2011.

[19] G. Liang and N. Vaidya. Error-free multi-valued consensus with byzantine failures. In *ACM PODC*, 2011.

[20] G. M. Masson, D. M. Blough, and G. F. Sullivan. *System diagnosis*. Fault-Tolerant Computer System Design. Prentice Hall, 1996.

[21] E. M. Palmer. On the spanning tree packing number of a graph: a survey. *Discrete Math.*, 230:13–21, March 2001.

[22] A. Patra and C. P. Rangan. Communication optimal multi-valued asynchronous byzantine agreement with optimal resilience. Cryptology ePrint Archive, 2009.

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

[24] F. P. Preparata, G. Metze, and R. T. Chien. On the connection assignment problem of diagnosable systems. *Electronic Computers, IEEE Transactions on*, EC-16(6):848–854, Dec. 1967.

[25] A. Silberschatz, P. B. Galvin, and G. Gagne. *Operating System concepts, chapter 17 Distributed file systems*. Addison-Wesley, 1994.

[26] A. C.-C. Yao. Some complexity questions related to distributive computing(preliminary report). In *STOC '79*, 1979.

[27] R. W. Yeung and N. Cai. Network error correction, part i: Basic concepts and upper bounds. *Communications in Information and Systems*, 2006.

Appendices

A Proof of Theorem 1

Suppose the cut between a certain set $s \subset \mathcal{V}$ and $\mathcal{V} - s$ is the minimum undirected cut in $\overline{\mathcal{G}}$, the undirected multigraph representation of $\mathcal{G}$. Now consider a constrained version of the MEQ problem wherein all nodes in s are have an identical L-bit input, denoted as x, and all nodes in $\mathcal{V} - s$ have another identical L-bit 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 $\mathcal{V} - s$, 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 $\mathcal{G}$ are constrained as described above, the execution time $t(\mathcal{G}, L, A)$ must satisfy $U(\mathcal{G})t(\mathcal{G}, L, A) \geq L$ for any L and any MEQ algorithm A. This implies that $C_{MEQ}(\mathcal{G}) \leq U(\mathcal{G})$.

B Proof of Theorem 2

Proof: We now show that each M_k is invertible with probability at least $\left(1 - \frac{n-1}{2^{B/R}}\right)^k$. The proof is done by induction, with $k = 1$ being the base case.

Base Case – $k = 1$:

$$M_1 = C_{1,1}A_1 \tag{6}$$

As showed in Appendix C, A_k is always invertible and $\det(A_k) = \pm 1$. Since $C_{1,1}$ is a $(m-1)$-by-$(m-1)$ diagonal matrix, it is invertible provided that all its $m - 1$ diagonal elements are non-zero. Remember that the diagonal elements of $C_{1,1}$ are chosen uniformly and independently from $GF(2^{B/R})$. The probability that they are all non-zero is $\left(1 - \frac{1}{2^{B/R}}\right)^{m-1} \geq 1 - \frac{m-1}{2^{B/R}}$.

Induction Step – k to $k + 1 \leq R$: The $(m-1)(k+1)$-by-$(m-1)(k+1)$ matrix M_{k+1} can be written as

$$M_{k+1} = \begin{pmatrix} M_k & D_k \\ F_k & C_{k+1,k+1}A_{k+1} \end{pmatrix}, \tag{7}$$

where

$$D_k = \left(A_1^T C_{1,k+1}, A_2^T C_{2,k+1}, \cdots, A_k^T C_{k,k+1}\right)^T \tag{8}$$

is an $(m-1)k$-by-$(m-1)$ matrix, and

$$F_k = \left(C_{k+1,1}A_{k+1}, C_{k+1,2}A_{k+1}, \cdots, C_{k+1,k}A_{k+1}\right) \tag{9}$$

is an $(m-1)$-by-$(m-1)k$ matrix.

Assuming that M_k is invertible, we transform M_{k+1} as follows:

$$\begin{aligned} M'_{k+1} &= \begin{pmatrix} I_{(m-1)k} & 0 \\ -F_k M_k^{-1} & I_{(m-1)} \end{pmatrix} M_{k+1} \begin{pmatrix} I_{(m-1)k} & 0 \\ 0 & A_{k+1}^{-1} \end{pmatrix} & (10) \\ &= \begin{pmatrix} I_{(m-1)k} & 0 \\ -F_k M_k^{-1} & I_{(m-1)} \end{pmatrix} \begin{pmatrix} M_k & D_k \\ F_k & C_{k+1,k+1}A_{k+1} \end{pmatrix} \begin{pmatrix} I_{(m-1)k} & 0 \\ 0 & A_{k+1}^{-1} \end{pmatrix} & (11) \\ &= \begin{pmatrix} M_k & D_k A_{k+1}^{-1} \\ 0 & C_{k+1,k+1} - F_k M_k^{-1} D_k A_{k+1}^{-1} \end{pmatrix}. & (12) \end{aligned}$$

Here I_q denotes a q-by-q identity matrix. Note that $|\det(M'_{k+1})| = |\det(M_{k+1})|$, since the matrix multiplied at the left has determinant 1, and the matrix multiplied at the right has determinant ± 1.

Observe that the diagonal elements of the $(m-1)$-by-$(m-1)$ diagonal matrix $C_{k+1,k+1}$ are chosen independently from $F_k M_k^{-1} D_k A_{k+1}^{-1}$. Then it can be proved that $C_{k+1,k+1} - F_k M_k^{-1} D_k A_{k+1}^{-1}$ is invertible with probability at least $1 - \frac{m-1}{2^{B/R}}$ (See Appendix D.) given that M_k is invertible, which happens with probability at least $\left(1 - \frac{m-1}{2^{B/R}}\right)^k$ according to the induction assumption. So we have

$$\Pr\{M_{k+1} \text{ is invertible}\} \geq \left(1 - \frac{m-1}{2^{B/R}}\right)^k \left(1 - \frac{m-1}{2^{B/R}}\right) = \left(1 - \frac{m-1}{2^{B/R}}\right)^{k+1}. \tag{13}$$

This completes the induction. Now we can see that M_R is invertible with probability

$$\geq \left(1 - \frac{m-1}{2^{B/R}}\right)^R \geq 1 - \frac{(m-1)R}{2^{B/R}} \to 1, \text{ as } L \to \infty. \tag{14}$$

□

C Proof of A_k being Invertible

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 to node i has a non-zero entry in column i. Since there must be at least one edge in T_k that is incident on node n, there must be at least one row of A_k that has only one non-zero element. Also, since every node is incident to 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. Hence $\det(A_k)$ equals to either 1 or -1, and A_k is invertible.

D Proof of $C_{k+1,k+1} - F_k M_k^{-1} D_k A_{k+1}^{-1}$ being Invertible

Consider Q be an arbitrary *fixed* q-by-q matrix. Consider a random q-by-q diagonal matrix C with m diagonal elements $c_1, \cdots, c_q$.

$$C = \begin{pmatrix} c_1 & 0 & \cdots & 0 \\ 0 & c_2 & \cdots & 0 \\ \vdots & & \ddots & \vdots \\ 0 & \cdots & 0 & c_q \end{pmatrix} \tag{15}$$

The diagonal elements of C are selected independently and uniformly randomly from $GF(2^p)$. Then we have:

Theorem 6 *The probability that the q-by-q matrix $C - Q$ is invertible is lower bounded by:*

$$\Pr\{(C-Q) \textit{ is invertible}\} \geq 1 - \frac{q}{2^p}. \tag{16}$$

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

$$\begin{aligned} \det(C-Q) &= \det \begin{pmatrix} (c_1 - Q_{1,1}) & -Q_{1,2} & \cdots & -Q_{1,q} \\ -Q_{2,1} & (c_2 - Q_{2,2}) & \cdots & -Q_{2,q} \\ \vdots & & \ddots & \vdots \\ -Q_{q,1} & \cdots & -Q_{q,q-1} & (c_m - Q_{q,q}) \end{pmatrix} && (17) \\ &= (c_1 - Q_{1,1})(c_2 - Q_{2,2}) \cdots (c_q - Q_{q,q}) \ + \ \text{other terms} && (18) \\ &= \Pi_{i=1}^q c_i \ + \ Q_{q-} && (19) \end{aligned}$$

The first term above, $\Pi_{i=1}^{q} c_i$, is a degree-q polynomial of $c_1, \cdots, c_q$. Q_{q-} is a polynomial of degree at most $q-1$ of $c_1, \cdots, c_q$, and it represents the remaining terms in $\det(C - Q)$. Notice that $\det(C - Q)$ cannot be identically zero since it contains only one degree q term. Then by the Schwartz-Zippel Theorem, the probability that $\det(C - Q) = 0$ is $\leq q/2^p$. Since $C - Q$ is invertible if and only if $\det(C - Q) \neq 0$, we conclude that

$$\Pr\{(C - A) \text{ is invertible}\} \geq 1 - \frac{q}{2^p} \tag{20}$$

By setting $C = C_{k+1,k+1}$, $Q = F_k M_k^{-1} D_k A_{k+1}^{-1}$, $q = m - 1$ and $p = B/R$, we proof that $C_{k+1,k+1} - F_k M_k^{-1} D_k A_{k+1}^{-1}$ is invertible with probability at least $1 - \frac{m-1}{2^{B/R}}$. □

E Proof of Theorem 3

In arbitrary point-to-point network $G(V, E)$, the capacity of the BB problem with node 1 being the source and up to $f < n/3$ faults satisfies the following upper bounds:

E.1 First Condition

C_{BB} is upper bounded by the capacity of MEQ in any subgraph G_k, that is,

$$C_{BB}(G) \leq \min_{G_k} C_{MEQ}(G_k);$$

Proof: Given any subgraph G_k of G with size $n-f$, let us rename the nodes in G_k as $g_1, g_2, \cdots, g_{n-t}$, and the nodes not in G_i as $b_1, b_2, \cdots, b_f$.

We first discuss the case when the source of the Byzantine broadcast problem is not in G_k. Without loss of generality, we can assume that b_1 is the source.

Given any algorithm, namely A, that solves BB in network G, with node b_1 as the source, with at most f failures, and achieves throughput R, in the following, we construct a protocol A' that solves MEQ in G_k with throughput R as follows. For the MEQ problem, let us assume that x_i is the input value at node g_i. Thus, the goal is to determine whether x_i is identical at all $g_i \in G_k$. A' is constructed as follows:

1. Every node $g_i \in G_k$ creates a local virtual network as follows:
   - (a) It creates one virtual node $g_{j,i}$ for each $g_j \in G_k$, $j \neq i$. Similarly, it creates one virtual node $b_{l,i}$ for each $b_l \notin G_k$. Node g_i 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 $(g_{j,i}, g_{l,i})$ has the same capacity as link (g_j, g_l) in G. Similarly, link $(g_{j,i}, b_{l,i})$ has the same capacity as link (g_j, b_l) in G.
   - (c) Node g_i connects itself with each of the virtual nodes $b_{l,i}$ such that link $(g_i, b_{l,i})$ has the same capacity as link (g_i, b_l) in G, and link $(b_{l,i}, g_i)$ has the same capacity as link (b_l, g_i) in G.
   - (d) Node g_i connects itself with each of the virtual nodes $g_{j,i}$ with one link $(g_i, g_{j,i})$ that has the same capacity as link (g_i, g_j) in G. There is no link from virtual node $g_{j,i}$ to node g_i.
   - (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 $b_{l,i}$ is assigned the execution code that node b_l should run in A. Virtual node $g_{j,i}$ is assigned the execution code that

node g_j should run in algorithm A, except that it drops the messages that should be sent to node g_i (since there is no link from virtual node $g_{j,i}$ to node g_i).

2. Every node $g_i \in G_k$ executes correctly as specified by algorithm A. When algorithm A specifies that g_i should send a message to an actual node $g_j \in G_k$, g_i sends the message to both the actual node g_j and the virtual node $g_{j,i}$. When it should send a message to node $b_l \notin G_k$ according to algorithm P, g_i sends the message only to the virtual node $b_{l,i}$. When it receives an message from virtual node $b_{l,i}$, it pretends that the message is received from the actual node b_l. Since there is no link from virtual node $g_{j,i}$ to node g_i, g_i will not receive messages from $g_{j,i}$.

3. Recall that node b_1 is the source nodes for the broadcast being performed by algorithm A. Thus, each virtual node $b_{1,i}$ should be given an input value for algorithm A. Each node $g_i \in G_k$ sets the input value at node $b_{1,i}$ equal to x_i (recall that x_i is the input for the MEQ operation at node g_i). Thus, algorithm A for node $b_{1,i}$ will have input x_i. Then, all the nodes in the network perform their part of algorithm A, with each node g_i simulating the behavior of the corresponding virtual nodes.

4. As we will see later, algorithm A will eventually terminate (at all nodes, including the virtual nodes). When algorithm A terminates, every node $g_i \in G_k$ obtains an output value x'_i. Each node g_j sets its output for the MEQ problem to 0 if $x_i = x'_i$, 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 $g_1, \cdots, g_{n-f}$, what they see from the execution of A' is the same as what they would have seen from an execution of A when node b_l is faulty ($1 \le l \le f$) and behaves like $b_{l,i}$ to node g_i. Since algorithm A solves BB with up to f failures in G, it will terminate in the above execution as well, and each node $g_i \in G_k$ 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:

- $x_1 = x_2 = \cdots = x_{n-f} = z$ (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 $b_{l,i}$ will have identical input, say, equal to z. It should be easy to see that the behavior of nodes in G_k will then be identical to the behavior as in network G wherein node b_1 has input z, with all the nodes behaving correctly. Then the output value x from A must equal to z for all $g_i \in G_k$, and hence all the nodes in G_k will set their outputs for the MEQ problem to 0 correctly (since $x_i = z$).

- $\exists g_i, g_j \in G_k \; s.t. \; x_i \neq x_j$ (the inputs for the MEQ problem at the nodes in G_k are not identical): In this case as well, as noted above, the output x at all the nodes in G_k is identical by the definition of algorithm A. However, $x_i \neq x_j$, it follows that, x must be different from at least one of x_i and x_j. Without loss of generality, assume that $x_i \neq x$. Then node g_i 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 BB in G at some throughput R, we can construct a protocol A' that solves the MEQ problem in G_k at the same throughput R, when the source is not in G_k. This implies that $C_{BB}(G) \le C_{MEQ}(G_k)$.

The discussion when the source, namely g_1, is in G_k is almost the same. We can construct the virtual network for each g_i in the same way as described above, with the following modifications:

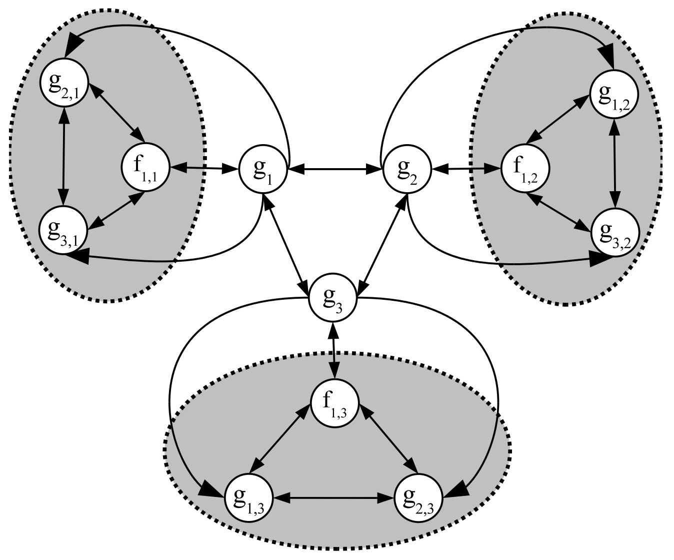


Figure 2: Solving MEQ in 3 nodes with Byzantine broadcast algorithm for 4 nodes.

(1) node g_1 sets x_1, its input for the MEQ problem, as the value that it will broadcast using algorithm P; and (2) node $g_i \neq g_1$ sets x_i as the initial input at node $g_{1,i}$, i.e., the virtual node corresponding to node g_1. Then we can prove that $C_{BB}(G) \leq C_{MEQ}(G_k)$ when the source is in G_k using the same argument above.

□

E.2 Second Condition

C_{BB} is upper bounded by the capacity of broadcast from node 1 in any subgraph G'_k in the absence of failures, that is,

$$C_{BB}(G) \leq \min_{G'_k} D(G'_k, 1).$$

Proof: Consider any G'_k and let $F_k \in E$ be the corresponding set of edges that are removed to construct G'_k. By the construction of G'_k, there must be at least one set $W \subset V$ that explains F_k and does not contain the source node 1, otherwise node 1 must have been removed. Without loss of generality, let W_1 be such a set that does not contain the source node. We are going to show that $C_{BB}(G) \leq MINCUT(G'_k, 1, i)$ for every peer node i that is in G'_k.

First consider any peer node i in G'_k but $i \notin W_1$. Let all the nodes in W_1 be faulty such that they refuse to communicate over edges in F_k, but otherwise behave correctly. In this case, since the source is fault-free, node i must be able to receive the value node 1 is trying to broadcast. So $C_{BB}(G) \leq MINCUT(G'_k, 1, i)$.

Now consider a peer node i in G'_k and $i \in W_1$. Notice that in this case $i \notin \bigcap_{l=1}^{K} W_l$, otherwise node i must have been removed. If there exists a set $W_j \subset V$ that explains F_k and contains neither node 1 nor node i, then following the same argument above, $C_{BB}(G) \leq MINCUT(G'_k, 1, i)$. Otherwise, there must exist a set W_j that contains node 1 but not node i, given that node i was not removed, i.e., $i \notin \bigcap_{l=1}^{K} W_l$. Without loss of generality, let W_2 be such a set. Define $V^- = V - W_1 - W_2$. V^- is not empty since W_1 and W_2 both contain at most f nodes and there are $n \geq 3f + 1$ nodes in V. Consider two scenarios: (1) nodes in W_1 are faulty and refuse to communicate over edges in F_k;

and (2) nodes in W_2 are faulty and refuse to communicate over edges in F_k. Observe that among edges between nodes in V^- and $W_1 \cup W_2$, only edges between V^- and $W_1 \cap W_2$ could have been removed, because otherwise F_k cannot be explained by both W_1 and W_2. So nodes in V^- cannot distinguish between these two scenarios. In scenario (1), the source (node 1) is not faulty. Hence nodes in V^- must agree with the value x that node 1 is trying broadcast, according to the validity condition. Since nodes in V^- cannot distinguish between the two scenarios, they must also set their outputs to x in scenario 2, even though in this case the source (node 1) is faulty. Then according to the agreement condition, node i must agree with nodes in V^- in scenario 2, which means that node i also learn x. So $C_{BB}(G) \le MINCUT(G'_k, 1, i)$.

Now we can conclude that $C_{BB}(G) \le D(G'_k, 1)$, and the theorem follows.

□

F Proof of Theorem 4

Theorem 4 *When performing Algorithm 3 with parameter $\rho = R^*$ in $\mathcal{G} = G'$, all matrices $M^{(k)}$ are invertible simultaneously with probability at least $1 - \binom{n}{n-f}(n-f-1)R^*/2^{B/R^*}$.*

Proof: Let K be the number of subgraphs G_k's included in G'. Without less of generality, rename these subgraphs as $G_1, \cdots, G_K$. Let

$$M^* = \prod_{k=1}^{K} M^{(k)}$$

. Since $R^* \le R_k$, according to Theorem 2, each $M^{(k)}$ matrix ($1 \le k \le K$) is invertible with non-zero probability. It implies that $\det(M^{(k)})$ is a not-identically-zero polynomial of the random coefficients of degree at most $(n-f-1)R^*$ (since $M^{(k)}$ is an square matrix of size $(n-f-1)R^*$). So

$$\det(M^*) = \prod_{k=1}^{K} \det\left(M^{(k)}\right)$$

is a non-identically-zero polynomial of the random coefficients of degree at most $K(n-f-1)R^*$. Notice that each coded symbol is used once in each subgraph G_k. So each random coefficient appears in at most one row in each $M^{(k)}$. It follows that the largest exponent of any random coefficient in $\det(M^*)$ is at most K.

According to Lemma 1 of [10], a non-identically-zero polynomial of degree no more than dv, in which the largest exponent of any variable is at most d, equals zero with probability at most $1-(1-d/q)^v$ for $d < q$, where q is the size of the field from which each variable is chosen. In our case, $d = K$, $v = (n-f-1)R^*$, and $q = 2^{B/R^*}$, so the probability that $\det(M^*)$ is non-zero is at least

$$\left(1 - K/2^{B/R^*}\right)^{(n-f-1)R^*} \ge 1 - K(n-f-1)R^*/2^{B/R^*}.$$

Since G' is a subgraph of G, and there are at most $\binom{n}{n-f}$ subgraphs G_k's in G, $K \le \binom{n}{n-f}$. Then the theorem follows. □

G Fault Diagnosis

For fault diagnosis, every node i in G' broadcasts everything it has received and sent in `Broadcast` and `Failure Detection` phases during the current generation with `Broadcast_Binary`. Due to the used of `Broadcast_Binary`, all fault-free nodes share the same information about what each node claims it has received and sent. We will show that, by comparing this information, at least one new pair of adjacent nodes, at least one of which must be faulty, will be identified.

First notice that everything a fault-free node sends in `Broadcast` and `Failure Detection` phases is a function of the information it receives, as specified by Algorithm 4. So if what a node i claims it has sent is inconsistent with what it claims to have received: (1) it claims to have received a symbol y during Broadcast phase, but it claims to have forwarded a different y' instead of y as it should have; or (2) the coded symbols it claims to have sent during Failure Detection are not a valid linear combination of x_i; or (3) the output bit of Algorithm 3 contradicts to x_i and the coded symbols it claims to have received, then node i must be faulty. Then faulty node i will be removed from G', and all its adjacent edges will be removed as well.

Now consider the case when every faulty node are "smart" enough so that what it claims to have sent is consistent with what it claims to have received. Then there must exist at least a pair of nodes i and j such that what node i claims to have sent to node j contradicts with what node j claims to have received from node i. Suppose on the contrary that there is no contradicting claims between two nodes, then the claims from all peers are all consistent with the $x(g)$ that node 1 claims it has sent. If so, then all output bits of Algorithm 3 from fault-free nodes should be 0. Since failure has been detected, at least one node k in G' must have set its output bit as 1. Realize that this fact contradicts with x_k and the coded symbols node k claims to have received, which contradicts with the assumption that every faulty node are "smart" enough so that what it claims to have sent is consistent with what it claims to have received. This completes the argument that at least one link in G' will be identified as guilty and will be removed.

It is easy to see that the claims of any two fault-free nodes never contradict, so at least one of the two nodes adjacent to a faulty link must be faulty.

After removing edges that are found faulty in G', we try to identify additional faulty nodes by applying the operation to construct the second class of subgraphs of G as described at the beginning of this section 5 G': find all subset $W_1, \cdots, W_K \subset V$ such that every W_k contains no more than f nodes and explains all edges that have been removed so far. Given that only edges adjacent to a faulty node can be removed, each W_k represents a potentially set of no more than f faulty nodes that explains the removed edges. So if some node $i \in \bigcap_{k=1}^{K} W_k$, it must be faulty. This part is similar to the work in system-level diagnosis [24, 20].

H Throughput of Algorithm 4

Consider the time cost of each operation of Algorithm 4:

- `Step 1`: At most B/D^* per generation since the broadcast from the source node 1 at rate D^* is achievable, as previously discussed. So total cost is $L/B \times B/D^* = L/D^*$.
- `Step 2`: At most B/R^* per generation as discussed. So total cost is $L/B \times B/R^* = L/R^*$.
- `Step 2(a)`: The per-generation cost for broadcasting the output bits of MEQ with `Broadcast_Binary` is some constant α independent of B and L. So total cost is $\alpha L/B$.
- `Step 3`: Each time `Fault Diagnosis` is performed, at most βB bits are being broadcast with `Broadcast_Binary` for some constant β. So the time cost is γB for some constant γ independent of B and L. As discussed previously, diagnosis is performed at most $f(f+1)$ times. So the total cost is $f(f+1)\gamma B$.

Then we can computed the throughput of Algorithm 4 as

$$TPT = \frac{L}{\frac{L}{D^*} + \frac{L}{R^*} + \alpha\frac{L}{B} + f(f+1)\gamma B} = \frac{D^*R^*}{D^* + R^* + \alpha\frac{D^*R^*}{B} + f(f+1)\gamma\frac{D^*R^*B}{L}} \tag{21}$$

Notice that D^*, R^*, f, α and γ are independent of B and L. By choosing sufficiently large B and $L = \Omega(B)$, the last two terms in the denominator can be made arbitrarily close to 0. So TPT can be made arbitrarily close to

$$\frac{D^*R^*}{D^* + R^*}. \tag{22}$$

Recall that for each G_k, we have $U(G_k)/2 \le R_k \le U(G_k)$. Since $R^* = \min_{G_k} R_k$ and $U^* = \min_{G_k} U(G_k)$, it follows that $U^*/2 \le R^* \le U^*$. Also recall that $C_{BB}(G) \le \min(U^*, D^*)$, then

1. When $D^* \le R^*$: Observe that TPT is an increasing function of R^*. It is minimized when R^* is minimized. So

   $$TPT \ge \frac{D^{*2}}{D^* + D^*} = D^*/2 \ge C_{BB}(G)/2. \tag{23}$$

   The last inequality is due to $D^* \ge C_{BB}(G)$.

2. When $R^* < D^* \le U^*$:

   $$TPT = D^* \frac{R^*}{D^* + R^*} \ge D^* \frac{R^*}{U^* + R^*} \ge C_{BB}(G)/3. \tag{24}$$

   The last inequality is due to $D^* \ge C_{BB}(G)$ and $U^* \le 2R^*$.

3. When $R^* \le U^* < D^*$: Observe that TPT is an increasing function of D^*. It is minimized when D^* is minimized. So

   $$TPT \quad \ge \quad U^* \frac{R^*}{U^* + R^*} \ge C_{BB}(G)/3. \tag{25}$$

   The second inequality is due to $U^* \ge C_{BB}(G)$ and $U^* \le 2R^*$.

In the above discussion, we implicitly assumed that transmissions during the `Broadcast` stage accomplish all at the same time. However, in reality, it may require transmissions over multiple hops. A node cannot forward a message/packet until it receives it. So for each generation, the information broadcast by the source propagates only one hop every B/D^* time units. So for a large network, the "time span" of the `Broadcast` stage can be much larger than B/D^*. This problem can be solved by pipelining: We divide the time horizon into rounds of $B/D^* + B/R^*$ time units. For each generation g, $x(g)$ from the source node 1 propagates one hop per round, using the first B/D^* time units, until `Broadcast` completes. Then the remaining B/R^* time units of the last round is used to perform `Failure Detection`. An example in which `Broadcast` takes 3 hops is shown in Figure 3. By pipelining, we achieve the throughput computed above.

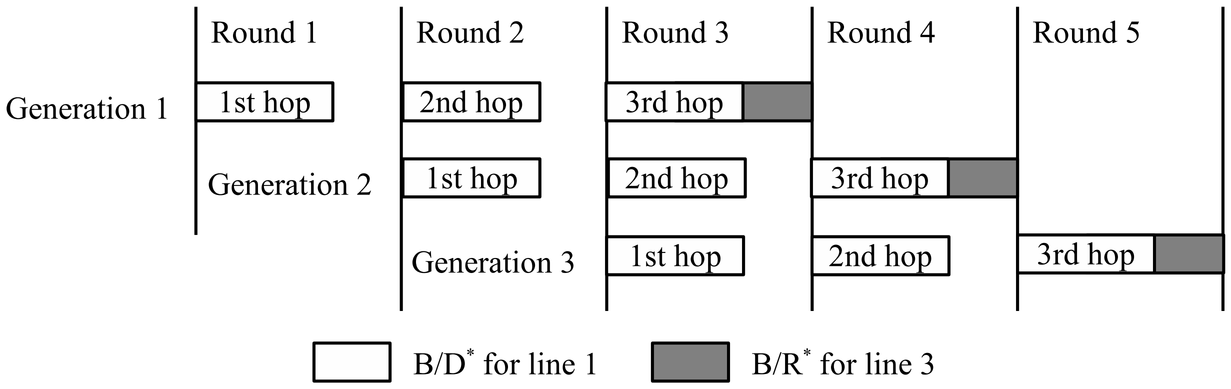


Figure 3: Example of pipelining