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

Guanfeng Liang
University of Illinois
Electrical and Computer Engineering
Urbana, Illinois
guanfeng.liang@gmail.com

Nitin H. Vaidya
University of Illinois
Electrical and Computer Engineering
Urbana, Illinois
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 *synchronous* point-to-point networks, where the rate of transmission over each communication link is limited by its "link capacity". The throughput of a particular BB algorithm is defined as the average number of bits that can be reliably broadcast to all fault-free nodes per unit time using the algorithm without violating the link capacity constraints. The *capacity* of BB in a given network is then defined as the supremum of all achievable BB throughputs in the given network, over all possible BB algorithms.

We develop NAB – a Network-Aware BB algorithm – for tolerating $f$ faults in arbitrary point-to-point networks consisting of $n \geq 3f + 1$ nodes and having $\geq 2f + 1$ directed node disjoint paths from each node $i$ to each node $j$. We also prove an upper bound on the capacity of BB, and conclude that NAB can achieve throughput at least 1/3 of the capacity. When the network satisfies an additional condition, NAB can achieve throughput at least 1/2 of the capacity. To the best of our knowledge, NAB is the first algorithm that can achieve a constant fraction of capacity of Byzantine Broadcast (BB) in general point-to-point networks.

## Categories and Subject Descriptors

C.2.4 [**Distributed Systems**]: Distributed applications

## General Terms

Algorithms, Theory

## Keywords

Broadcast, Byzantine faults, capacity, directed graph

## 1. INTRODUCTION

The problem of Byzantine Broadcast (BB) – also known as the Byzantine Generals problem [11] – was introduced by Pease, Shostak and Lamport in their 1980 paper [19]. 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 [20]. Informally, Byzantine Broadcast (BB) can be described as follows. 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 *synchronous* 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 [7, 6, 3, 10, 2, 18], performance of BB in *arbitrary* point-to-point network has not been investigated previously. When capacities of the different links are not identical, previously proposed algorithms can perform poorly. In fact, one can easily construct example networks in which previously proposed algorithms achieve throughput that is arbitrarily worse than the optimal throughput. Our prior work [13] introduces a BB algorithm that achieves the optimal throughput in 4-node networks with arbitrary link capacity constraints. But this does not apply to networks with > 4 nodes.

## Problem Formulation

We consider a *synchronous* system consisting of $n$ nodes, named $1, 2, \cdots, n$, with one node designated as the *sender* or *source* node. In particular, 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 following conditions must be satisfied:

**Termination:** Every fault-free node $i$ must eventually decide on an *output* value of $L$ bits; let us denote the output value of fault-free node $i$ as $y_i$.

[*]This research is supported in part by National Science Foundation award 1059540 and Army Research Office grant W-911-NF-0710287. 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.

*PODC'12,* July 16–18, 2012, Madeira, Portugal.
Copyright 2012 ACM 978-1-4503-1450-3/12/07 ...$10.00.

**Agreement:** All fault-free nodes must agree on an identical output value, i.e., there exists $y$ such that $y_i = y$ 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., $y = x$.

### *Failure Model*

The faulty nodes are controlled by an adversary that has a complete knowledge of the network topology, the algorithm, and the input value $x$. 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 incorrect or inconsistent messages to the neighbors. We assume that the set of faulty nodes remains *fixed* across different instances of execution of the BB algorithm. When a faulty node fails to send a message to a neighbor as required by the algorithm, we assume that the recipient node interprets the missing message as being some default value. We also assume that $f > 0$, since the case when $f = 0$ is trivial.

### *Network Model*

We assume a synchronous point-to-point network modeled as a directed simple graph $\mathcal{G}(\mathcal{V}, \mathcal{E})$, where the set of vertices $\mathcal{V} = \{1, 2, \cdots, n\}$ represents the nodes in the network, and the set of edges $\mathcal{E}$ represents the links in the network. With a slight abuse of terminology, we will use the terms *edge* and *link*, and *node* and *vertex*, interchangeably. We assume that $n \geq 3f + 1$ since it is necessary for the existence of a correct BB algorithm [7]. We also require that there exist $\geq 2f + 1$ directed node disjoint paths from each node $i$ to each node $j$ in the network.

In the given network, links may not exist between all node pairs. For each directed link $e = (i, j) \in \mathcal{E}$, its *capacity*, denoted as $z_e$ specifies the maximum amount of information that can be transmitted on that link per unit time. Specifically, we assume that up to $z_e\tau$ bits can be reliably sent from node $i$ to node $j$ over time duration $\tau$ (for any non-negative $\tau$). This is a deterministic model of *capacity* that has been commonly used in other work [12, 4, 8, 9]. All link capacities are assumed to be positive integers.[1] 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.

### *Throughput and Capacity of BB*

When defining the throughput of a given BB algorithm in a given network, we consider $Q$ independent instances of BB. The source node is given an $L$-bit input for each of these $Q$ instances, and the *validity* and *agreement* properties need to be satisfied for each instance *separately* (i.e., independent of the outcome for the other instances).

For any BB algorithm $\mathcal{A}$, denote $t(\mathcal{G}, L, Q, \mathcal{A})$ as the duration of time required, in the worst case, to complete $Q$ instances of $L$-bit Byzantine Broadcast, without violating the capacity constraints of the links in $\mathcal{G}$. Throughput of algorithm $\mathcal{A}$ in network $\mathcal{G}$ for $L$-bit inputs is then defined as

$$T(\mathcal{G}, L, \mathcal{A}) = \lim_{Q\to\infty} \frac{LQ}{t(\mathcal{G}, L, Q, \mathcal{A})}.$$

We then define capacity $C_{BB}$ as follows.

Capacity $C_{BB}$ of Byzantine Broadcast in network $\mathcal{G}$ is defined as the supremum over the throughput of all algorithms $\mathcal{A}$ that solve the BB problem and all values of $L$. That is,

$$C_{BB}(\mathcal{G}) \ = \ \sup_{\mathcal{A},L} T(\mathcal{G}, L, \mathcal{A}).$$

## 2. ALGORITHM OVERVIEW

This section provides an overview of the structure of NAB – a Network-Aware Byzantine broadcast algorithm – for arbitrary point-to-point networks. Each instance of our NAB algorithm performs Byzantine broadcast of an $L$-bit value. We assume that the NAB algorithm is used repeatedly, and during all these repeated executions, the cumulative number of distinct faulty nodes is upper bounded by $f$. Due to this assumption, the algorithm can perform well by amortizing the cost of fault tolerance over a large number of executions. Larger values of $L$ also result in better performance for the algorithm. The algorithm is intended to be used for sufficiently large $L$ and $Q$, to be elaborated later in Section 5.

The $k$-th instance of NAB executes on a network corresponding to graph $\mathcal{G}_k(\mathcal{V}_k, \mathcal{E}_k)$, defined as follows:

- For the first instance, $k = 1$, and $\mathcal{G}_1 = \mathcal{G}$. Thus, $\mathcal{V}_1 = \mathcal{V}$ and $\mathcal{E}_1 = \mathcal{E}$.
- The $k$-th instance of NAB occurs on graph $\mathcal{G}_k$ in the following sense: (i) all the fault-free nodes know the node and edge sets $\mathcal{V}_k$ and $\mathcal{E}_k$, (ii) only the nodes corresponding to the vertices in $\mathcal{V}_k$ need to participate in the $k$-th instance of BB, and (iii) only the links corresponding to the edges in $\mathcal{E}_k$ are used for communication in the $k$-th instance of NAB (communication received on other links is ignored).
- During the $k$-th instance of NAB using graph $\mathcal{G}_k$, if misbehavior by some faulty node(s) is detected, then, as described later, additional information is gleaned about the potential identity of the faulty node(s). In this case, $\mathcal{G}_{k+1}$ is obtained by removing from $\mathcal{G}_k$ appropriately chosen edges and possibly some vertices, based on dispute control [1].
  On the other hand, if during the $k$-th instance, no misbehavior is detected, then $\mathcal{G}_{k+1} = \mathcal{G}_k$.

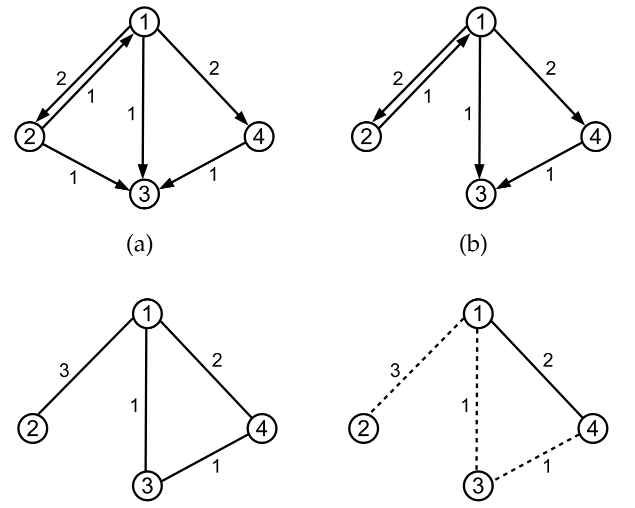
The $k$-th instance of NAB algorithm consists of three phases, as described next. The main contributions of this paper are (i) the algorithm used in Phase 2 below, and (ii) a performance analysis of NAB.

If graph $\mathcal{G}_k$ does *not* contain the source node 1, then (as will be clearer later) by the start of the $k$-th instance of NAB, all the fault-free nodes already know that the source node is surely faulty; in this case, the fault-free nodes can agree on a default value for the output, and terminate the algorithm. Hereafter, we will assume that the source node 1 is in $\mathcal{G}_k$.

## Phase 1: Unreliable Broadcast

In Phase 1, source node 1 broadcasts $L$ bits to all the other nodes in $\mathcal{G}_k$. This phase makes no effort to detect or tolerate

[1] 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 long time unit.



(c) Undirected graph corresponding to Figure 1(b)

(d) A spanning tree in the undirected graph of Figure 1(c) shown in dotted edges

**Figure 1: Example graphs. Numbers next to the edges indicate link capacities.**

misbehavior by faulty nodes. Now let us analyze the time required to perform unreliable broadcast in Phase 1.

*MINCUT*($\mathcal{G}_k$, 1, $j$) denotes the minimum cut in the directed graph $\mathcal{G}_k$ from source node 1 to node $j$. Let us define

$$\gamma_k = \min_{j \in \mathcal{V}_k} MINCUT(\mathcal{G}_k, 1, j).$$

*MINCUT*($\mathcal{G}_k$, 1, $j$) is equal to the maximum flow rate possible from node 1 to node $j \in \mathcal{V}_k$. It is well-known [17] that $\gamma_k$ is the maximum rate achievable for broadcast from node 1 to *all* the other nodes in $\mathcal{V}_k$, under the capacity constraints on the links in $\mathcal{E}_k$ (this can be achieved using $\gamma_k$ unit-capacity spanning trees embedded in $\mathcal{G}_k$ [15]). Thus, the least amount of time in which $L$ bits can be broadcast by node 1 in graph $\mathcal{G}_k$ is given by $L/\gamma_k$. To simplify the analysis, we ignore propagation delays. Analogous results can be obtained in the presence of propagation delays as well [12].

Clearly, $\gamma_k$ depends on the capacities of the links in $\mathcal{G}_k$. For example, if $\mathcal{G}_k$ were the directed graph in Figure 1(a), then $MINCUT(\mathcal{G}_k, 1, 2) = 2$, $MINCUT(\mathcal{G}_k, 1, 3) = 3$, and $MINCUT(\mathcal{G}_k, 1, 4) = 2$; hence $\gamma_k = 2$.

At the end of the broadcast operation in Phase 1 of the $k$-th instance of NAB, each node should have received $L$ bits. One of the following four outcomes will occur:

(i) The source node 1 is fault-free, and all the fault-free nodes correctly receive the source node's $L$-bit input for the $k$-th instance of NAB, or

(ii) The source node 1 is fault-free, but some of the fault-free nodes receive incorrect $L$-bit values due to misbehavior by some faulty node(s), or

(iii) The source node 1 is faulty, but all the fault-free nodes still receive an identical $L$-bit value in Phase 1, or

(iv) The source node is faulty, and all the fault-free nodes do not receive an identical $L$-bit value in Phase 1.

The values received by the fault-free nodes in cases (i) and (iii) satisfy the *agreement* and *validity* conditions, whereas in cases (ii) and (iv) at least one of the two conditions is violated.

## Phase 2: Failure Detection

Phase 2 performs the following two operations. As stipulated in the fault model, a faulty node may not follow the algorithm specification correctly.

**(Step 2.1) Equality check:** Using an *Equality Check* algorithm, the nodes in $\mathcal{V}_k$ perform a comparison of the $L$-bit value they received in Phase 1, to determine if all the nodes received an identical value. The source node 1 also participates in this comparison operation (treating its input as the value "received from" itself).

Section 3 presents the Equality Check algorithm, which is designed to *guarantee* that if the values received by the fault-free nodes in Phase 1 are *not identical*, then at least one fault-free node will detect the *mismatch*.

**(Step 2.2) Agreeing on the outcome of equality check:** Using a previously proposed Byzantine broadcast algorithm, such as [19], each node performs Byzantine broadcast of a 1-bit *flag* to other nodes in $\mathcal{G}_k$ indicating whether it detected a mismatch during equality check.

If any node broadcasts in step 2.2 that it has detected a mismatch, then subsequently *Phase 3 is performed*. On the other hand, if no node announces a mismatch in step 2.2 above, then *Phase 3 is not performed*; in this case, each fault-free node agrees on the value it received in Phase 1, and the $k$-th instance of *NAB is completed*.

We will later prove that, when Phase 3 is *not* performed, the values agreed above by the fault-free nodes satisfy the *validity* and *agreement* conditions for the $k$-th instance of NAB. On the other hand, when Phase 3 is performed during the $k$-th instance of NAB, as noted below, Phase 3 results in correct outcome for the $k$-th instance. When Phase 3 is performed, Phase 3 determines $\mathcal{G}_{k+1}$; otherwise, $\mathcal{G}_{k+1} = \mathcal{G}_k$.

## Phase 3: Dispute Control

Phase 3 employs a *dispute control* mechanism that has also been used in prior work [1, 14]. Appendix A provides the details of the dispute control algorithm used in Phase 3. Here we summarize the outcomes of this phase – this summary should suffice for understanding the main contributions of this paper.

The dispute control in Phase 3 has very high overhead, due to the large amount of data that needs to be transmitted. From the above discussion of Phase 2, it follows that Phase 3 is performed *only if* at least one faulty node misbehaves during Phases 1 or 2. The outcomes from Phase 3 performed during the $k$-th instance of NAB are as follows.

**Outcome 1:** Phase 3 results in correct Byzantine broadcast for the $k$-th instance of NAB. This is obtained as a byproduct of the dispute control mechanism.

**Outcome 2:** By the end of Phase 3, either one of the nodes in $\mathcal{V}_k$ is correctly identified as faulty, or/and at least one pair of nodes in $\mathcal{V}_k$, say nodes $a, b$, is identified as being "in dispute" with each other. When a node pair $a, b$ is found *in dispute*, it is guaranteed that (i) *at least* one of these two nodes is faulty, and (ii) at least one of the directed edges $(a, b)$ and $(b, a)$ is in $\mathcal{E}_k$. Note that the dispute control phase *never* finds two fault-free nodes in dispute with each other.

**Outcome 3:** Phase 3 in the $k$-th instance computes graph $\mathcal{G}_{k+1}$. In particular, any nodes that can be inferred as being faulty based on their behavior so far are excluded from $\mathcal{V}_{k+1}$; links attached to such nodes are excluded from $\mathcal{E}_{k+1}$. In Appendix A we elaborate on how the faulty nodes are identified. Then, for each node pair in $\mathcal{V}_{k+1}$, if that node pair has been found in dispute at least in one instance of NAB so far, the links between the node pair are excluded from $\mathcal{E}_{k+1}$. Phase 3 ensures that all the fault-free nodes compute an identical graph $\mathcal{G}_{k+1} = (\mathcal{V}_{k+1}, \mathcal{E}_{k+1})$ to be used during the next instance of NAB.

Consider two special cases for the $k$-th instance of NAB:

**Case 1:** If graph $\mathcal{G}_k$ does not contain the source node 1, it implies that all the fault-free nodes are aware that node 1 is faulty. In this case, they can safely agree on a default value as the outcome for the $k$-th instance of NAB.

**Case 2:** Similarly, if the source node is in $\mathcal{G}_k$ but at least $f$ other nodes are excluded from $\mathcal{G}_k$, that implies that the remaining nodes in $\mathcal{G}_k$ are all fault-free; in this case, algorithm NAB can be reduced to just Phase 1.

Observe that during each execution of Phase 3, either a new pair of nodes *in dispute* is identified, or a new node is identified as faulty. Once a node is found to be in dispute with $f + 1$ distinct nodes, it can be identified as faulty, and excluded from the algorithm's execution. Therefore, dispute control needs to be performed at most $f(f + 1)$ times over repeated executions of NAB. Thus, even though each dispute control phase is expensive, the bounded number ensures that the amortized cost over a large number of instances of NAB is small, as reflected in the performance analysis of NAB (in Section 5 and Appendix C).

# 3. EQUALITY CHECK ALGORITHM WITH PARAMETER $\rho_K$

We now present the Equality Check algorithm (Algorithm 1 below) used in Phase 2, which has an integer parameter $\rho_k$ for the $k$-th instance of NAB. Later in this section, we will elaborate on the choice of $\rho_k$, which is dependent on capacities of the links in $\mathcal{G}_k$.

Let us denote by $x_i$ the $L$-bit value received by fault-free node $i \in \mathcal{V}_k$ in Phase 1 of the $k$-th instance. For simplicity, we do not include index $k$ in the notation $x_i$. To simplify the presentation, let us assume that $L/\rho_k$ is an integer. Thus we can represent the $L$-bit value $x_i$ as $\rho_k$ symbols from Galois Field $GF(2^{L/\rho_k})$. In particular, we represent $x_i$ as a vector $\mathbf{X_i} = [\mathbf{X_i(1)}, \mathbf{X_i(2)}, \cdots, \mathbf{X_i(\rho_k)}]$, where each symbol $X_i(j) \in GF(2^{L/\rho_k})$ can be represented using $L/\rho_k$ bits. As discussed earlier, for convenience, we assume that all the link capacities are integers.

In the Equality Check algorithm, $z_e$ symbols of size $L/\rho_k$ bits are transmitted on each link $e$ of capacity $z_e$. Therefore, the Equality Check algorithm requires time duration $L/\rho_k$.

## 3.1 Salient Feature of the Algorithm

In the Equality Check algorithm, a single round of communication occurs between adjacent nodes. No node is required to forward packets received from other nodes during the algorithm. This implies that, while a faulty node may send incorrect packets to its neighbors, it cannot tamper information sent between fault-free nodes. This feature of Equality Check is important in being able to prove its correctness despite the presence of faulty nodes in $\mathcal{G}_k$.

## 3.2 Choice of Parameter $\rho_k$

**Algorithm 1** Equality Check in $\mathcal{G}_k$ with parameter $\rho_k$

```
Each node $i \in \mathcal{V}_k$ should performs these steps:
  1. On each outgoing link $e = (i, j) \in \mathcal{E}_k$ whose capacity
     is $z_e$, node $i$ transmits $z_e$ linear combinations of the $\rho_k$
     symbols in vector $\mathbf{X_i}$, with the weights for the linear
     combinations being chosen from $GF(2^{L/\rho_k})$.
     More formally, for each outgoing edge $e = (i, j) \in \mathcal{E}_k$ of
     capacity $z_e$, a $\rho_k \times z_e$ matrix $\mathbf{C_e}$ is specified as a part of
     the algorithm. Entries in $\mathbf{C_e}$ are chosen from $GF(2^{L/\rho_k})$.
     Node $i$ sends to node $j$ a vector $\mathbf{Y_e}$ of $z_e$ symbols obtained
     as the matrix product $\mathbf{Y_e} = \mathbf{X_i C_e}$. Each element of $\mathbf{Y_e}$ is
     said to be a "coded symbol". The choice of the matrix
     $\mathbf{C_e}$ affects the correctness of the algorithm, as elaborated
     later.
  2. On each incoming edge $d = (j, i) \in \mathcal{E}_k$, node $i$ receives a
     vector $\mathbf{Y_d}$ containing $z_d$ symbols from $GF(2^{L/\rho_k})$. Node
     $i$ then checks, for each incoming edge $d$, whether $\mathbf{Y_d} =$
     $\mathbf{X_i C_d}$. The check is said to fail iff $\mathbf{Y_d} \neq \mathbf{X_i C_d}$.
  3. If checks of symbols received on any incoming edge fail
     in the previous step, then node $i$ sets a 1-bit flag equal
     to MISMATCH; else the flag is set to NULL. This flag is
     broadcast in Step 2.2 in Phase 2 above.
```

We define a set $\Omega_k$ as follows using the disputes identified through the first $(k - 1)$ instances of NAB.

$$\Omega_k = \{ H \mid H \text{ is a subgraph of } \mathcal{G}_k \text{ containing } (n-f) \text{ nodes such that no two nodes in } H \text{ have been found } \textit{in dispute} \text{ through the first } (k-1) \text{ instances} \}$$

As noted in the discussion of Phase 3 (Dispute Control), fault-free nodes are never found in dispute *with each other* (fault-free nodes may be found in dispute with faulty nodes, however). This implies that $\mathcal{G}_k$ includes all the fault-free nodes. There are at least $n - f$ fault-free nodes in the network. This implies that set $\Omega_k$ is non-empty.

Corresponding to a directed graph $H(V, E)$, let us define an *undirected* graph $\overline{H}(V, \overline{E})$ as follows: (i) both $H$ and $\overline{H}$ contain the same set of vertices, (ii) undirected edge $(i, j) \in \overline{E}$ if either $(i, j) \in E$ or $(j, i) \in E$, and (iii) capacity of undirected edge $(i, j) \in \overline{E}$ is defined to be equal to the sum of the capacities of directed links $(i, j)$ and $(j, i)$ in $E$ (if a directed link does not exist in $E$, here we treat its capacity as 0). For example, Figure 1(c) shows the undirected graph corresponding to the directed graph in Figure 1(b).

Define a set of undirected graphs $\overline{\Omega}_k$ as follows. $\overline{\Omega}_k$ contains undirected version of each directed graph in $\Omega$: $\overline{\Omega}_k = \{\overline{H} | H \in \Omega_k\}$. Define

$$U_k = \min_{\overline{H} \in \overline{\Omega}_k} \min_{i,j \in \overline{H}} MINCUT(\overline{H}, i, j)$$

as the minimum value of the *undirected* MINCUTs between all pairs of nodes in all the undirected graphs in the set $\overline{\Omega}_k$. For instance, suppose that $n = 4$, $f = 1$ and the graph shown in Figure 1(a) is $\mathcal{G}$, whereas $\mathcal{G}_k$ is the graph shown in Figure 1(b). Thus, nodes 2 and 3 have been found in dispute previously. Then, $\Omega_k$ and $\overline{\Omega}_k$ each contain two subgraphs, one subgraph corresponding to the node set $\{1, 2, 4\}$, and the other subgraph corresponding to the node set $\{1, 3, 4\}$. In this example, $U_k = 2$.

Parameter $\rho_k$ is chosen such that

$$\rho_k \leq \frac{U_k}{2}.$$

Under such constraint on $\rho_k$, we will prove the correctness of the Equality Check algorithm, with its execution time being $L/\rho_k$.

## 3.3 Correctness of Equality Check

The correctness of Algorithm 1 depends on the choices of the parameter $\rho_k$ and the set of coding matrices $\{\mathbf{C_e} | e \in \mathcal{E}_k\}$. Let us say that a set of coding matrices is *correct* if the resulting Equality Check (Algorithm 1) satisfies the following requirement:

**(EC) if** there exists a pair of fault-free nodes $i, j \in \mathcal{G}_k$ such that $\mathbf{X_i} \neq \mathbf{X_j}$ (i.e., $x_i \neq x_j$),
**then** the 1-bit flag at *at least one* fault-free node is set to MISMATCH.

Recall that $\mathbf{X_i}$ is a vector representation of the $L$-bit value $x_i$ received by node $i$ in Phase 1 of NAB. Two consequences of the above correctness condition are:
**Consequence 1:** If some node (possibly the source node) misbehaves during Phase 1 leading to outcomes (ii) or (iv) for Phase 1, then at least one fault-free node will set its flag to MISMATCH. In this case, the fault-free nodes (possibly including the sender) do not share identical $L$-bit values $\mathbf{X_i}$'s as the outcome of Phase 1.
**Consequence 2:** If no misbehavior occurs in Phase 1 (thus the values received by fault-free nodes in Phase 1 are correct), but MISMATCH flag at some fault-free node is set in *Equality Check*, then misbehavior must have occurred in Phase 2.

The following theorem shows that when $\rho_k \leq U_k/2$, and when $L$ is sufficiently large, there exists a set of coding matrices $\{\mathbf{C_e} | e \in \mathcal{E}_k\}$ that are correct.

Theorem 1. *For $\rho_k \leq U_k/2$, if the entries of the coding matrices $\{\mathbf{C_e} | e \in \mathcal{E}_k\}$ in step 1 of Algorithm 1 are chosen independently and uniformly at random from $GF(2^{L/\rho_k})$, then $\{\mathbf{C_e} | e \in \mathcal{E}_k\}$ is correct with probability $\geq 1 - 2^{-L/\rho_k}\left[\binom{n}{n-f}(n-f-1)\rho_k\right]$. Note that when $L$ is large enough, $1 - 2^{-L/\rho_k}\left[\binom{n}{n-f}(n-f-1)\rho_k\right] > 0$.*

**Proof sketch:** The complete proof of Theorem 1 is presented in Appendix B. Our goal is to prove that property (EC) above holds with a non-zero probability. That is, regardless of which (up to $f$) nodes in $\mathcal{G}$ are faulty and what values $\mathbf{X_i}$'s equal to, whenever $\mathbf{X_i} \neq \mathbf{X_j}$ for some pair of fault-free nodes $i$ and $j$ in $\mathcal{G}_k$ during the $k$-th instance, at least one fault-free node (which may be different from nodes $i$ and $j$) will set its 1-bit flag to MISMATCH. To prove this, we consider every subgraph of $H \in \Omega_k$ (see definition of $\Omega_k$ above). By definition of $\Omega_k$, no two nodes in $H$ have been found in dispute through the first $(k-1)$ instances of NAB. Therefore, $H$ represents one *potential* set of $n-f$ fault-free nodes in $\mathcal{G}_k$. For each edge $e = (i, j)$ in $H$, steps 1-2 of Algorithm 1 together have the effect of checking whether or not $(\mathbf{X_i} - \mathbf{X_j})\mathbf{C_e} = \mathbf{0}$. Without loss of generality, for the purpose of this proof, rename the nodes in $H$ as $1, \cdots, n-f$. Denote $\mathbf{D_i} = \mathbf{X_i} - \mathbf{X_{n-f}}$ for $i = 1, \cdots, (n-f-1)$, then

$$(\mathbf{X_i} - \mathbf{X_j})\mathbf{C_e} = \mathbf{0} \Leftrightarrow \begin{cases} (\mathbf{D_i} - \mathbf{D_j})\mathbf{C_e} = \mathbf{0} & , \quad \text{if } i, j < n-f; \\ \mathbf{D_i}\mathbf{C_e} = \mathbf{0} & , \quad \text{if } j = n-f; \\ -\mathbf{D_j}\mathbf{C_e} = \mathbf{0} & , \quad \text{if } i = n-f. \end{cases}$$

Define $\mathbf{D_H} = [\mathbf{D_1}, \mathbf{D_2}, \cdots, \mathbf{D_{n-f-1}}]$. Let $m$ be the sum of the capacities of all the directed edges in $H$. As elaborated in Appendix B, we define $\mathbf{C_H}$ to be a $(n-f-1)\rho_k \times m$ matrix whose entries are obtained using the elements of $\mathbf{C_e}$ for each edge $e$ in $H$ in an appropriate manner. For the suitably defined $\mathbf{C_H}$ matrix, we can show that the comparisons in steps 1-2 of Algorithm 1 at *all* the nodes in $H \in \Omega_k$ are equivalent to checking whether or not $\mathbf{D_H}\,\mathbf{C_H} = \mathbf{0}$.

We show that for a particular subgraph $H \in \Omega_k$, when $\rho_k \leq U_k/2$, $m \geq (n-f-1)\rho_k$ and $L$ is large enough, with non-zero probability $\mathbf{C_H}$ contains a $(n-f-1)\rho_k \times (n-f-1)\rho_k$ invertible submatrix if the set of coding matrices $\{\mathbf{C_e} | e \in \mathcal{E}_k\}$ are generated randomly as described in Theorem 1. In this case $\mathbf{D_H}\mathbf{C_H} = \mathbf{0}$ if and only if $\mathbf{D_H} = \mathbf{0}$, i.e., $\mathbf{X_1} = \mathbf{X_2} = \cdots = \mathbf{X_{n-f}}$. In other words, if all nodes in subgraph $H$ are fault-free, and $\mathbf{X_i} \neq \mathbf{X_j}$ for two fault-free nodes $i, j$, then $\mathbf{D_H}\mathbf{C_H} \neq \mathbf{0}$ and hence the check in step 2 of Algorithm 1 fails at some fault-free node in $H$.

We then further show that, for large enough $L$, with a non-zero probability, this is also *simultaneously* true for all subgraphs $H \in \Omega_k$. This implies that, for large enough $L$, correct coding matrices ($\mathbf{C_e}$ for each $e \in \mathcal{E}_k$) can be found. Notice that for a given network, the correctness of a set of coding matrices is *independent* of the values of $x_i$'s. This set of matrices are specified as a part of the algorithm specification.

# 4. CORRECTNESS OF NAB

For Phase 1 (Unreliable Broadcast) and Phase 3 (Dispute Control), the proof that the outcomes claimed in Section 2 indeed follows directly from the prior literature cited in Section 2 (and elaborated in Appendix A). Now consider two cases:

**Case 1 –** The values received by the fault-free nodes in Phase 1 are *not identical*: Then the correctness of Equality Check ensures that a fault-free node will detect the mismatch, and consequently Phase 3 will be performed. As a byproduct of Dispute Control in Phase 3, the fault-free nodes will correctly agree on a value that satisfies the *validity* and *agreement* conditions.
**Case 2 –** The values received by the fault-free nodes in Phase 1 are identical: If no node announces a mismatch in step 2.2, then the fault-free nodes will agree on the value received in Phase 1. It is easy to see that this is a correct outcome. On the other hand, if some node announces a mismatch in step 2, then Dispute Control will be performed, which will result in correct outcome for the broadcast of the $k$-th instance. Thus, in all cases, NAB will lead to correct outcome in each instance.

# 5. THROUGHPUT AND CAPACITY

## 5.1 Throughput of NAB for Large $L$ and $Q$

In this section, we present a lower bound on the achievable throughput with NAB when the input size $L$ for each instance and the number of instances $Q$ are both "large" (in an order sense) compared to $n$. Complete proof can be found in Appendix C. Two consequences of $L$ and $Q$ being large:

$L$ **being large** ($\omega(n^\alpha)$ for some constant $\alpha > 0$): the overhead of 1-bit broadcasts performed in step 2.2 of Phase 2 becomes negligible when amortized over the $L$ bits being broadcast by the source in each instance of NAB.
$Q$ **being large** ($\omega(n^{\beta+2})$ for some constant $\beta > 0$): the average overhead of dispute control per instance of NAB becomes negligible. Recall that dispute control needs to be performed at most $f(f+1)$ times over $Q$ executions of NAB.

It then suffices to consider only the time it takes to complete the Unreliable Broadcast in Phase 1 and Equality Check in Phase 2. For the $k$-th instance of NAB, as discussed previously, the unreliable broadcast in Phase 1 can be done in $L/\gamma_k$ time units (see definition of $\gamma_k$ in Section 2). We now define

$$\begin{aligned}\Gamma \;=\; \{\, H \mid\ & H \text{ is a subgraph of } \mathcal{G} \text{ containing source node 1} \\ & \text{such that } \mathcal{G}_k \text{ may equal } H \text{ in some execution of} \\ & \text{NAB for some } k \,\}\end{aligned}$$

Appendix D provides a systematic construction of the set $\Gamma$. Define the minimum value of all possible $\gamma_k$:

$$\gamma^* = \min_{\mathcal{G}_k\in\Gamma}\gamma_k = \min_{\mathcal{G}_k\in\Gamma}\ \min_{j\in\mathcal{V}_k} MINCUT(\mathcal{G}_k, 1, j).$$

Then an upper bound of the execution time of Phase 1 in all instances of NAB is $L/\gamma^*$.

With parameter $\rho_k = U_k/2$, the execution time of the Equality Check in Phase 2 is $L/\rho_k$. Recall that $U_k$ is defined as the minimum value of the MINCUTs between all pairs of nodes in all undirected graphs in the set $\overline{\Omega}_k$. As discussed in Appendix B.2, $\overline{\Omega}_k \subseteq \overline{\Omega}_1$, where $\mathcal{G}_1 = \mathcal{G}$. Hence $U_k \geq U_1$ in all possible $\mathcal{G}_k$. Define

$$\rho^* = \frac{U_1}{2} = \min_{\overline{H}\in\Omega_1}\ \min_{nodes\ i,j\ in\ \overline{H}} MINCUT(\overline{H}, i, j).$$

Then $\rho_k \geq \rho^*$ for all possible $\mathcal{G}_k$ and the execution time of the Equality Check is upper-bounded by $L/\rho^*$. So the throughput of NAB for large $Q$ and $L$ can be lower bounded by

$$\lim_{L\to\infty} T(\mathcal{G}, L, NAB) \geq \frac{L}{L/\gamma^* + L/\rho^*} = \frac{\gamma^*\rho^*}{\gamma^* + \rho^*}. \qquad (1)$$

To simplify the discussion above, we ignored propagation delays. Appendix C also describes how to approach this bound even when propagation delays are considered.

## 5.2 An Upper Bound on Capacity of BB

We prove (in Appendix E) the following upper bound of the capacity of BB:

Theorem 2. *In point-to-point network $\mathcal{G}(\mathcal{V}, \mathcal{E})$, the capacity of Byzantine broadcast ($C_{BB}$) with node 1 as the source satisfies the following upper bound: $C_{BB}(\mathcal{G}) \leq \min(\gamma^*, 2\rho^*)$.*

Given the throughput lower bound $T_{NAB}(\mathcal{G})$ in (Eq.1) and the upper bound on $C_{BB}(\mathcal{G})$ from Theorem 2, the result below can be obtained through simple calculation. Readers are referred to Appendix G of our report [15] for the proof.

Theorem 3. *In point-to-point network $\mathcal{G}(\mathcal{V}, \mathcal{E})$:*

$$\lim_{L\to\infty} T(\mathcal{G}, L, NAB) \geq \min(\gamma^*, 2\rho^*)/3 \geq C_{BB}(\mathcal{G})/3.$$

*Moreover, when $\gamma^* \leq \rho^*$:*

$$\lim_{L\to\infty} T(\mathcal{G}, L, NAB) \geq \min(\gamma^*, 2\rho^*)/2 \geq C_{BB}(\mathcal{G})/2.$$

# 6. CONCLUSION

This paper presents NAB, a network-aware Byzantine broadcast algorithm for point-to-point networks. We derive an upper bound on the capacity of Byzantine broadcast, and show that NAB can asymptotically achieve throughput at least 1/3 fraction of the capacity over a large number of execution instances, when $L$ is large. The fraction can be improved to at least 1/2 when the network satisfies an additional condition.

# 7. REFERENCES

[1] Z. Beerliova-Trubiniova and M. Hirt. Efficient multi-party computation with dispute control. In *IACR Theory of Cryptography Conference (TCC)*, 2006.

[2] Z. Beerliova-Trubiniova and M. Hirt. Perfectly-secure mpc with linear communication complexity. In *IACR Theory of Cryptography Conference (TCC)*, 2008.

[3] P. Berman, J. A. Garay, and K. J. Perry. Bit optimal distributed consensus. In *Computer science*. Plenum Press, 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 *USENIX Symposium on Operating Systems Design and Implementation (OSDI)*, 1999.

[6] B. A. Coan and J. L. Welch. Modular construction of a Byzantine agreement protocol with optimal message bit complexity. *Journal of Information and Computation*, 1992.

[7] M. J. Fischer, N. A. Lynch, and M. Merritt. Easy impossibility proofs for distributed consensus problems. In *ACM symposium on Principles of Distributed Computing (PODC)*, 1985.

[8] T. Ho, B. Leong, R. Koetter, M. Medard, M. Effros, and D. Karger. Byzantine modification detection in multicast networks using randomized network coding (extended version). Technical report, (http://www.its.caltech.edu/ tho/multicast.ps), 2004.

[9] S. Jaggi, M. Langberg, S. Katti, T. Ho, D. Katabi, and M. Medard. Resilient network coding in the presence of Byzantine adversaries. In *IEEE International Conference on Computer Communications (INFOCOM)*, 2007.

[10] V. King and J. Saia. Breaking the $O(n^2)$ bit barrier: Scalable Byzantine agreement with an adaptive adversary. In *ACM symposium on Principles of Distributed Computing (PODC)*, 2010.

[11] L. Lamport, R. Shostak, and M. Pease. The Byzantine generals problem. *ACM Transaction on Programming Languages and Systems*, 1982.

[12] S.-Y. Li, R. Yeung, and N. Cai. Linear network coding. *IEEE Transactions on Information Theory*, 2003.

[13] G. Liang and N. Vaidya. Capacity of Byzantine agreement with finite link capacity. In *IEEE International Conference on Computer Communications (INFOCOM)*, 2011.

[14] G. Liang and N. Vaidya. Error-free multi-valued consensus with Byzantine failures. In *ACM Symposium on Principles of Distributed Computing (PODC)*, 2011.

[15] G. Liang and N. Vaidya. Byzantine broadcast in point-to-point networks using local linear coding. Technical report, arXiv (http://arxiv.org/abs/1106.1845), June 2011 (revised May 2012).

[16] E. M. Palmer. On the spanning tree packing number of a graph: a survey. *Journal of Discrete Mathematics*, 2001.

[17] C. H. Papadimitriou and K. Steiglitz. *Combinatorial Optimization: Algorithms and Complexity*. Courier Dover Publications, 1998.

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

[19] M. Pease, R. Shostak, and L. Lamport. Reaching

agreement in the presence of faults. *Journal of the ACM (JACM)*, 1980.

[20] A. Silberschatz, P. B. Galvin, and G. Gagne. *Operating System Concepts*, chapter 17 Distributed File Systems. Addison-Wesley, 1994.

# APPENDIX

Some details are omitted for the sack of space. Please see [15] for the complete proofs.

## A. DISPUTE CONTROL

The dispute control algorithm motivated by the work in [1] is performed in the $k$-th instance of NAB only if at least one node misbehaves during Phases 1 or 2. The goal of dispute control is to learn some information about the identity of at least one faulty node. In particular, the dispute control algorithm will identify a new node as being faulty, or/and identify a new node pair in dispute (at least one of the nodes in the pair is guaranteed to be faulty). The steps in dispute control in the $k$-th instance of NAB are as follows:

**(DC1)** Each node $i$ in $\mathcal{V}_k$ uses a previously proposed Byzantine broadcast algorithm, such as [6] (extended to use $2f+1$ node disjoint paths; see Appendix C), to broadcast to all other nodes in $\mathcal{V}_k$ all the messages that this node $i$ claims to have received from other nodes, and sent to the other nodes, during Phases 1 and 2 of the $k$-th instance. Source node 1 also uses an existing Byzantine broadcast algorithm [6] to broadcast its $L$-bit input for the $k$-th instance to all the other nodes. Thus, at the end of this step, all the fault-free nodes will reach correct agreement for the output for the $k$-th instance.

**(DC2)** If for some node pair $a, b \in \mathcal{V}_k$, a message that node $a$ claims above to have sent to node $b$ mismatches with the claim of received messages made by node $b$, then node pair $a, b$ is found in dispute. In step DC1, since a Byzantine broadcast algorithm is used to disseminate the claims, all the fault-free nodes will identify identical node pairs in dispute.

It should be clear that a pair of fault-free nodes will never be found in dispute with each other in this step.

**(DC3)** The NAB algorithm is deterministic in nature. Therefore, the messages that should be sent by each node in Phases 1 and 2 can be completely determined by the messages that the node receives, and, in case of node 1, its initial input. Thus, if the claims of the messages sent by some node $i$ are inconsistent with the messages it claims to have received, and its initial input (in case of node 1), then that node $i$ must be faulty. Again, all fault-free nodes identify these faulty nodes identically. Any nodes thus identified as faulty until now (including all previous instances of NAB) are deemed to be "in dispute" with all their neighbors (to whom the faulty nodes have incoming or outgoing links).

It should be clear that a fault-free node will never be found to be faulty in this step.

**(DC4)** Consider the node pairs that have been identified as being in dispute in DC2 and DC3 of at least one instances of NAB so far. We will say that a set of nodes $F_i$, where $|F_i| \leq f$, "explains" all the disputes so far, if for each pair $a, b$ found in dispute so far, at least one of $a$ and $b$ is in $F_i$.

It should be easy to see that for any set of disputes that may be observed, there must be at least one such set that *explains* the disputes. It is easy to argue that the nodes in the set intersection $\bigcap_{\delta=1}^{\Delta} F_\delta$ must be necessarily faulty (in fact, the nodes in the set intersection are also guaranteed to include nodes identified as faulty in step DC3).

Then, $\mathcal{V}_{k+1}$ is obtained as $\mathcal{V}_k - \bigcap_{\delta=1}^{\Delta} F_\delta$. $\mathcal{E}_{k+1}$ is obtained by removing from $\mathcal{E}_k$ edges incident on nodes in $\bigcap_{\delta=1}^{\Delta} F_\delta$, and also excluding edges between all node pairs that have been found in dispute so far.

As noted earlier, the above dispute control phase may be executed in at most $f(f+1)$ instances of NAB.

## B. PROOF OF THEOREM 1

To prove Theorem 1, we first prove that when the coding matrices are generated at random as described, for a particular subgraph $H \in \Omega_k$, with non-zero probability, the coding matrices $\{\mathbf{C_e} | e \in \mathcal{G}_k\}$ define a matrix $\mathbf{C_H}$ (as defined later) such that $\mathbf{D_H C_H} = \mathbf{0}$ if and only if $\mathbf{D_H} = \mathbf{0}$. Then we prove that this is also *simultaneously* true for all subgraphs $H \in \Omega_k$.

### B.1 For a given subgraph $H \in \Omega_k$

Consider any subgraph $H \in \Omega_k$. For each edge $e = (i, j)$ in $H$, we "expand" the corresponding coding matrix $\mathbf{C_e}$ (of size $\rho_k \times z_e$) to a $(n-f-1)\rho_k \times z_e$ matrix $\mathbf{B_e}$ as follows: $\mathbf{B_e}$ consists $n-f-1$ blocks, each block is a $\rho_k \times z_e$ matrix:

- If $i \neq n-f$ and $j \neq n-f$: the $i$-th and $j$-th block equal to $\mathbf{C_e}$ and $-\mathbf{C_e}$, respectively. The other blocks are all set to $\mathbf{0}$: $\mathbf{B_e}^T = \begin{pmatrix} \mathbf{0}\cdots\mathbf{0} & \mathbf{C_e}^T & \mathbf{0}\cdots\mathbf{0} & -\mathbf{C_e}^T & \mathbf{0}\cdots\mathbf{0} \end{pmatrix}$. Here $()^T$ denotes the transpose of a matrix or vector.
- If $i = n-f$: the $j$-th block equals to $-\mathbf{C_e}$, and the other blocks are all set to 0 matrix: $\mathbf{B_e}^T = \begin{pmatrix} \mathbf{0}\cdots\mathbf{0} & -\mathbf{C_e}^T & \mathbf{0}\cdots\mathbf{0} \end{pmatrix}$.
- If $j = n-f$: the $i$-th block equals to $\mathbf{C_e}$, and the other blocks are all set to 0 matrix: $\mathbf{B_e}^T = \begin{pmatrix} \mathbf{0}\cdots\mathbf{0} & \mathbf{C_e}^T & \mathbf{0}\cdots\mathbf{0} \end{pmatrix}$.

Let $D_{i,\beta} = X_i(\beta) - X_{n-f}(\beta)$ for $i < n-f$ as the difference between $\mathbf{X_i}$ and $\mathbf{X_{n-f}}$ in the $\beta$-th element. Recall that $\mathbf{D_i} = \mathbf{X_i} - \mathbf{X_{n-f}} = \begin{pmatrix} D_{i,1} & \cdots & D_{i,\rho_k} \end{pmatrix}$ and $\mathbf{D_H} = \begin{pmatrix} \mathbf{D_1} & \cdots & \mathbf{D_{n-f-1}} \end{pmatrix}$. So $\mathbf{D_H}$ is a row vector of $(n-f-1)\rho_k$ elements from $GF(2^{L/\rho_k})$ that captures the differences between $\mathbf{X_i}$ and $\mathbf{X_{n-f}}$ for all $i < n-f$. It should be easy to see that $(\mathbf{X_i} - \mathbf{X_j})\mathbf{C_e} = \mathbf{0} \Leftrightarrow \mathbf{D_H B_e} = \mathbf{0}$. So for edge $e$, steps 1-2 of Algorithm 1 have the effect of checking whether or not $\mathbf{D_H B_e} = \mathbf{0}$.

If we label the set of edges in $H$ as $e1, e2, \cdots$, and let $m$ be the sum of the capacities of all edges in $H$, then we construct a $(n-f-1)\rho_k \times m$ matrix $\mathbf{C_H}$ by concatenating all expanded coding matrices: $\mathbf{C_H} = \begin{pmatrix} \mathbf{B_{e1}} & \mathbf{B_{e2}} & \cdots \end{pmatrix}$, where each column of $\mathbf{C_H}$ represents one coded symbol sent in $H$ over the corresponding edge. Then steps 1-2 of Algorithm 1 for all edges in $H$ have the same effect as checking whether or not $\mathbf{D_H C_H} = \mathbf{0}$. So to prove Theorem 1, we need to show that there exists $\mathbf{C_H}$ such that $\mathbf{D_H C_H} = \mathbf{0} \Leftrightarrow \mathbf{D_H} = \mathbf{0}$.

It is obvious that if $\mathbf{D_H} = \mathbf{0}$, then $\mathbf{D_H C_H} = \mathbf{0}$ for any $\mathbf{C_H}$. So all left to show is that there exists $\mathbf{C_H}$ such that $\mathbf{D_H C_H} = \mathbf{0} \Rightarrow \mathbf{D_H} = \mathbf{0}$. It is then sufficient to show that $\mathbf{C_H}$ (probably with columns permuted) contains a $(n-f-1)\rho_k \times (n-f-1)\rho_k$ submatrix $\mathbf{M_H}$ that is *invertible*, because when such an invertible submatrix exist, $\mathbf{D_H C_H} = \mathbf{0} \Rightarrow \mathbf{D_H M_H} = \mathbf{0} \Rightarrow \mathbf{D_H} = \mathbf{0}$.

Now we describe how one such submatrix $\mathbf{M_H}$ can be obtained. Notice that each column of $\mathbf{C_H}$ represents one coded symbol sent on the corresponding edge. A $(n-f-1)\rho_k \times (n-f-1)$ submatrix $\mathbf{S}$ of $\mathbf{C_H}$ is said to be a "spanning matrix" of $H$ if the edges corresponding to the columns of $\mathbf{S}$ form a undirected spanning tree of $\overline{H}$, the *undirected* representation of $H$. In Figure 1(d), an undirected spanning tree of the undirected graph in Figure 1(c) is shown in dotted edges. It is worth pointing out that an undirected spanning tree in an undirected graph $\overline{H}$ does not necessarily correspond to a directed spanning tree in the corresponding directed graph $H$. For example, the directed edges in Figure 1(b) corresponding to the dotted undirected edges in Figure 1(d) do not form a spanning tree in the directed graph in Figure 1(b).

It is known that in an undirected graph whose MINCUT equals to $U$, at least $U/2$ undirected unit-capacity spanning trees can be embedded [16]. This implies that $\mathbf{C_H}$ contains a set of $U_k/2$ spanning matrices such that no two spanning matrices in the set covers the same column in $\mathbf{C_H}$. Let $\{\mathbf{S_1}, \cdots, \mathbf{S_{\rho_k}}\}$ be one set of $\rho_k \leq U_k/2$ such spanning matrices of $H$. Then union of these spanning matrices forms an $(n-f-1)\rho_k \times (n-f-1)\rho_k$ matrix $\mathbf{M_H} = \begin{pmatrix} \mathbf{S_1} & \cdots & \mathbf{S_{\rho_k}} \end{pmatrix}$. $\mathbf{M_H}$ is not necessarily a submatrix of $\mathbf{C_H}$, but it is always a submatrix of a column-permuted version $\mathbf{C_H}$.

Next, we will show that when the set of coding matrices are generated as described in Theorem 1, with non-zero probability we obtain an invertible square matrix $\mathbf{M_H}$. When $\mathbf{M_H}$ is invertible, $\mathbf{D_H M_H} = \mathbf{0} \Leftrightarrow \mathbf{D_H} = \mathbf{0} \Leftrightarrow \mathbf{X_1} = \cdots = \mathbf{X_{n-f}}$.

For the following discussion, it is convenient to reorder the elements of $\mathbf{D_H}$ into $\mathbf{\tilde{D}_H} = \begin{pmatrix} D_{1,1} \cdots D_{n-f-1,1} & \cdots D_{1,\rho_k} \cdots D_{n-f-1,\rho_k} \end{pmatrix}$, so that the $(\beta-1)(n-f-1)+1$-th through the $\beta(n-f-1)$ elements of $\mathbf{\tilde{D}_H}$ represent the difference between $\mathbf{X_i}$ ($i = 1, \cdots, n-f-1$) and $\mathbf{X_{n-f}}$ in the $\beta$-th element.

We also reorder the rows of each spanning matrix $\mathbf{S_q}$ ($q = 1, \cdots, \rho_k$) accordingly. It can be showed that after reordering, $\mathbf{S_q}$ becomes $\mathbf{\tilde{S}_q} = \begin{pmatrix} \mathbf{A_q S_{q,1}} \\ \vdots \\ \mathbf{A_q S_{q,\rho_k}} \end{pmatrix}$.

Here $\mathbf{A_q}$ and $\mathbf{S_{q,p}}$ are all $(n-f-1) \times (n-f-1)$ square matrices. $\mathbf{A_q}$ is called the *adjacency* matrix of the spanning tree corresponding to $\mathbf{S_q}$ and is formed as follows. Suppose that the $r$-th column of $\mathbf{S_q}$ corresponds to a coded symbol sent over a directed edge $(i, j)$ in $H$, then

**1.** If $i \neq n-f$ and $j \neq n-f$, then the $r$-th column of $\mathbf{A_q}$ has the $i$-th element as 1 and the $j$-th element as -1, the remaining entries in that column are all 0;
**2.** If $i = n-f$, then the $j$-th element of the $r$-th column of $\mathbf{A_q}$ is set to -1, the remaining elements of that column are all 0;
**3.** If $j = n-f$, then the $i$-th element of the $r$-th column of $\mathbf{A_q}$ is set to 1, the remaining elements of that column are all 0.

For example, suppose $\overline{H}$ is the graph shown in Figure 1(c), and $\mathbf{S_q}$ corresponds to a spanning tree of $H$ consisting of the dotted edges in Figure 1(d). Suppose that we index the corresponding directed edges in the graph shown in Figure 1(b) in the following order: (1,2), (1,3), (4,3). The resulting adjacency matrix $\mathbf{A_q} = \begin{pmatrix} 1 & 1 & 0 \\ -1 & 0 & 0 \\ 0 & -1 & -1 \end{pmatrix}$.

On the other hand, each square matrix $\mathbf{S_{q,p}}$ is a diagonal matrix. The $r$-th diagonal element of $\mathbf{S_{q,p}}$ equals to the $p$-th coefficient used to compute the coded symbol corresponding to the $r$-th column of $\mathbf{S_q}$. In the previous example, suppose the second column of $\mathbf{S_q}$ corresponds to a coded packet $aX_1(1) + bX_1(2)$ being sent on link $(1,3)$. Then the second diagonal elements of $\mathbf{S_{q,1}}$ and $\mathbf{S_{q,2}}$ are $a$ and $b$, respectively.

After the reordering, $\mathbf{M_H}$ can be written as $\mathbf{\tilde{M}_H}$ that has the following structure: $\mathbf{\tilde{M}_H} = \begin{pmatrix} \mathbf{A_1 S_{1,1}} & \mathbf{A_2 S_{2,1}} & \cdots & \mathbf{A_{\rho_k} C_{\rho_k,\rho_k}} \\ \vdots & & \ddots & \vdots \\ \mathbf{A_1 S_{1,\rho_k}} & \mathbf{A_2 S_{2,\rho_k}} & \cdots & \mathbf{A_{\rho_k} S_{\rho_k,\rho_k}} \end{pmatrix}$.

Notice that $\mathbf{\tilde{M}_H}$ is obtained by permuting the rows of $\mathbf{M_H}$. So showing $\mathbf{M_H}$ is invertible is equivalent to showing $\mathbf{\tilde{M}_H}$ is invertible.

Define $\mathbf{M_q} = \begin{pmatrix} \mathbf{A_1 S_{1,1}} & \cdots & \mathbf{A_q S_{q,1}} \\ \vdots & \ddots & \vdots \\ \mathbf{A_1 S_{1,q}} & \cdots & \mathbf{A_q S_{q,q}} \end{pmatrix}$ for $1 \leq q \leq \rho_k$. Note that $\mathbf{M_{q1}}$ is a sub-matrix of $\mathbf{M_{q2}}$ when $q1 < q2$, and $\mathbf{M_{\rho_k}} = \mathbf{\tilde{M}_H}$. We prove the following lemma:

Lemma 1. *For any $\rho_k \leq U_k/2$, with probability at least $\left(1 - \frac{n-f-1}{2^{L/\rho_k}}\right)^{\rho_k}$, matrices $\mathbf{\tilde{M}_H}$ and $\mathbf{M_H}$ are invertible.*

Proof. We now show that each $\mathbf{M_q}$ is invertible with probability at least $\left(1 - \frac{n-f-1}{2^{L/\rho_k}}\right)^{q}$ for all $q \leq \rho_k$. The proof is by induction, with $q = 1$ being the base case.

### *Base Case:* $q = 1$.

For $q = 1$, $\mathbf{M_1} = \mathbf{A_1 S_{1,1}}$. As showed in Appendix C.3 of [15], $\mathbf{A_q}$ is always invertible and $\det(\mathbf{A_q}) = \mathbf{\pm 1}$. Since $\mathbf{S_{1,1}}$ is a $(n-f-1)$-by-$(n-f-1)$ diagonal matrix, it is invertible provided that all its $(n-f-1)$ diagonal elements are non-zero. Remember that the diagonal elements of $\mathbf{S_{1,1}}$ are chosen uniformly and independently from $GF(2^{L/\rho_k})$. The probability that they are all non-zero is $\left(1 - \frac{1}{2^{L/\rho_k}}\right)^{n-f-1} \geq 1 - \frac{n-f-1}{2^{L/\rho_k}}$.

### *Induction Step:* $q < \rho_k$ *to* $q + 1 \leq \rho_k$.

For this part, we rewrite $\mathbf{M_{q+1}} = \begin{pmatrix} \mathbf{M_q} & \mathbf{P_q} \\ \mathbf{F_q} & \mathbf{A_{q+1} S_{q+1,q+1}} \end{pmatrix}$, where $\mathbf{P_q} = \begin{pmatrix} \mathbf{A_{q+1} S_{q+1,1}} \\ \vdots \\ \mathbf{A_{q+1} S_{q+1,q}} \end{pmatrix}$ is an $(n-f-1)q$-by-$(n-f-1)$ matrix, and $\mathbf{F_q} = \begin{pmatrix} \mathbf{A_1 S_{1,q+1}} & \cdots & \mathbf{A_q S_{q,q+1}} \end{pmatrix}$ is an $(n-f-1)$-by-$(n-f-1)q$ matrix. Assuming that $\mathbf{M_q}$ is invertible, we transform $\mathbf{M_{q+1}}$ into $\mathbf{M'_{q+1}}$ as follows:

$$\mathbf{M'_{q+1}} = \begin{pmatrix} \mathbf{I_{(n-f-1)q}} & \mathbf{0} \\ \mathbf{0} & \mathbf{A_{q+1}^{-1}} \end{pmatrix} \begin{pmatrix} \mathbf{M_q} & \mathbf{P_q} \\ \mathbf{F_q} & \mathbf{A_{q+1} S_{q+1,q+1}} \end{pmatrix} \begin{pmatrix} \mathbf{I_{(n-f-1)q}} & -\mathbf{M_q^{-1} P_q} \\ \mathbf{0} & \mathbf{I_{(n-f-1)}} \end{pmatrix}$$

$$= \begin{pmatrix} \mathbf{M_q} & \mathbf{0} \\ \mathbf{A_{q+1}^{-1} F_q} & \mathbf{S_{q+1,q+1}} - \mathbf{A_{q+1}^{-1} F_q M_q^{-1} P_q} \end{pmatrix}.$$

Here $\mathbf{I_{(n-f-1)q}}$ and $\mathbf{I_{(n-f-1)}}$ each denote a $(n-f-1)q \times (n-f-1)q$ and a $(n-f-1) \times (n-f-1)$ identity matrices. Note that $|\det(\mathbf{M'_{q+1}})| = |\det(\mathbf{M_{q+1}})|$, since in the first equation above, 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 $(n-f-1) \times (n-f-1)$ diagonal matrix $\mathbf{S_{q+1,q+1}}$ are chosen independently from $\mathbf{M_q}$ and $\mathbf{A_{q+1}^{-1} F_q M_q^{-1} P_q}$. It can be proved that $\mathbf{S_{q+1,q+1}} - \mathbf{A_{q+1}^{-1} F_q M_q^{-1} P_q}$ is invertible with probability at least $1 - \frac{n-f-1}{2^{L/\rho_k}}$ (see Appendix C.4 of [15]). According to the induction assumption, $\mathbf{M_q}$ is invertible with probability at least $\left(1 - \frac{n-f-1}{2^{L/\rho_k}}\right)^{q}$. So we have $\Pr\{\mathbf{M_{q+1}} \text{ is invertible}\} \geq \left(1 - \frac{n-f-1}{2^{L/\rho_k}}\right)^{q+1}$. This completes the induction. Now we can see that $\mathbf{M_{\rho_k}} = \mathbf{\tilde{M}_H}$ is invertible with probability $\geq \left(1 - \frac{n-f-1}{2^{L/\rho_k}}\right)^{\rho_k} \geq 1 - \frac{(n-f-1)\rho_k}{2^{L/\rho_k}} \to 1$, as $L \to \infty$. □

Now we have proved that there exists a set of coding matrices $\{\mathbf{C_e} | e \in \mathcal{E}_k\}$ such that the resulting $\mathbf{C_H}$ satisfies the condition that $\mathbf{D_H}\mathbf{C_H} = \mathbf{0}$ if and only if $\mathbf{D_H} = \mathbf{0}$.

## B.2 For all subgraphs in $\Omega_k$

In this section, we are going to show that, for $\mathcal{G}_k$, if the coding matrices $\{\mathbf{C_e} | e \in \mathcal{E}_k\}$ are generated as described in Theorem 1, then with non-zero probability the set of square matrices $\{\mathbf{M_H} | H \in \Omega_k\}$ are all invertible *simultaneously*. When this is true, there exists a set of coding matrices that is correct. Note that the random coefficients are first chosen for all edges in $\mathcal{G}_k$, and then coefficients in graph $H$ come from the corresponding edges in $\mathcal{G}_k$. This implies that the coefficients in the polynomials for different $\mathbf{M_H}$ for different $H$ are overlapping sets.

According to Lemma 1, each $\mathbf{M_H}$ ($H \in \Omega_k$) is *not* invertible with probability at most $\frac{(n-f-1)\rho_k}{2^{L/\rho_k}}$. According to the union bound, it follows that the probability that all matrices $\{\mathbf{M_H} | H \in \Omega_k\}$ are simultaneously invertible with probability at least $1 - |\Omega_k|\frac{(n-f-1)\rho_k}{2^{L/\rho_k}}$. According to the way $\mathcal{G}_k$ is constructed and the definition of $\Omega_k$, it should not be hard to see that $\mathcal{G}_k$ is a subgraph of $\mathcal{G}_1 = \mathcal{G}$, and $\Omega_k \subseteq \Omega_1$. Notice that $|\Omega_1| = \binom{n}{n-f}$. So $|\Omega_k| \leq \binom{n}{n-f}$ and all $\mathbf{M_H}$ ($H \in \Omega_k$) are simultaneously invertible with probability at least $1 - \binom{n}{n-f}\frac{(n-f-1)\rho_k}{2^{L/\rho_k}}$.

This result shown here implies that for sufficiently large $L$, there exist a set of correct coding matrices $\{\mathbf{C_e} | e \in \mathcal{E}_k\}$. By considering *all* subgraphs in $\Omega_k$, we essentially ensure that equality check is performed between all pairs of fault-free nodes in $\mathcal{G}_k$: for any pair of fault-free nodes $(i, j)$, there exists an $H \in \Omega_k$ consisting of only fault-free nodes that includes $i$ and $j$ both; hence $x_i$ and $x_j$ will be checked for equality within this $H$. Then Theorem 1 follows.

# C. THROUGHPUT OF NAB

First consider the time cost of each operation in instance $k$ of NAB :

**Phase 1**: It takes $L/\gamma_k \leq L/\gamma^*$ time units, since unreliable broadcast from the source node 1 at rate $\gamma_k$ is achievable and $\gamma_k \geq \gamma^*$, as discussed in Section 2.

**Phase 2 – Equality check**: As discussed previously, it takes $L/\rho_k \leq L/\rho^*$ time units.

**Phase 2 – Broadcasting outcomes of equality check**: To reliably broadcast the 1-bit flags from the equality check algorithm, a previously proposed Byzantine broadcast algorithm, such as [6], is used. The algorithm from [6], denoted as `Broadcast_Default` hereafter, reliably broadcasts 1 bit by communicating no more than $P(n)$ bits in a *complete* graph, where $P(n)$ is a polynomial of $n$. In our setting, $\mathcal{G}$ might not be complete. However, from each node $i$ to each node $j$, $2f+1$ node-disjoint paths exists. In this case, since there are at most $f$ faulty nodes, reliable *end-to-end* communication from node $i$ to node $j$ can be achieved by sending the same copy of data along a set of $2f + 1$ node-disjoint paths and taking the majority at node $j$. By doing this, we can emulate a complete graph in an incomplete graph $\mathcal{G}$. Then it can be showed that, by running `Broadcast_Default` on top of the emulated complete graph, reliably broadcasting the 1-bit flags can be completed in $O(n^\alpha)$ time units, for some constant $\alpha > 0$.

**Phase 3**: If Phase 3 is performed in instance $k$, every node $i$ in $\mathcal{V}_k$ uses `Broadcast_Default` to reliably broadcast all the messages that it claims to have received from other nodes, and sent to the other nodes, during Phase 1 and 2 of the $k$-th instance. Similar to the discussion above about broadcasting the outcomes of equality check, it can be showed that the time it takes to complete Phase 3 is $O(Ln^\beta)$ for some constant $\beta > 0$.

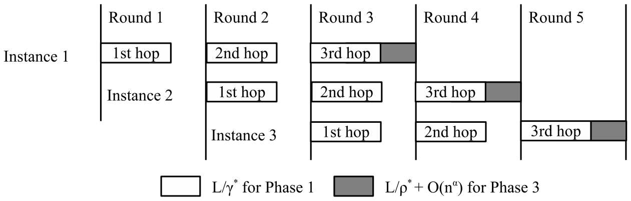


**Figure 2: Example of pipelining**

Now consider a sequence of $Q > 0$ instances of NAB. As discussed previously, Phase 3 will be performed at most $f(f+1)$ times throughout the execution of the algorithm. So we have the following upper bound of the execution time of $Q$ instances of NAB: $t(\mathcal{G}, L, Q, NAB) \leq Q\left(\frac{L}{\gamma^*} + \frac{L}{\rho^*} + O(n^\alpha)\right) + f(f+1)O(Ln^\beta)$. Given that $f < n$, throughput of NAB can be lower bounded by

$$\begin{aligned} T(\mathcal{G}, L, NAB) &= \lim_{Q\to\infty} \frac{LQ}{t(\mathcal{G}, L, Q, NAB)} \\ &\geq \lim_{Q\to\infty} \frac{LQ}{Q\left(\frac{L}{\gamma^*} + \frac{L}{\rho^*} + O(n^\alpha)\right) + f(f+1)O(Ln^\beta)} \\ &\geq \lim_{Q\to\infty} \left(\frac{\gamma^* + \rho^*}{\gamma^*\rho^*} + \frac{O(n^\alpha)}{L} + \frac{O(n^{\beta+2})}{Q}\right)^{-1}. \end{aligned}$$

Note that for a given graph $\mathcal{G}, n, \gamma^*, \rho^*, \alpha, \beta$ are all constants independent of $L$ and $Q$. So for sufficiently large values of $L$ and $Q$, the last two terms in the last inequality become negligible compared to the first term, and the throughput of NAB approaches to a value that is at least as large as $\frac{\gamma^*\rho^*}{\gamma^*+\rho^*}$.

In the above discussion, we implicitly assumed that transmissions during the unreliable broadcast in Phase 1 are performed all at the same time, by assuming no propagation delay. However, when propagation delay is considered, a node cannot forward a message/symbol until it finishes receiving it. So for the $k$-th instance of NAB, the information broadcast by the source propagates only one hop every $L/\gamma_k$ time units. So for a large network, the "time span" of Phase 1 can be much larger than $L/\gamma_k$. This problem can be solved by pipelining: We divide the time horizon into rounds of $\left(\frac{L}{\gamma^*} + \frac{L}{\rho^*} + O(n^\alpha)\right)$ time units. For each instance of NAB, the $L$-bit input from the source node 1 propagates one hop per round, using the first $L/\gamma^*$ time units, until Phase 1 completes. Then the remaining $\left(\frac{L}{\rho^*} + O(n^\alpha)\right)$ time units of the last round is used to perform Phase 2, using all the links. An example in which the broadcast in Phase 1 takes 3 hops is shown in Figure 2. By pipelining, we achieve the lower bound from Eq.1.

# D. CONSTRUCTION OF $\Gamma$

A subgraph of $\mathcal{G}$ belonging to $\Gamma$ is obtained as follows: We will say that edges in $W \subset \mathcal{E}$ are "explainable" if there exists a set $F \subset \mathcal{V}$ such that (i) $F$ contains at most $f$ nodes, and (ii) each edge in $W$ is incident on at least one node in $F$. Set $F$ is then said to "explain set $W$".

Consider each *explainable* set of edges $W \subset \mathcal{E}$. Suppose that

$F_1, \cdots, F_\Delta$ are all the subsets of $\mathcal{V}$ that *explain* edge set $W$. A subgraph $\Psi_W$ of $\mathcal{G}$ is obtained by removing edges in $W$ from $\mathcal{E}$, and nodes in $\bigcap_{\delta=1}^{\Delta} F_\delta$ from $\mathcal{V}$.[2] In general, $\Psi_W$ above may or may not contain the source node 1. Only those $\Psi_W$'s that do contain node 1 belongs to $\Gamma$.

# E. PROOF OF THEOREM 2

In arbitrary point-to-point network $\mathcal{G}(\mathcal{V}, \mathcal{E})$, 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 $C_{BB}(\mathcal{G}) \leq \gamma^*$

Proof. Consider any $\Psi_W \in \Gamma$ and let $W$ is the set of edges in $\mathcal{G}$ but not in $\Psi_W$. By the construction of $\Gamma$, there must be at least one set $F \subset \mathcal{V}$ that explains $W$ and does not contain the source node 1. We are going to show that $C_{BB}(\mathcal{G}) \leq MINCUT(\Psi_W, 1, i)$ for every node $i \neq 1$ that is in $\Psi_W$.

Notice that there must exist a set of nodes that explains $W$ and does not contain node 1; otherwise node 1 is not in $\Psi_W$. Without loss of generality, assume that $F_1$ is one such set nodes.

First consider any node $i \neq 1$ in $\Psi_W$ such that $i \notin F_1$. For $f > 0$, such a node $i$ must exist since $|F_1| \leq f$ and $\Psi_W$ contains $n - f$ nodes where $n > 3f$. Let all the nodes in $F_1$ be faulty such that they refuse to communicate over edges in $W$, but otherwise behave correctly. In this case, since the source is fault-free, node $i$ must be able to receive the $L$-bit input that node 1 is trying to broadcast. So $C_{BB}(\mathcal{G}) \leq MINCUT(\Psi_W, 1, i)$.

Next we consider a node $i \neq 1$ in $\Psi_W$ and $i \in F_1$. Since $F_1$ is non-empty, such a node $i$ exists. Notice that node $i$ cannot be contained in all sets of nodes that explain $W$, otherwise node $i$ cannot be in $\Psi_W$. Then there are only two possibilities:
**Case 1:** There exist a set $F$ explaining $W$ that contains neither node 1 nor node $i$. In this case, $C_{BB}(\mathcal{G}) \leq MINCUT(\Psi_W, 1, i)$ according to the above argument by replacing $F_1$ with $F$.
**Case 2:** any set $F$ that explains $W$ and does not contain node $i$ contains node 1. Let $F_2$ be one such set containing node 1 but not node $i$.

Define $V^- = \mathcal{V} - F_1 - F_2$. $V^-$ is not empty since $F_1$ and $F_2$ both contain at most $f$ nodes and there are $n \geq 3f + 1$ nodes in $\mathcal{V}$. Consider two scenarios with the same input value $x$: (1) Nodes in $F_1$ (which does not contain node 1) are faulty and they behave as if links in $W$ are broken, but otherwise behave correctly; and (2) Nodes in $F_2$ (contains node 1) are faulty and they behave as if links in $W$ are broken, but otherwise behave correctly. In both cases, nodes in $V^-$ are fault-free.

Observe that among edges between nodes in $V^-$ and $F_1 \cup F_2$, only edges between $V^-$ and $F_1 \cap F_2$ could have been removed, because otherwise $W$ cannot be explained by both $F_1$ and $F_2$. So nodes in $V^-$ cannot distinguish between the two scenarios above. 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 to 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 have to learn $x$. So $C_{BB}(\mathcal{G}) \leq MINCUT(\Psi_W, 1, i)$. Recall that $\gamma^*$ equals to the minimum of $MINCUT(\Psi_W, 1, i)$ over all $\Psi_W \in \Gamma$ and $i$. This completes the proof. □

## E.2 $C_{BB}(\mathcal{G}) \leq 2\rho^*$

Proof. For a subgraph $H \in \Omega_1$ and the corresponding $\overline{H} \in \overline{\Omega}_1$, denote $U_H = \min_{nodes\ i,j\ in\ \overline{H}} MINCUT(\overline{H}, i, j)$. We will prove the upper bound by showing that $C_{BB}(G) \leq U_H$ for every $H \in \Omega_1$.

Suppose on the contrary that Byzantine broadcast can be done at a rate $R > U_H + \epsilon$ for some constant $\epsilon > 0$. So there exists a BB algorithm, named $\mathcal{A}$, that can broadcast $t(U_H + \epsilon)$ bits in using $t$ time units, for some $t > 0$.

Let $E$ be a set of edges in $H$ that corresponds to one of the minimum-cuts in $\overline{H}$. In other words, $\sum_{e \in E} z_e = U_H$, and the nodes in $H$ can be partitioned into two non-empty sets $\mathcal{L}$ and $\mathcal{R}$ such that $\mathcal{L}$ and $\mathcal{R}$ are disconnected from each other if edges in $E$ are removed. Also denote $F$ as the set of nodes that are in $\mathcal{G}$ but not in $H$. Notice that since $H$ contains $(n - f)$ nodes, $F$ contains $f$ nodes.

Notice that in $t$ time units, at most $tU_H < t(U_H + \epsilon)$ bits of information can be sent over edges in $E$. According to the pigeonhole principle, there must exist two different input values of $t(U_H + \epsilon)$ bits, denoted as $u$ and $v$, such that in the absence of misbehavior, broadcasting $u$ and $v$ with algorithm $\mathcal{A}$ results in the same communication pattern over edges in $E$.

First consider the case when $F$ contains the source node 1. Consider the three scenarios using algorithm $\mathcal{A}$:

1. Node 1 broadcasts $u$, and none of the nodes misbehave. So all nodes should set their outputs to $u$.
2. Node 1 broadcasts $v$, and none of the nodes misbehave. So all nodes should set their outputs to $v$.
3. Nodes in $F$ are faulty (includes the source node 1). The faulty nodes in $F$ behave to nodes in $\mathcal{L}$ as in scenario 1, and behave to nodes in $\mathcal{R}$ as in scenario 2.

We show in [15] that nodes in $\mathcal{L}$ cannot distinguish scenario 1 from scenario 3, and nodes in $\mathcal{R}$ cannot distinguish scenario 2 from scenario 3. So in scenario 3, nodes in $\mathcal{L}$ set their outputs to $u$ and nodes in $\mathcal{R}$ set their outputs to $v$. This violates the agreement condition and contradicts with the assumption that $\mathcal{A}$ solves BB at rate $U_H + \epsilon$. Hence $C_{BB}(\mathcal{G}) \leq U_H$.

Next consider the case when $F$ does not contain the source node 1. Without loss of generality, suppose that node 1 is in $\mathcal{L}$. Consider the following three scenarios:

1. Node 1 broadcasts $u$, and none of the nodes misbehave. So all nodes should set their outputs to $u$.
2. Node 1 broadcasts $v$, and none of the nodes misbehave. So all nodes should set their outputs to $v$.
3. Node 1 broadcasts $u$, and nodes in $F$ are faulty. The faulty nodes in $F$ behave to nodes in $\mathcal{L}$ as in scenario 1, and behave to nodes in $\mathcal{R}$ as in scenario 2.

In this case, we show in [15] that nodes in $\mathcal{L}$ cannot distinguish scenario 1 from scenario 3, and nodes in $\mathcal{R}$ cannot distinguish scenario 2 from scenario 3. So in scenario 3, nodes in $\mathcal{L}$ set their outputs to $u$ and nodes in $\mathcal{R}$ set their outputs to $v$. This violates the agreement condition and contradicts with the assumption that $\mathcal{A}$ solves BB at rate $U_H + \epsilon$. Hence $C_{BB}(\mathcal{G}) \leq U_H$. Recall that $\rho^*$ equals to the minimum of $MINCUT(\overline{H}, i, j)$ over all $\overline{H} \in \Omega_1$ and $(i, j)$. This completes the proof. □

[2] It is possible that $\Psi_W$ for different $W$ may be identical. This does not affect the correctness of our algorithm.