Relay Protocol for Approximate Byzantine Consensus

Matthew Ding
matthew.c.ding2012@gmail.com

Westford Academy

October 10, 2020

Abstract

This paper presents a novel algorithm for Approximate Byzantine Consensus (ABC), called Relay-ABC. The algorithm allows machines to achieve approximate consensus to arbitrary exactness in the presence of byzantine failures. The algorithm relies on the novel usage of a relayed messaging system and signed messages with unforgeable signatures that are unique to each node. The use of signatures and relays allows the strict necessary network conditions of traditional approximate byzantine consensus algorithms to be circumvented.

We also provide theoretical guarantees of validity and convergence for Relay-ABC. To do this, we utilize the idea that the iteration of states in the network can be modelled by a sequence of transition matrices. We extend previous methods, which use transition matrices to prove ABC convergence, by having each state vector model not just one iteration, but a set of D iterations, where D is a diameter property of the graph. This allows us to accurately model the delays of messages inherent within the relay system.

1 Introduction

The idea of byzantine fault-tolerance was first introduced by Lamport et al. [7]. Byzantine consensus has since become a large research topic, with applications such as blockchain technology [3] and machine learning [1].

Dolev et al. [5] modified and extended the problem of byzantine agreement by introducing *approximate Byzantine agreement*, allowing machines to reach approximate consensus rather than exact consensus. This was motivated by the fact that exact consensus in asynchronous systems was proven to be impossible [6]. Additionally, in synchronous systems, approximate byzantine consensus can be used to create algorithms that do not require complete knowledge of the network topology [8].

This approximate byzantine consensus problem aims to have all honest machines converge to a single state within the convex hull of initial states as the number of iterations approaches infinity [5]. Vaidya [9] utilizes a method describing the progression of states in the network using transition matrices to prove consensus.

The main contribution of this paper is to utilize two key tools that have seen a lot of use and success in traditional byzantine consensus problems: messages with unforgeable signatures [4, 7, 11] and relaying messages [7, 12]; our work is a generalization of [9], with signatures and relays used to circumvent certain network assumptions that would otherwise be necessary. To the best of my knowledge, this is the first time either of these two tools have been used in approximate byzantine consensus algorithms.

Byzantine consensus has had applications to machine learning by having machines perform gradient descent steps to minimize a loss function on top of consensus techniques. The combination of delays and signatures may have additional applications in byzantine gradient descent. [14] extends approximate byzantine consensus algorithms for use in byzantine gradient descent methods. We leave the question as to whether Relay-ABC may have a similar extension to our future work.

2 Problem Formulation

2.1 Definitions

2.1.1 Byzantine Agreement Definitions

- Let m be the total number of machines (or "nodes"), h the number of honest machines, and b be the number of byzantine machines ($h+b=m$).
- Let B denote the set of byzantine nodes and H denote the set of honest nodes. The set of all nodes V is equal to $B \cup H$.
- Denote N_i^I to be the set of all machines that have incoming edges from machine i. Denote N_i^O to be the set of all machines that have outgoing edges to machine i
- Define $dist(i,j)$, for some $i,j \in V$ as the length of the shortest path from node i to j
- In this paper, a "non-zero value" refers to a value that can be lower bounded by some positive constant. In the context of transition matrices, this value doesn't approach 0 as the number of iterations approaches infinity.

2.1.2 Matrix Definitions

Let M denote some arbitrary matrix, and $M[t]$ denote some arbitrary matrix with respect to iteration t.

- $M_i[t]$ denotes the ith row of matrix $M[t]$
- $M_{ij}[t]$ denotes the element at row i and column j of matrix $M[t]$
- We define the first row and column in every matrix as row/column 0
- Matrix Splicing: $M[a, b : c, d]$ denotes the submatrix spliced by top row a, bottom row b, left column c and right column d (all inclusive) of matrix M
- A non-zero column is a matrix column filled entirely with non-zero elements

2.2 Decentralized Learning Model

We consider a static, directed network $G(V, E)$, where $V = \{0, 1, 2, ...m - 1\}$ and E representing communication links between neighboring nodes. If $(i, j) \in E$, then node i may send messages to node j. Our protocol is analyzed in the synchronous communication setting, where communication occurs over a sequence of iterations. Messages sent during an iteration are guaranteed to be received by the intended recipient.

Each node $i \in V$ starts with an initial real-valued input. The goal of the protocol is to approach a state that satisfies the following two conditions:

- Validity condition: After each iteration of the protocol, the state of each honest node remains within the convex hull of the initial inputs of all honest nodes.
- Convergence condition: The difference between the states of any two honest nodes approaches zero as the number of iterations approaches infinity

2.3 Byzantine Failure Model

Among the m machines in the decentralized network, b of them are byzantine machines. Byzantine machines may deviate arbitrarily from the protocol. For example, a byzantine machine may output any arbitrary real value, and send mismatching messages to each of its neighbors. However, a key restriction of byzantine nodes is that they cannot forge signatures of honest users.

3 Relay-ABC Algorithm

3.1 Assumptions

Definition 3.1. Honest Subgraph: Define the honest subgraph as the graph that is formed by removing all byzantine nodes and all edges connected to byzantine nodes in the original graph.

- The number of byzantine machines is strictly less than one-third the total number of machines ($b < \frac{1}{3}m$).

- We assume the honest subgraph is bidirectionally connected (there exists directed paths from every honest node to every other honest node).
- We assume the diameter of the honest subgraph is upper-bounded by D.

3.2 Our Contributions

Our work is an extension of the work done in [9]. Our network assumptions are much less restrictive. In particular, we assume no network connectivity assumptions besides the honest subgraph being bidirectionally connected, which is necessary for any algorithm to achieve consensus over all honest nodes.

The goal of this protocol is to use a relay system to bypass traditional network connectivity assumptions outlined in [10], which has a necessary but insufficient condition that each node has an indegree of at least $2b+1$. This requires that each honest node have at least $b+1$ incoming edges from honest neighbors. On the other hand, bidirectional connectivity of the honest subgraph may possibly be achieved when each honest node has a maximum indegree of as low as one honest neighbor.

This comes at the tradeoff of higher communication costs, as now machines send to each other at most m sets of parameters in each message, as opposed to one parameter in most other algorithms in literature.

Our paper provides theoretical guarantees of validity and convergence to an approximate byzantine consensus algorithm with message delays, which to our knowledge has never been done before. Our work also introduces the use of unforgeable signatures. Signatures have seen lots of successful usage in standard byzantine consensus, but until now have not been used in approximate byzantine consensus methods.

3.3 High-Level Idea

Since the honest subgraph is bidirectionally connected, this allows all honest machines to receive signed messages from every other honest machine through a broadcast and relay system. Thus we create a pseudo-complete communication graph over the course of an entire phase of D rounds. Since a complete graph does indeed satisfy the necessary conditions of [10], our relay protocol achieves convergence as well.

We use a trimmed-mean aggregation step [14, 9] to ensure byzantine robustness. The trimmed-mean step works by eliminating the greatest b values and the smallest b values, and then taking the arithmetic mean of the remaining values. By removing the greatest and least b values, we ensure that the maximum and minimum values of the set of remaining values are both values of honest machines. This prevents byzantine machines from making the states of honest machines deviate arbitrarily.

Each machine i keeps track of their own vector v_i. This vector consists of $(v_i(0), v_i(1)...v_i(m-1))$, where $v_i(j)$ **represents machine i's most recently updated record of machine j's state.** v_i may not always contain a state

that was received from an actual message for each machine in the network, as machine i may not have received a message from all machines.

Machine i only starts performing trimmed-mean steps after D iterations, as this ensures that the first broadcast of all honest machines has had sufficient time to relay across the entire graph and reach every other honest node. This means that each honest machine is outputting an identical message, their initial input value, for the first D iterations.

We choose the specific value D, the upper-bound of the diameter of the honest subgraph, so after D iterations, all honest vectors will always contain more honest parameters than byzantine parameters. D is in the worst-case $O(h)$, but with high probability is $O(\log h)$ [2].

For all honest machines i and j, any state $v_i(j)$ in the vector will contain a valid signature from machine j, as well as a iteration marker, which shows which iteration the parameter was calculated on. Denote $T(v_i(j))$ to be the iteration that parameter $v_i(j)$ was calculated on.

We now introduce the Relay-ABC Algorithm in the following subsection.

3.4 Relay-ABC Algorithm

Algorithm 1: Relay-ABC

Remark. *This algorithm is implemented by a specific machine i. Each machine $i \in H$ will implement this algorithm concurrently.*

Result: Each state $v_i(i)$ converges to the same value within the convex hull of the initial states as Iteration $t \to \infty$

Initialization:

$v_i(i) \leftarrow$ Intial State of node i (with signature i and iteration marker -1).

for *Iteration* $t \leftarrow 0$ **to** T **do**

Broadcast v_i to all machines $j \in N_i^O$

Receive v_j from all machines $j \in N_i^I$

Remark. *When receiving v_j, ignore all parameters received that are not properly signed or without a proper iteration marker. If no proper message is from a node, set their incoming value to be an arbitrary predefined real value (e.g. 0).*

$G_i \leftarrow N_i^O \cup \{i\}$

for $j \leftarrow 0$ **to** $m-1$ **do**

Remark. *In the next two lines, we do the following: Out of all parameters $v(j)$ received from the broadcast step, set $v_i(j)$ to the value with the highest iteration marker.*

if $j \neq i$ **then**

$g' \leftarrow \arg\max_{g:\{g \in G_i\}} T(v_g(j))$

$v_i(j) \leftarrow v_{g'}(j)$

end

end

if $t \geq D$ **then**

Trimmed-mean update step:

In a new vector, sort the values of v_i in increasing order:

$$v_i^* \leftarrow sort(v_i) \tag{1}$$

Ignore the least and greatest b values, and set the value of $v_i(i)$ to be the average of all remaining values in v_i^*, as defined below:

$$v_i(i) \leftarrow \frac{1}{m-2b} \sum_{k=b}^{m-b-1} v_i^*(k) \tag{2}$$

Add signature i and iteration marker t to $v_i(i)$

end

end

4 Theoretical Guarantees

4.1 Overview

In the following section, we prove that the Relay-ABC algorithm satisfies the validity and convergence conditions.

We define transition matrix $M[t]$ and construct it so that it models the state update of all honest nodes as defined in Algorithm 1. The transition matrices are then used to prove that Algorithm 1 guarantees convergence over all honest nodes.

4.2 Matrix Definitions

We introduce several definitions for matrices, most of which are adapted from [9].

To make analysis easier, we will differentiate between *iterations* (r) and *phases* (t). Define an *iteration* as a single instance in time of communications. During each iteration, every node sends and receives messages from its neighbors. Define a *phase* as a set of D iterations. Therefore the ith phase contains iterations $(i-1)D$ to iD. Note that the first iteration of the procol is iteration 0, while the first phase is phase 1. Since the distance between any two honest nodes is at most D, any message from an honest node is guaranteed to reach all other honest nodes within D iterations. We consider phases of D iterations for theoretical analysis, not for any actual implementation in the protocol.

Denote $v[t]$ as the column vector consisting of the states of all honest nodes in phase t (over all D iterations). $\|v[t]\| = hD$, and $v_{rh+i}[t]$ represents the state of node i at iteration r of phase t.

Denote $v[0]$ to be the column vector consisting of the initial states of all honest nodes during the first D iterations. Since all machines output the same identical message for the first D iterations, $v[0]$ consists of D identical $h \times 1$ column vectors stacked on top of each other.

We express the iterative update of the state of a fault-free node $i \in H$ in any single phase using the matrix form below:

$$v_i[t] = M_i[t-1] * v[t-1] \tag{3}$$

The row vector $M_i[t]$ satisfies the following conditions:

$M_i[t]$ is a stochastic row vector of size hD (proven in Appendix C). Thus, $M_{ij}[t] \geq 0$ for $0 \leq j \leq hD-1$, and

$$\sum_{j=1}^{hD} M_{ij}[t] = 1$$

By stacking hD stochastic row matrices $M_i[t]$ on top of one another, where M[t] is an $hD \times hD$ matrix, we can represent to state update of all honest nodes in a single matrix multiplication:

$$v[t] = M[t-1] * v[t-1] \tag{4}$$

The matrix M[t] models the state update detailed in Algorithm 1. We detail the specifics of the construction in Section 4.3.

Each element in a row of the transition matrix represents some weight of the state column of the previous D iterations. We describe every update of (2) of a single node during a single phase as a matrix row.

By repeating the transition matrix update of (4) $T + 1$ times, we get:

$$v[T] = \prod_{t=0}^{T} M[t] * v[0] \tag{5}$$

Note that this is an extension of the transition matrix idea of [9]. Instead of a $h \times h$ transition matrix and a $h \times 1$ state vector representing the update of a single iteration, we use an expanded transition matrix to describe the update for the set of the next phase (D iterations) using the set of states from the previous phase.

4.3 Transition Matrix Construction

In this section, we introduce how to construct transition matrices to exactly model the transition of states as dictated by the Relay-ABC algorithm (Algorithm 1).

We define how to construct a given row of the transition matrix. We introduce some new definitions and notation, most of which is adapted from [9]:

Let us consider an arbitrary honest node i performing the update step (2) at some iteration. Vector v_i is the set of all most updated states from each machine known to machine i, and is the set of all values being considered in the trimmed-mean step. It is known that $\|v_i\| = m$. Define set L and S to be the largest and smallest b values, respectively, in vector v_i. Let N_i^* denote the set all all states that were not removed in the trimmed-mean update step. It is known that L and S are disjoint sets, $|L| = |S| = b$, $N_i^* = v_i - (L \cup S)$, and $\|N_i^*\| = m - 2b$.

Denote f to be the number of faulty states within N_i^* (faulty states that were not trimmed away.) Define subsets L^* and S^* such that $L^* \subseteq L$, $S^* \subseteq S$, $|L^*| = |S^*| = X$, and that L^* and S^* consist of only honest states (which may be arbitrarily chosen.) It is shown in [9] that such subsets always exist.

Algorithm 2: Row Construction Algorithm

Remark. *This algorithm describes how to construct row t of M, the $hD \times hD$ transition matrix.*

Initialization: For all integers k such that $0 \leq k < hD : M_{tk} \leftarrow 0$

if $f = 0$ **then**
  $G \leftarrow$ Case 1 Construction
else
  $G \leftarrow$ Case 2 Construction
for *Node* $k \in V$ **do**
  if $k \neq i$ **then**
    $iter = t \bmod h - dist(i, k)$
  else
    $iter = t \bmod h - 1$
  $j = iter \bmod D$
  $j = j * h + k$
  if $iter < 0$ **then**

$$M_{tj} = M_{tj} + G_{ik} \tag{6}$$

  else
    for $v \leftarrow 0$ **to** $hD - 1$ **do**

$$M_{tv} = M_{tv} + G_{ik} * M_{jv} \tag{7}$$

    end
end

4.3.1 Matrix Construction Algorithm Overview

To construct our matrix, we consider two cases: Case 1 where N_i^* contains no states from faulty nodes ($f = 0$), and Case 2 where N_i^* contains states from at least one faulty node ($f > 0$) [9]. We describe our transition matrix by using elements of the transition matrices in [9].

Without loss of generality, we denote the first iteration of the given phase corresponding to the transition matrix as iteration 0. This is for simplicity, and in actuality all iterations will be shifted upwards by some positive multiple of D. Here we will describe how to construct row M_t, which represents iteration $\lfloor \frac{t}{h} \rfloor$, where $t < hD$. Assume without loss of generality that row t is a Node i row.

In the first iteration of updates of a phase (represented by a single transition matrix), every value $v_i(j)$ in (2) can be represented exactly by a value in the state vector (a state of the previous phase). However, we note that this is not true for any subsequent iteration: updates in iteration 1 may utilize states of iteration 0, which is a state of the current phase rather than the previous one.

Our paper's main contribution is to extend the transition matrix work of [9]: any state from the current phase, rather than the previous state, may be represented as a convex combination of states from the previous phase. Specifically,

the state of node i at iteration r of the current phase may be represented as

$$\sum_{k=0}^{hD-1} M_{(rh+i)(k)} * v_k \tag{8}$$

We prove that matrix M is stochastic in both Case 1 and Case 2 in Appendix C.

4.3.2 Case 1: $f = 0$

Define matrix G to be the matrix constructed in Section 5.1.1 of [9], using our value of v_i as N_i^-. G is an $h \times h$ stochastic matrix.

Algorithm 2 is motivated by the fact that at iteration t, node i is receiving a state from every other node. In particular, node i receives the state of node j from iteration $t - dist(i, j)$, as this will always be the most up-to-date iteration of node j received by node i. We construct the matrix such that each state in N_i^* is given an equal weight, as defined in (2).

Since each node in N_i^* is honest, each of its states can either be described by an element in a transition matrix, or a row of values in the transition matrix.

4.3.3 Case 2: $f > 0$

Define matrix G to be the matrix constructed in Section 5.1.2 of [9], using our value of v_i as N_i^-. G is an $h \times h$ stochastic matrix.

This construction is motivated by the fact every state in N_i^* can we represented by a weighted average of two honest nodes, one in L^* and one in S^*. This allows even the behavior of byzantine nodes to be able to be represented in the transition matrix of honest states.

4.4 Validity Proof

The update in (2) of each node always results in a convex combination of some set of node states during some iterations. This means that any update will always stay within the convex hull of the set of initial input states, proving the validity condition.

4.5 Convergence Proof

4.5.1 Matrix Characteristics

Definition 4.1. Node i Row: row j of Matrix M is considered a Node i Row iff $j \bmod h \cong i$

A Node i Row represents the state update of node i during some iteration of the phase.

To characterize what the weights of matrix rows representing iterations after iteration 0, we introduce the following observation:

Theorem 1. *If an element of the transition matrix M_{ij} is a non-zero weight for $i < h$, then the value of M_{zj} is non-zero as well, $\forall z$ such that z is a Node i row and $z \geq h$*

Proof. Row i represents the matrix update of node i in iteration 0, while row z represents the matrix update of node i in any iteration $[1, D]$. We denote $z = kh + i$, for some integer $k < D$.

We proof the theorem with induction:

Base case ($k = 0$): If $k = 0$, then $z = i$. The theorem is trivially true.

Induction Step ($0 < k < D$): We noted previously that not every update can be exactly expressed as a convex combination of weights from the previous phase. Specifically, node i will always use the most up-to-date value of its own state, which is no longer a state of the previous phase.

However, this state may still be represented as a *combination* of weights of the previous phase: in iteration t, node i uses state of node i in iteration $t - 1$. Iteration $t - 1$ of phase a is represented by row $h(t - 1) + i$ of matrix M:

$$v_i(i) = M_{h(t-1)+i} * v[a] \tag{9}$$

Therefore, instead of using a single weight to denote the state of node i in iteration $t - 1$, we may instead add every single weight of row $h(t - 1) + i$ to row $ht + i$, scaled by a factor of $\frac{1}{hD}$. The induction step is completed by setting k as t. □

We now describe specific characteristics of which weights are non-zero in transition matrix M:

We first note the existence of a diagonal of non-zero values at the rightmost h columns of the transition matrix.

Theorem 2. *$\forall k, i$ such that $k, i \in \mathbb{R}$, $0 \leq k < D$ and $0 \leq i \leq h$, $M_{kh+i,h(D-1)+i}$ is a non-zero value.*

Proof. For $k = 0$, row $kh + i$ represents the update of node i during iteration 0, or the first iteration of the previous phase. Column $h(D - 1) + i$ represents the state of node i during iteration $D - 1$ of the previous phase, or the last iteration of the previous phase. Since each honest node always uses the state of the previous iteration in its update, this matrix value is non-zero.

For $k > 0$, the result generalizes from Theorem 1. □

Now we introduce some definitions relating to the network graph.

Definition 4.2. Complete Graph: A complete graph is a graph with vertex set V, and edge set E', such that $\forall i, j, i \neq j : (i, j) \in E'$. The graph contains b byzantine nodes, h honest nodes, and $\|V\| = m$.

A complete graph describes the de-facto communication during an entire phase: since the longest path between any two honest nodes is at most D, any two nodes may communicate with each other for at least one iteration during every single phase. We now introduce a graph that represents the network graph after the trimming in (2).

$$\begin{bmatrix} 0 & 0 & \frac{1}{3} & \mathbf{\frac{1}{3}} & \frac{1}{3} & 0 \\ 0 & 0 & 0 & \frac{1}{3} & \mathbf{\frac{1}{3}} & \frac{1}{3} \\ \frac{1}{3} & 0 & 0 & 0 & \frac{1}{3} & \mathbf{\frac{1}{3}} \\ 0 & 0 & \frac{1}{9} & \mathbf{\frac{2}{9}} & \frac{2}{9} & \frac{4}{9} \\ \frac{1}{9} & 0 & \frac{1}{9} & \frac{2}{9} & \mathbf{\frac{1}{3}} & \frac{2}{9} \\ \frac{1}{9} & 0 & 0 & \frac{4}{9} & \frac{2}{9} & \mathbf{\frac{2}{9}} \end{bmatrix}$$

Figure 1: A sample matrix illustrating Theorem 2, with $h = 3$ and $D = 2$ is shown.

Definition 4.3. Reduced Graph: A reduced graph is a complete graph with all nodes in set B removed, along with their incoming and outgoing edges. Additional, we remove any arbitrary set of b incoming edges from each remaining node. Note that there are several reduced graph for every complete graph, but only a finite number of them. Define R_f to be the set of all reduced graphs for a given complete graph, and define r as $\|R_f\|$. Note that this definition comes from [9].

Note that even though nodes do not have edges connecting to themselves, they can also "send" messages to themselves. Thus the adjacency matrix A of a reduced graph will always be non-zero at $A_{ii}, \forall i \in V$

The reduced graph represents the communication links between the entire graph after trimming is done.

We introduce one last theorem describing the qualities of transition matrix M.

Theorem 3. *Every Node i row contains a non-zero value in column z, where z mod h is a node with an incoming edge of node i in some reduced graph in R_f*

Proof. For rows t such that $\lfloor \frac{t}{h} \rfloor = 0$ (which correspond to the updates of node i in iteration 0), we note that every single node received some message from every other node from the previous phase. The reduced graph represents some form of trimming of all incoming information, where an incoming edge of the reduced graph represents an incoming state that is not trimmed. Column z mod $h \cong j$, column z represents a state of node j. The theorem is proven by induction.

Base Step The base step is proven in A

Induction Step For rows t such that $\lfloor \frac{t}{h} \rfloor > 0$, the result generalizes from Theorem 1 □

This theorem describes how every state update in the matrix is based off of at least one weight corresponding to each node in the graph.

4.5.2 Transition Matrix Behavior

To prove convergence, we show that a finite number of matrices in the transition matrix product forms a non-zero column in the stochastic matrix (scrambling matrix). This guarantees that the matrix $\lim_{T\to\infty} \prod_{t=0}^{T} M[t]$ has identical rows, which in turn ensures the convergence condition [9, 13].

To do this, we introduce a repeated matrix product $Q[t]$, which represents a repeated product of $2rD + 1$ matrices. Specifically, we define the following:

$$Q[t] = \prod_{i=t(2rD+1)}^{(t+1)(2rD+1)} M[i] \tag{10}$$

We may write our transition matrix update thus as:

$$v[T] = \prod_{t=0}^{T/(2rD+1)} Q[t] * v[0] \tag{11}$$

Now we introduce a new theorem that explains the behavior of transition matrices.

Definition 4.4. Matrix Inequality: We define matrices $A < B$ iff $\forall i, j : A_{ij} < B_{ij}$.

Theorem 4. *$\forall t$, matrix $Q[t]$ contains at least one non-zero column within the last n columns of the matrix.*

Proof. We introduce Lemma 5, which explains how the product of two transition matrices creates a $h \times h$ adjacency matrix of a reduced graph at the bottom right corner of the matrix.

Lemma 5. Define $M^1 * M^2$ to be two arbitrary transition matrices constructed from Algorithm 2. $M^1 * M^2[h(D-1)+1, hD : h(D-1)+1, hD] \geq \beta * r_f$, where r_f is the adjacency matrix of some reduced graph and β is some positive constant [9].

Proof. The proof is derived from Theorem 3. Within any row of matrix M^1, if there is a non-zero value in column z, where $z \bmod h \cong j$, then there will be a non-zero value in column $h(D-1)+j$ of matrix $M^1 * M^2$. This is due to diagonals of Theorem 2 in matrix M^2. □

We now use a key result of [9] to show the generation of a partial non-zero column in this bottom-right matrix.

Lemma 6. The product of $2rD$ arbitrary transition matrices (denoted as Z) results in a non-zero column of the matrix $Z[h(D-1)+1, hD : h(D-1)+1, hD]$.

$$\begin{bmatrix} 0 & \frac{1}{3} & \frac{1}{3} & \frac{1}{3} & 0 & 0 \\ 0 & 0 & 0 & \frac{1}{3} & \frac{1}{3} & \frac{1}{3} \\ \frac{1}{3} & 0 & \frac{1}{3} & 0 & 0 & \frac{1}{3} \\ 0 & 0 & \frac{1}{9} & \frac{2}{9} & \mathbf{\frac{2}{9}} & \frac{4}{9} \\ \frac{1}{9} & 0 & \frac{1}{9} & \frac{2}{9} & \mathbf{\frac{1}{3}} & \frac{2}{9} \\ \frac{1}{9} & 0 & 0 & \frac{4}{9} & \mathbf{\frac{2}{9}} & \frac{2}{9} \end{bmatrix}$$

Figure 2: A sample matrix Z with a non-zero column in the bottom rightmost $h \times h$ matrix.

$$\begin{bmatrix} 0 & 0 & \frac{1}{3} & \frac{1}{3} & \mathbf{\frac{1}{3}} & 0 \\ 0 & 0 & 0 & \frac{1}{3} & \mathbf{\frac{1}{3}} & \frac{1}{3} \\ \frac{1}{3} & 0 & 0 & 0 & \mathbf{\frac{1}{3}} & \frac{1}{3} \\ 0 & 0 & \frac{1}{9} & \frac{2}{9} & \mathbf{\frac{2}{9}} & \frac{4}{9} \\ \frac{1}{9} & 0 & \frac{1}{9} & \frac{2}{9} & \mathbf{\frac{1}{3}} & \frac{2}{9} \\ \frac{1}{9} & 0 & 0 & \frac{4}{9} & \mathbf{\frac{2}{9}} & \frac{2}{9} \end{bmatrix}$$

Figure 3: A sample matrix $Z * M$ with a non-zero column in the entire matrix in the same column as the partial non-zero column of Figure 2 above.

Proof. From Lemma 5, the product of $2rD$ transition matrices can be represented as a product of rD matrices, where each matrix has some arbitrary adjacency matrix of a reduced graph in its bottom rightmost $h \times h$ submatrix. Given that $\|R_f\| = r$, by the pigeonhole principle, we can conclude that at least one arbitrary reduced graph is repeated at least D times. In Appendix B, it is also proven that there exists a directed path from some node to all other nodes in all reduced graphs. Since the shortest directed path of between any two honest nodes is at most length D, it is proven in [9] that a non-zero column is formed in matrix $Z[h(D-1)+1, hD : h(D-1)+1, hD]$ □

Given that matrix Z has a non-zero column in the bottom rightmost $h \times h$ matrix, it can be shown that $Z * M$, where M is an arbitrary transition matrix, creates a non-zero column of the entire matrix in the same column where the $h \times h$ matrix column was. This is once again due to the diagonals of Theorem 2 and a simple application of linear algebra.

Given that Z represents the product of $2rD$ matrices and M represents a single transition matrix, we have shown that the product of $2rD + 1$ transition matrices creates a non-zero column. This concludes the proof of Theorem 4. □

Theorem 7. $\lim_{T\to\infty} v[T] = c * \mathbf{1}$*, where* c *is some constant and* $\mathbf{1}$ *is the column vector of ones. Note this proves the convergence condition.*

Proof.

$$\lim_{T\to\infty} v[T] =$$
$$\lim_{T\to\infty} \prod_{t=0}^{T} M[t] * v[0] =$$
$$\lim_{T\to\infty} \prod_{t=0}^{T/(2rD+1)} Q[t] * v[0]$$

From Theorem 4, we have shown that $\forall t$, matrix $Q[t]$ contains at least one non-zero column. Since $Q[t]$ is also stochastic, it is a scrambling matrix. [9] proves that the product of any infinite number of scrambling matrices converges to a matrix with identical rows. Thus the product $\prod_{t=0}^{T/(2rD+1)} Q[t] * v[0]$ results in a column vector with identical elements, proving the theorem. □

Acknowledgements

First of all, I would like to thank MIT and the MIT PRIMES program for giving me this wonderful opportunity to conduct research.

I'd like to thank Jun Wan, Prof. Lili Su, and Prof. Nitin Vaidya for all of our discussions. Finally, the biggest thanks goes out to my research mentor, Hanshen Xiao, for all of his support and guidance.

References

[1] Peva Blanchard et al. *Byzantine-Tolerant Machine Learning*. URL: `https://arxiv.org/pdf/1703.02757.pdf`. (accessed: 9.20.2020).

[2] Fan Chung and Linyuan Lu. *The Diameter of Sparse Random Graphs*. URL: `https://people.math.sc.edu/lu/papers/diameter.pdf`. (accessed: 9.27.2020).

[3] Miguel Correia. *Essentials of Blockchain Technology*. CRC Press, 2019. ISBN: 0367027712.

[4] Danny Dolev and H. Raymond Strong. *Authenticated Algorithms for Byzantine Agreement*. URL: `http://www2.imm.dtu.dk/courses/02220/2015/L12/DolevStrong83.pdf`. (accessed: 9.18.2020).

[5] Danny Dolev et al. *Reaching Approximate Agreement in the Presence of Faults*. URL: `https://groups.csail.mit.edu/tds/papers/Lynch/jacm86.pdf`. (accessed: 9.18.2020).

[6] Michael J. Fischer, Nancy A. Lynch, and Michael S. Paterson. *Impossibility of Distributed Consensus with One Faulty Process*. URL: `https://groups.csail.mit.edu/tds/papers/Lynch/jacm85.pdf`. (accessed: 9.20.2020).

[7] Leslie Lamport, Robert Shostak, and Marshall Pease. *The Byzantine Generals Problem.* URL: https://people.eecs.berkeley.edu/~luca/cs174/byzantine.pdf. (accessed: 9.18.2020).

[8] Lewis Tseng and Nitin Vaidya. *Iterative Approximate Byzantine Consensus in Arbitrary Directed Graphs.* URL: http://disc.ece.illinois.edu/publications/podc35-2012-vaidya.pdf. (accessed: 9.20.2020).

[9] Nitin Vaidya. *Matrix Representation of Iterative Approximate Byzantine Consensus in Directed Graphs.* URL: https://arxiv.org/abs/1203.1888. (accessed: 8.26.2020).

[10] Nitin Vaidya, Lewis Tseng, and Guanfeng Liang. *Iterative Approximate Byzantine Consensus in Arbitrary Directed Graphs - Part II: Synchronous and Asynchronous Systems.* URL: https://arxiv.org/abs/1202.6094. (accessed: 8.26.2020).

[11] Jun Wan et al. *Expected Constant Round Byzantine Broadcast under Dishonest Majority.* URL: https://eprint.iacr.org/2020/590.pdf. (accessed: 9.18.2020).

[12] Ye Wang and Roger Wattenhofer. *Asynchronous Byzantine Agreement in Incomplete Networks.* URL: https://arxiv.org/pdf/2005.12725.pdf. (accessed: 9.18.2020).

[13] Jacob Wolfowitz. *Products of Indecomposable, Aperiodic, Stochastic Matrices.* URL: https://www.ams.org/journals/proc/1963-014-05/S0002-9939-1963-0154756-3/S0002-9939-1963-0154756-3.pdf. (accessed: 9.23.2020).

[14] Zhixiong Yang and Waheed U. Bajwa. *BRIDGE: Byzantine-resilient Decentralized Gradient Descent.* URL: https://arxiv.org/abs/1908.08098. (accessed: 8.26.2020).

A Reduced Graph Proof

For each row i of an arbitrary transition matrix M, we seek to prove that at least $h - b + 1$ elements in M_i are lower bounded by some arbitrary positive constant β [9]. We do this through a proof by induction.

A.1 Base Step: $\lfloor \frac{t}{h} \rfloor = 0$

In Algorithm 2, a value of t such that $\lfloor \frac{t}{h} \rfloor = 0$ implies that row t models a node's update in iteration 1 of the phase. This also implies that $\forall k, iter < 0$.

$\forall k$, column j is a unique value. This implies that through each iteration of the for-loop over all nodes, a unique column is being considered. We now consider two additional cases (whether matrix G is Case 1 or Case 2 construction), and prove that they both satisfy the desired condition.

A.1.1 Case 1

$G_{ik} > \beta$ iff $k \in N_i^* \cup i$ [9]. Since $N_i^* \subset V$, and $\|N_i^* \cup i\| = m - 2b + 1 = h - b + 1$, we can conclude that at least $h - b + 1$ elements of M_i are lower-bounded by β.

A.1.2 Case 2

$G_{ik} > \beta$ iff $k \in (N_i^* \cap H) \cup i \cup L^* \cup S*$ [9]. In [9], it is shown that $(N_i^* \cap H) \cup i \cup L^* \cup S* = \|m \cap H\| - b - 1$. Since $(N_i^* \cap H) \cup i \cup L^* \cup S* \subset V$, and $\|m \cap H\| - b - 1 = h - b + 1$, we have concluded the prove of the base case.

A.2 Reduced Graph Inequality

Each node in a reduced graph has a total of $m - 2b$ incoming edges. When you include a nodes ability to communicate with itself, each row of the adjacency matrix of the reduced graph contains $m - 2b + 1 = h - b + 1$ ones. We now note that from Algorithm 2, M_{tj} is non-zero only if $(t \bmod h, j \bmod h) \in E$, and that at most one non-zero value exists on some column j for each value of $j \bmod h$. This concludes the proof of the base step of the theorem.

B Source Component Proof

We define a source component of a graph as a node that has a directed path to every other path in a graph. In this section we prove that any arbitrary reduced graph contains at least one source component.

A reduced graph is constructed by removing n incoming edges from each node of a fully connected directed graph of at least $2n + 1$ nodes. In this proof, we assume that the reduced graph has exactly $2n + 1$ nodes. The case where the number of nodes exceeds $2n + 1$ is a simple generalization.

In a fully connected graph of $2n + 1$ nodes, there are totally (2n+1)2n outgoing edges. Thus, after removing n incoming edges (which are also outgoing edges of some other arbitrary node) from each node, there still exists at least $(2n + 1)n$ outgoing edges left in the graph. By Pigeonhole Principle, at least one node in the reduced graph, let us denote it as v_0, has at least n outgoing edges.

Denote the set of all nodes with direct incoming edges from v_0 as set S. We know that $\|S\| \geq n$. For each of the nodes which are not in the set S (there are no more than n nodes not in S), it is noted that each of them has n incoming edges.

Assume that none of these nodes have incoming edges from v_0 or any node in set S. Thus, they can only have edges from at most a total of $2n+1-2-n = n-1$ nodes. However, it is known that all nodes have n incoming edges. This is a contradiction. Thus, they must either have one incoming edge from a node in S, or an incoming edge from v_0. Therefore we have proven that v_0 is a source component.

C Transition Matrix M is stochastic

We prove that M is stochastic through strong induction. Let t be an arbitrary row of matrix M. We prove that the sum of all elements in M_t is 1.

C.1 Base Case: $\lfloor \frac{t}{h} \rfloor = 0$

In Algorithm 2, a value of t such that $\lfloor \frac{t}{h} \rfloor = 0$ implies that row t models a node's update in iteration 0 of the phase. This also implies that $\forall k, iter < 0$.

Throughout the entire for-loop over Node $k \in V$, only summation step 6 is used. Each summation step adds G_{ik} to the value of $\sum_{k=0}^{hD-1} M_{tk}$, for some k. Thus the entire for-loop adds a total of

$$\sum_{k=0}^{m-1} G_{ik} \tag{12}$$

to $\sum_{k=0}^{m-1} G_{ik}$, split over multiple elements. Since $\sum_{k=0}^{m-1} G_{ik} = 1$, M_t is a stochastic vector.

C.2 Base Case: $\lfloor \frac{t}{h} \rfloor > 0$

Without loss of generality, assume that $\lfloor \frac{t}{h} \rfloor = \alpha$, for some $\alpha > 0$. By strong induction, we assume that M_g is a stochastic vector, for all $\lfloor \frac{g}{h} \rfloor < \alpha$.

Summation step (6) adds G_{ik} to the value of $\sum_{k=0}^{hD-1} M_{tk}$, for some k. However, since $\lfloor \frac{j}{h} \rfloor < \lfloor \frac{t}{h} \rfloor = \alpha$, we know that M_{jv} is a stochastic vector. Therefore summation step (7) adds G_{ik} to the value of $\sum_{k=0}^{hD-1} M_{tk}$, for some k. Therefore once again, the entire for-loop adds a total of

$$\sum_{k=0}^{m-1} G_{ik} \tag{13}$$

to $\sum_{k=0}^{m-1} G_{ik}$, split over multiple elements. Similar to the base case, this proves that M_t is a stochastic vector, completing the proof.