Iterative Approximate Byzantine Consensus under a Generalized Fault Model*

Lewis Tseng[1,3], and Nitin Vaidya[2,3]

[1] Department of Computer Science,
[2] Department of Electrical and Computer Engineering, and
[3] Coordinated Science Laboratory
University of Illinois at Urbana-Champaign
Email: {ltseng3, nhv}@illinois.edu

Abstract. In this work, we consider a *generalized* fault model [7,9,5] that can be used to represent a wide range of failure scenarios, including correlated failures and non-uniform node reliabilities. Under the generalized fault model, we explore iterative approximate Byzantine consensus (IABC) algorithms [14] in arbitrary directed networks. We prove a *tight* necessary and sufficient condition on the underlying communication graph for the existence of IABC algorithms.

We propose a new IABC algorithm for the generalized fault model, and present a *transition matrix*-based proof to show the correctness of the proposed algorithm. While transition matrices have been used to prove correctness of non-fault-tolerant consensus [6], this paper is the first to extend the technique to Byzantine fault-tolerant algorithms.

Keywords: iterative consensus, graph property, generalized fault model

1 Introduction

Dolev et al. [3] introduced the notion of *approximate Byzantine consensus* by relaxing the requirement of exact consensus [12]. The goal in approximate consensus is to allow the fault-free nodes to agree on values that are approximately equal to each other (and not necessarily exactly identical). The fault model assumed in much of the work on Byzantine consensus allows up to f Byzantine faulty nodes in the network. We will refer to this fault model as the "f-total" fault model [11,10,3,12]. In prior work, other fault models have been explored as well. For instance, in the "f-local" fault model, up to f neighbors of each node in the network may be faulty [8,1,11]. In this paper, we consider a *generalized* fault model (to be described in the next section), which is similar to the

* This research is supported in part by National Science Foundation award CNS 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.

fault model presented in [7,9,5]. The generalized fault model specifies a "fault domain", which is a collection of feasible fault sets. For example, in a system consisting of four nodes, namely, nodes $1, 2, 3$ and 4, the fault domain could be specified as $\mathcal{F} = \{ \{1\}, \{2, 3, 4\} \}$. Thus, in this case, either node 1 may be faulty, or any subset of nodes in $\{2, 3, 4\}$ may be faulty. However, node 1 may not be faulty together with another node in the same execution. This fault model is general in the sense that the other fault models, such as f-total, and f-local models, are special cases of the generalized fault model.

In this work, we consider "iterative" algorithms for achieving approximate Byzantine consensus in synchronous point-to-point networks that are modeled as arbitrary directed graphs. The *iterative approximate Byzantine consensus* (IABC) algorithms [14] of interest have the following properties:

- *Initial state* of each node is equal to a real-valued input provided to that node.
- *Memory-less*: the computation of new state at each node is based only on local information, i.e., node's own state and states from neighboring nodes.
- *Validity* condition: After each iteration of an IABC algorithm, the state of each fault-free node must remain in the convex hull of the states of the fault-free nodes at the end of the previous iteration.
- *Convergence* condition: For any $\epsilon > 0$, there exists an iteration r such that the states of the fault-free nodes are guaranteed to be within ϵ in every iteration $r' \geq r$.

1.1 Main Contributions

This paper is a generalization of our recent work on IABC algorithms under the f-total fault model [14]. There are two contributions of this paper:

- We identify a necessary (Section 4) condition on the underlying communication graph for the existence of a correct IABC algorithm under the generalized fault model. Moreover, we show that the necessary condition is also sufficient by introducing a new IABC algorithm for the generalized fault model (Section 5) that uses only "local" information.

- We present a *transition matrix* representation of the new IABC algorithm (Section 6). This representation is then used to prove the correctness of the proposed algorithm (Section 6.4). Transition matrices have been used previously to prove correctness of non-fault-tolerant consensus [6]. However, this paper is the first to develop transition matrix representation for Byzantine fault-tolerant consensus. We make the following observation: for a given evolution of the state vector corresponding to the state of the fault-free nodes, many alternate state transition matrices may potentially be chosen to emulate that evolution correctly. However, for any given state evolution, we can suitably "design" the transition matrices so that the classical results on matrix products can be applied to prove convergence of our algorithm in all networks that satisfy the necessary condition.

2 Models

Communication Model: The system is assumed to be synchronous. The communication network is modeled as a simple directed graph $G(\mathcal{V}, \mathcal{E})$, where $\mathcal{V} = \{1, \ldots, n\}$ is the set of n nodes, and $\mathcal{E}$ is the set of directed edges between the nodes in $\mathcal{V}$. We assume that $n \geq 2$, since the consensus problem for $n = 1$ is trivial. Node i can reliably transmit messages to node j if and only if the directed edge (i, j) is in $\mathcal{E}$. Each node can send messages to itself as well; however, for convenience, we exclude self-loops from set $\mathcal{E}$. That is, $(i, i) \notin \mathcal{E}$ for $i \in \mathcal{V}$. With a slight abuse of terminology, we will use the terms *edge* and *link* interchangeably in our presentation.

For each node i, let N_i^- be the set of nodes from which i has incoming edges. That is, $N_i^- = \{ j \mid (j, i) \in \mathcal{E} \}$. Similarly, define N_i^+ as the set of nodes to which node i has outgoing edges. That is, $N_i^+ = \{ j \mid (i, j) \in \mathcal{E} \}$. Nodes in N_i^- and N_i^+ are, respectively, said to be incoming and outgoing neighbors of node i. Since we exclude self-loops from $\mathcal{E}$, $i \notin N_i^-$ and $i \notin N_i^+$. However, we note again that each node can indeed send messages to itself.

Generalized Byzantine Fault Model: A Byzantine faulty node may misbehave arbitrarily. Possible misbehavior includes transmitting incorrect and mismatching (or inconsistent) messages to different neighbors. The faulty nodes may collaborate with each other. Moreover, the faulty nodes are assumed to have a complete knowledge of the execution of the algorithm, including the states of all the nodes, the algorithm specification, and the network topology.

The generalized fault model we consider is similar to fault models presented in [7,9,5]. The generalized fault model is characterized using *fault domain* $\mathcal{F} \subseteq 2^{\mathcal{V}}$ as follows: Nodes in set F may fail during an execution of the algorithm only if there exists set $F^* \in \mathcal{F}$ such that $F \subseteq F^*$. Set F is then said to be a *feasible* fault set.

Definition 1. *Set $F \subseteq \mathcal{V}$ is said to be a feasible fault set, if there exists $F^* \in \mathcal{F}$ such that $F \subseteq F^*$.*

Thus, each set in $\mathcal{F}$ specifies nodes that may all potentially fail during a single execution of the algorithm. This feature can be used to capture the notion of correlated failures. For example, consider a system consisting of four nodes, namely, nodes 1, 2, 3, and 4. Suppose that

$$\mathcal{F} = \{ \{1\}, \{2\}, \{3, 4\} \}.$$

This definition of $\mathcal{F}$ implies that during an execution either (i) node 1 may fail, (ii) node 2 may fail, or (iii) any subset of $\{3, 4\}$ may fail, and no other combination of nodes may fail (e.g., nodes 1 and 3 cannot both fail in a single execution). In this case, the reason that the set $\{3, 4\}$ is in the fault domain may be that the failures of nodes 3 and 4 are correlated.

The generalized fault model is also useful to capture variations in node reliability [7,9,5]. For instance, in the above example, nodes 1 and 2 may be more

reliable than nodes 3 and 4. Therefore, while nodes 3 and 4 may fail in the same execution, nodes 1 and 2 are less likely to fail together in the same execution. Therefore, $\{1, 2\} \notin \mathcal{F}$.

Local knowledge of $\mathcal{F}$: To implement our IABC Algorithm presented in Section 5, it is sufficient for each node i to know $N_i^- \cap F$, for each feasible fault set F. In other words, each node only needs to know the set of its incoming neighbors that may fail in the same execution of the algorithm. Thus, the iterative algorithm can be implemented using only "local" information regarding $\mathcal{F}$.

3 Iterative Approximate Byzantine Consensus (IABC) Algorithms

In this section, we describe the structure of the IABC algorithms of interest, and state the validity and convergence conditions that they must satisfy.

Each node i maintains state v_i, with $v_i[t]$ denoting the state of node i at the end of the t-th iteration of the algorithm. Initial state of node i, $v_i[0]$, is equal to the initial input provided to node i. At the start of the t-th iteration ($t > 0$), the state of node i is $v_i[t-1]$. The IABC algorithms of interest will require each node i to perform the following three steps in iteration t, where $t > 0$. Note that the faulty nodes may deviate from this specification.

1. *Transmit step:* Transmit current state, namely $v_i[t-1]$, on all outgoing edges and self-loop (to nodes in N_i^+ and node i itself).
2. *Receive step:* Receive values on all incoming edges and self-loop (from nodes in N_i^- and itself). If the node does not receive the message from an incoming neighbor, the message value is assumed to be equal to some *default* value. Denote by $r_i[t]$ the vector of values received by node i from its incoming neighbors and itself. The size of vector $r_i[t]$ is $|N_i^-| + 1$.
3. *Update step:* Node i updates its state using a transition function Z_i as follows. Z_i is a part of the specification of the algorithm, and takes the vector $r_i[t]$ as the input.

$$v_i[t] = Z_i \ (\ r_i[t] \ ) \tag{1}$$

The following conditions must be satisfied by an IABC algorithm when the set of faulty nodes (in a given execution) is F:

- *Validity:* $\forall t > 0$, and all fault-free nodes $i \in \mathcal{V} - F$,
  $v_i[t] \geq \min_{j \in \mathcal{V}-F} \ v_j[t-1]$ and $v_i[t] \leq \max_{j \in \mathcal{V}-F} \ v_j[t-1]$.[1]
- *Convergence:* for all *fault-free* nodes $i, j \in \mathcal{V} - F$, $\lim_{t \to \infty} \ (v_i[t] - v_j[t]) = 0$

An IABC algorithm is said to be correct if it satisfies the above validity and convergence conditions in the given graph $G(\mathcal{V}, \mathcal{E})$. For a given fault domain $\mathcal{F}$ for graph $G(\mathcal{V}, \mathcal{E})$, the objective here is to identify the necessary and sufficient conditions for the existence of a correct IABC algorithm.

[1] For sets X and Y, $X - Y$ contains elements that are in X but not in Y. That is, $X - Y = \{i \mid i \in X, \ i \notin Y\}$.

4 Necessary Condition

In this section, we develop a necessary condition for the existence of a correct IABC algorithm. The necessary condition will be proved to be also sufficient in Section 6.

4.1 Preliminaries

To facilitate the statement of the necessary condition, we first introduce the notions of "source component" and "reduced graph" using the following three definitions.

Definition 2. Graph Decomposition: *Let H be a directed graph. Partition graph H into strongly connected components, $H_1, H_2, \cdots, H_h$, where h is a non-zero integer dependent on graph H, such that*

- *every pair of nodes within the same strongly connected component has directed paths in H to each other, and*
- *for each pair of nodes, say i and j, that belong to two different strongly connected components, either i does not have a directed path to j in H, or j does not have a directed path to i in H, or both.*

Construct a graph H^d wherein each strongly connected component H_k above is represented by vertex c_k, and there is an edge from vertex c_k to vertex c_l if and only if the nodes in H_k have directed paths in H to the nodes in H_l. H^d is called the decomposition graph of H.

It is known that for any directed graph H, the corresponding decomposition graph H^d is a directed acyclic graph (DAG) [2].

Definition 3. Source Component*: Let H be a directed graph, and let H^d be its decomposition graph as per Definition 2. Strongly connected component H_k of H is said to be a source component if the corresponding vertex c_k in H^d is not reachable from any other vertex in H^d.*

Definition 4. Reduced Graph: *For a given graph $G(\mathcal{V}, \mathcal{E})$ and a feasible fault set F, a reduced graph $G_F(\mathcal{V}_F, \mathcal{E}_F)$ is obtained as follows:*

- *Node set is obtained as $\mathcal{V}_F = \mathcal{V} - F$.*
- *For each node $i \in \mathcal{V}_F$, a feasible fault set $F_x(i)$ is chosen, and then the edge set $\mathcal{E}_F$ is obtained as follows:*
  - *remove from $\mathcal{E}$ all the links incident on the nodes in F, i.e., all the incoming and outgoing links of nodes in F, and*
  - *for each $j \in F_x(i) \cap \mathcal{V}_F \cap N_i^-$, remove link (j, i) from $\mathcal{E}$.*

  Feasible fault sets $F_x(i)$ and $F_x(j)$ chosen for $i \neq j$ may or may not be identical.

Note that for a given $G(\mathcal{V}, \mathcal{E})$ and a given F, multiple reduced graphs G_F may exist, depending on the choice of F_x sets above.

4.2 Necessary Condition

For a correct IABC algorithm to exist, the network graph $G(\mathcal{V}, \mathcal{E})$ must satisfy the necessary condition stated in Theorem 1 below.

Theorem 1. *Suppose that a correct IABC algorithm exists for $G(\mathcal{V}, \mathcal{E})$. Then, any reduced graph G_F, corresponding to any feasible fault set F, must contain exactly one source component.*

Proof Sketch: A complete proof is presented in our technical report [13].The proof is by contradiction. Let us assume that a correct IABC algorithm exists, and for some feasible fault set F, and feasible sets $F_x(i)$ for each $i \in \mathcal{V} - F$, the resulting reduced graph contains two source components. Let L and R denote the nodes in the two source components, respectively. Thus, L and R are disjoint and non-empty. Let $C = (\mathcal{V} - F - L - R)$ be the remaining nodes in the reduced graph. C may or may not be non-empty. Assume that the nodes in F (if non-empty) are all faulty, and all the nodes in L, R, and C (if non-empty) are fault-free. Suppose that each node in L has initial input equal to m, each node in R has initial input equal to M, where $M > m$, and each node in C has an input in the range $[m, M]$. As elaborated in [13], the faulty nodes can behave in such a manner that, in each iteration, nodes in L and R are forced to maintain their updated state equal to m and M, respectively, so as to satisfy the validity condition. This ensures that, no matter how many iterations are performed, the convergence condition cannot be satisfied. □

5 Algorithm 1

We will prove that there exists an IABC algorithm – particularly *Algorithm 1* below – that satisfies the validity and convergence conditions provided that the graph $G(\mathcal{V}, \mathcal{E})$ satisfies the necessary condition in Theorem 1. This implies that the necessary condition in Theorem 1 is also sufficient. *Algorithm 1* has the three-step structure described in Section 3. This algorithm is a generalization – to accommodate the generalized fault model – of iterative algorithms that were analyzed in prior work [3,12,11], including in our own prior work as well [14]. The key difference from previous algorithms is in the *Update* step below. Now, we describe the steps need to be followed by all the fault-free nodes in iteration t ($t > 0$).

Algorithm 1

1. *Transmit step:* Transmit current state $v_i[t-1]$ on all outgoing edges and self-loop.
2. *Receive step:* Receive values on all incoming edges and self-loop. These values form vector $r_i[t]$ of size $|N_i^-|+1$ (including the value from node i itself). When a fault-free node expects to receive a message from an incoming neighbor but does not receive the message, the message value is assumed to be equal to its own state, i.e., $v_i[t-1]$.
3. *Update step:* Sort the values in $r_i[t]$ in an increasing order (breaking ties arbitrarily). Let D be a vector of nodes arranged in an order "consistent" with $r_i[t]$: specifically, $D(1)$ is the node that sent the smallest value in $r_i[t]$, $D(2)$ is the node that sent the second smallest value in $r_i[t]$, and so on. The size of vector D is also $|N_i^-|+1$.
   From vector $r_i[t]$, eliminate the smallest f_1 values, and the largest f_2 values, where f_1 and f_2 are defined as follows:
   - f_1 is the largest number such that there exists a feasible fault set $F' \subseteq N_i^-$ containing nodes $D(1), D(2), ..., D(f_1)$. Recall that $i \notin N_i^-$.
   - f_2 is the largest number such that there exists a feasible fault set $F'' \subseteq N_i^-$ containing nodes $D(|N_i^-|-f_2+2), D(|N_i^-|-f_2+3), ..., D(|N_i^-|+1)$.

   F' and F'' above may or may not be identical.
   Let $N_i^*[t]$ denote the set of nodes from whom the remaining $|N_i^-|+1-f_1-f_2$ values in $r_i[t]$ were received, and let w_j denote the value received from node $j \in N_i^*[t]$. Note that $i \in N_i^*[t]$. Hence, for convenience, define $w_i = v_i[t-1]$ to be the value node i receives from itself. Observe that if $j \in N_i^*[t]$ is fault-free, then $w_j = v_j[t-1]$.
   Define

$$v_i[t] \;=\; Z_i(r_i[t]) \;=\; \sum_{j \in N_i^*[t]} a_i \, w_j \tag{2}$$

   where

$$a_i = \frac{1}{|N_i^*[t]|} = \frac{1}{|N_i^-|+1-f_1-f_2}$$

   The "weight" of each term on the right-hand side of (2) is a_i, and these weights add to 1. Also, $0 < a_i \leq 1$. Although f_1, f_2 and a_i may be different for each iteration t, for simplicity, we do not explicitly represent this dependence on t in the notations.

Observe that $f_1 + f_2$ nodes whose values are eliminated in the *Update* step above are all in N_i^-. Thus, the above algorithm can be implemented by node i if it knows which of its incoming neighbors may fail in a single execution of the algorithm; node i does not need to know the entire fault domain $\mathcal{F}$ as such.

The main difference between the Algorithm 1 and IABC algorithms in prior work is in the choice of the values eliminated from vector $r_i[t]$ in the *Update* step. The manner in which the values are eliminated ensures that the values received

from nodes $D(f_1+1)$ and $D(|N_i^-|-f_2+1)$ (i.e., the smallest and largest values that survive in $r_i[t]$) are within the convex hull of the state of fault-free nodes, even if nodes $D(f_1+1)$ and $D(|N_i^-|-f_2+1)$ may not be fault-free. This property is useful in proving algorithm correctness (as discussed below).

6 Sufficiency: Correctness of Algorithm 1

We will show that Algorithm 1 satisfies validity and convergence conditions, provided that $G(\mathcal{V},\mathcal{E})$ satisfies the condition below, which matches the necessary condition stated in Theorem 1.

Sufficient condition: *Any reduced graph G_F corresponding to any feasible fault set F contains exactly one* source component*.*

In the rest of this section, we assume that $G(\mathcal{V},\mathcal{F})$ satisfies the above condition. We first introduce some standard matrix tools to facilitate our proof. Then, we develop a transition matrix representation of the *Update* step in Algorithm 1, and show how to use these tools to prove the correctness of Algorithm 1 in $G(\mathcal{V},\mathcal{F})$.

When presenting matrix products, for convenience of presentation, we adopt the "backward" product convention below, where $a \leq b$,

$$\Pi_{i=a}^{b}\mathbf{A}[i] = \mathbf{A}[b]\mathbf{A}[b-1]\cdots\mathbf{A}[a] \tag{3}$$

6.1 Matrix Preliminaries

In the discussion below, we use boldface upper case letters to denote matrices, rows of matrices, and their elements. For instance, $\mathbf{A}$ denotes a matrix, $\mathbf{A}_i$ denotes the i-th row of matrix $\mathbf{A}$, and $\mathbf{A}_{ij}$ denotes the element at the intersection of the i-th row and the j-th column of matrix $\mathbf{A}$.

Definition 5. *A vector is said to be stochastic if all the elements of the vector are non-negative, and the elements add up to 1. A matrix is said to be row stochastic if each row of the matrix is a stochastic vector.*

For a row stochastic matrix $\mathbf{A}$, coefficients of ergodicity $\delta(\mathbf{A})$ and $\lambda(\mathbf{A})$ are defined as follows [15]:

$$\delta(\mathbf{A}) = \max_j \max_{i_1,i_2} |\mathbf{A}_{i_1 j} - \mathbf{A}_{i_2 j}|$$

$$\lambda(\mathbf{A}) = 1 - \min_{i_1,i_2} \sum_j \min(\mathbf{A}_{i_1 j}, \mathbf{A}_{i_2 j})$$

It is easy to show that $0 \leq \delta(\mathbf{A}) \leq 1$ and $0 \leq \lambda(\mathbf{A}) \leq 1$, and that the rows of $\mathbf{A}$ are all identical if and only if $\delta(\mathbf{A}) = 0$. Also, $\lambda(\mathbf{A}) = 0$ if and only if $\delta(\mathbf{A}) = 0$.

The next result from [4] establishes a relation between the coefficient of ergodicity $\delta(\cdot)$ of a product of row stochastic matrices, and the coefficients of ergodicity $\lambda(\cdot)$ of the individual matrices defining the product.

Lemma 1. *For any p square row stochastic matrices $\mathbf{A}(1), \mathbf{A}(2), \ldots \mathbf{A}(p)$,*

$$\delta(\mathbf{A}(p)\mathbf{A}(p-1)\cdots\mathbf{A}(1)) \;\leq\; \Pi_{i=1}^{p}\; \lambda(\mathbf{A}(i)).$$

Lemma 1 is proved in [4]. It implies that if, for all i, $\lambda(\mathbf{A}(i)) \leq 1-\gamma$ for some γ, where $0 < \gamma \leq 1$, then $\delta(\mathbf{A}(p)\mathbf{A}(p-1)\cdots\mathbf{A}(1))$ will approach zero as p approaches ∞. We now define a *scrambling* matrix [4,15].

Definition 6. *A row stochastic matrix $\mathbf{A}$ is said to be a scrambling matrix if*

$$\lambda(\mathbf{A}) < 1$$

The following lemma follows easily from the above definition of $\lambda(\cdot)$.

Lemma 2. *If any column of a row stochastic matrix $\mathbf{A}$ contains only non-zero elements that are all lower bounded by some constant γ, where $0 < \gamma \leq 1$, then $\mathbf{A}$ is a scrambling matrix, and $\lambda(\mathbf{A}) \leq 1-\gamma$.*

6.2 Transition Matrix Representation

In our discussion below, $\mathbf{M}[t]$ is a square matrix, $\mathbf{M}_i[t]$ is the i-th row of the matrix, and $\mathbf{M}_{ij}[t]$ is the element at the intersection of the i-th row and j-th column of $\mathbf{M}[t]$.

For a given execution of Algorithm 1, let F denote the actual set of faulty nodes in that execution. Let $|F| = \psi$. Without loss of generality, suppose that nodes 1 through $(n-\psi)$ are fault-free, and if $\psi > 0$, nodes $(n-\psi+1)$ through n are faulty. Denote by $v[0]$ the column vector consisting of the initial states of all the fault-free nodes. Denote by $v[t]$, where $t \geq 1$, the column vector consisting of the states of all the fault-free nodes at the end of the t-th iteration. The i-th element of vector $v[t]$ is state $v_i[t]$. The size of vector $v[t]$ is $(n-\psi)$.

We will show that the iterative update of the state of a fault-free node i ($1 \leq i \leq n-\psi$) performed in (2) in Algorithm 1 can be expressed using the matrix form below.

$$v_i[t] = \mathbf{M}_i[t]\; v[t-1] \tag{4}$$

where $\mathbf{M}_i[t]$ is a stochastic row vector of size $n-\psi$. That is, $\mathbf{M}_{ij}[t] \geq 0$, for $1 \leq j \leq n-\psi$, and $\sum_{1\leq j\leq n-\psi} \mathbf{M}_{ij}[t] = 1$.[2] By "stacking" (4) for different i, $1 \leq i \leq n-\psi$, we will represent the *Update* step of Algorithm 1 at all the fault-free nodes together using (5) below.

$$v[t] = \mathbf{M}[t]\; v[t-1] \tag{5}$$

where $\mathbf{M}[t]$ is a $(n-\psi)\times(n-\psi)$ row stochastic matrix, with its i-th row being equal to $\mathbf{M}_i[t]$ in (4). $\mathbf{M}[t]$ is said to be a transition matrix.

[2] In addition to t, the row vector $\mathbf{M}_i[t]$ may depend on the state vector $v[t-1]$ as well as the behavior of the faulty nodes in F. For simplicity, the notation $\mathbf{M}_i[t]$ does not explicitly represent this dependence.

By repeated application of (5), we can represent the *Update* step of Algorithm 1 at the t-th iterations ($t \geq 1$) as:

$$v[t] = \left(\Pi_{k=1}^{t} \mathbf{M}[k] \right) v[0] \tag{6}$$

Recall that we adopt the "backward" product convention as presented in (3).

In the rest of this section, we will first "construct" transition matrices $\mathbf{M}[k]$ ($1 \leq k \leq t$) that satisfy certain desirable properties. Then, we will identify a connection between these transition matrices and the sufficiency condition stated above, and use this connection to establish convergence property for Algorithm 1. The validity property also follows from the transition matrix representation.

6.3 Construction of the Transition Matrix

We will construct a transition matrix with the property described in Lemma 3 below.

Lemma 3. *The* Update *step of Algorithm 1 at the fault-free nodes can be expressed using row stochastic transition matrix* $\mathbf{M}[t]$*, such that there exists a feasible fault set* $F_x(i)$ *for each* $i \in \mathcal{V} - F$*, and for all* $j \in \{i\} \cup ((\mathcal{V}_F - F_x(i)) \cap N_i^-)$*,*

$$\mathbf{M}_{ij}[t] \ \geq \ \beta$$

where β *is a constant (to be defined later), and* $0 < \beta \leq 1$*.*

Proof. We prove the correctness of Lemma 3 by constructing $\mathbf{M}_i[t]$ for $1 \leq i \leq n - \psi$ that satisfies the conditions in Lemma 3. Recall that F is the set of faulty nodes, and $|F| = \psi$. As stated before, without loss of generality, nodes 1 through $n - \psi$ are assumed to be fault-free, and the remaining ψ nodes faulty.

Consider a fault-free node i performing the *Update* step in Algorithm 1. In the *Update* step, recall that the smallest f_1 and the largest f_2 values are eliminated from $r_i[t]$, where the choice of f_1 and f_2 is described in Algorithm 1. Let us denote by $\mathcal{S}$ and $\mathcal{L}$, respectively, the set of nodes[3] from whom the smallest f_1 and the largest f_2 values were received by node i in iteration t. Define sets $\mathcal{S}_g$ and $\mathcal{L}_g$ to be subsets of $\mathcal{S}$ and $\mathcal{L}$ that contain all the fault-free nodes in $\mathcal{S}$ and $\mathcal{L}$, respectively. That is, $\mathcal{S}_g = \mathcal{S} \cap (\mathcal{V} - F)$ and $\mathcal{L}_g = \mathcal{L} \cap (\mathcal{V} - F)$.

Construction of $\mathbf{M}_i[t]$ differs somewhat depending on whether sets $\mathcal{S}_g, \mathcal{L}_g$ and $N_i^*[t] \cap F$ are empty or not. We divide the possibilities into 6 cases. Due to space limitation, here we present the construction for one of the cases (named Case I). The construction for the remaining cases is presented in [13].

In Case I, $\mathcal{S}_g \neq \emptyset, \mathcal{L}_g \neq \emptyset$, and $N_i^*[t] \cap F \neq \emptyset$. Let $m_{\mathcal{S}}$ and $m_{\mathcal{L}}$ be defined as shown below. Recall that the nodes in $\mathcal{S}_g$ and $\mathcal{L}_g$ are all fault-free, and therefore, for any node $j \in \mathcal{S}_g \cup \mathcal{L}_g$, $w_j = v_j[t-1]$ (in the notation of Algorithm 1).

$$m_{\mathcal{S}} = \frac{\sum_{j \in \mathcal{S}_g} v_j[t-1]}{|\mathcal{S}_g|} \qquad \text{and} \qquad m_{\mathcal{L}} = \frac{\sum_{j \in \mathcal{L}_g} v_j[t-1]}{|\mathcal{L}_g|}$$

[3] Although $\mathcal{S}$ and $\mathcal{L}$ may be different for each t, for simplicity, we do not explicitly represent this dependence on t in the notations $\mathcal{S}$ and $\mathcal{L}$.

Now, consider any node $k \in N_i^*[t]$. By the definition of sets $\mathcal{S}_g$ and $\mathcal{L}_g$, $m_{\mathcal{S}} \leq w_k \leq m_{\mathcal{L}}$. Therefore, we can find weights $S_k \geq 0$ and $L_k \geq 0$ such that $S_k + L_k = 1$, and

$$w_k = S_k \; m_{\mathcal{S}} + L_k \; m_{\mathcal{L}} \tag{7}$$

$$= \frac{S_k}{|\mathcal{S}_g|} \sum_{j \in \mathcal{S}_g} v_j[t-1] + \frac{L_k}{|\mathcal{L}_g|} \sum_{j \in \mathcal{L}_g} v_j[t-1] \tag{8}$$

Clearly, at least one of S_k and L_k must be $\geq 1/2$. We now define elements $\mathbf{M}_{ij}[t]$ of row $\mathbf{M}_i[t]$:

- For $j \in N_i^*[t] \cap (\mathcal{V} - F)$: In this case, j is either a fault-free incoming neighbor of i, or i itself. For each such j, define $\mathbf{M}_{ij}[t] = a_i$. This is obtained by observing in (2) that the contribution of such a node j to the new state $v_i[t]$ is $a_i \; w_j = a_i \; v_j[t-1]$.
  The elements of $\mathbf{M}_i[t]$ defined here add up to
  $$|N_i^*[t] \cap (\mathcal{V} - F)| \; a_i$$
- For $j \in \mathcal{S}_g \cup \mathcal{L}_g$: In this case, j is a fault-free node in $\mathcal{S}$ or $\mathcal{L}$.
  For each $j \in \mathcal{S}_g$,
  $$\mathbf{M}_{ij}[t] \; = \; a_i \sum_{k \in N_i^*[t] \cap F} \frac{S_k}{|\mathcal{S}_g|}$$
  and for each node $j \in \mathcal{L}_g$,
  $$\mathbf{M}_{ij}[t] \; = \; a_i \sum_{k \in N_i^*[t] \cap F} \frac{L_k}{|\mathcal{L}_g|}$$
  To obtain these two expressions, we represent value w_k sent by each faulty node k in $N_i^*[t]$, i.e., $k \in N_i^*[t] \cap F$, using (8). Recall that this node k contributes $a_i w_k$ to (2). The above two expressions are then obtained by summing (8) over all the faulty nodes in $N_i^*[t] \cap F$, and replacing this sum by equivalent contributions by nodes in $\mathcal{S}_g$ and $\mathcal{L}_g$.
  The elements of $\mathbf{M}_i[t]$ defined here add up to
  $$a_i \sum_{k \in N_i^*[t] \cap F} (S_k + L_k) = |N_i^*[t] \cap F| \; a_i.$$
- For $j \in (\mathcal{V} - F) - (N_i^*[t] \cup \mathcal{S}_g \cup \mathcal{L}_g)$: These fault-free nodes have not yet been considered above. For each such node j, define $\mathbf{M}_{ij}[t] = 0$.

With the above definition of $\mathbf{M}_i[t]$, it should be easy to see that $\mathbf{M}_i[t] \, v[t-1]$ is, in fact, identical to $v_i[t]$ obtained using (2). Thus, the above construction of $\mathbf{M}_i[t]$ results in the contribution of the faulty nodes in $N_i^*[t]$ to (2) being replaced by an equivalent contribution from fault-free nodes in $\mathcal{L}_g$ and $\mathcal{S}_g$.

Properties of $\boldsymbol{M}_i[t]$: First, we show that $\mathbf{M}[t]$ is row stochastic. Observe that all the elements of $\mathbf{M}_i[t]$ are non-negative. Also, all the elements of $\mathbf{M}_i[t]$ above add up to

$$|N_i^*[t] \cap (\mathcal{V} - F)|\ a_i + |N_i^*[t] \cap F|\ a_i = |N_i^*[t]|\ a_i = 1$$

because $a_i = 1/|N_i^*[t]|$ as defined in Algorithm 1. Thus, $\mathbf{M}_i[t]$ is a stochastic row vector.

Recall that from the above discussion, for $k \in N_i^*[t]$, one of S_k and L_k must be $\geq 1/2$. Without loss of generality, assume that $S_s \geq 1/2$ for some node $s \in N_i^*[t] \cap F$. Consequently, for each node $j \in \mathcal{S}_g$, $\mathbf{M}_{ij}[t] \geq \frac{a_i}{|\mathcal{S}_g|} S_s \geq \frac{a_i}{2|\mathcal{S}_g|}$. Also, for each fault-free node j in $N_i^*[t]$, $\mathbf{M}_{ij}[t] = a_i$. Thus, if β is chosen such that

$$0 < \beta \leq \frac{a_i}{2|\mathcal{S}_g|} \tag{9}$$

and $F_x(i)$ is defined to be equal to $\mathcal{L}$, then the condition in the lemma holds for node i. That is, $\mathbf{M}_{ij}[t] \geq \beta$ for $j \in \{i\} \cup ((\mathcal{V}_F - F_x(i)) \cap N_i^-)$.

All Cases Together: Using similar constructions in other cases as well (presented in [13]) and a suitable choice of β (presented in [13]), we can obtain a row stochastic matrix $\mathbf{M}[t]$, and for each $i \in \mathcal{V} - F$ identify a feasible fault set $F_x(i)$, such that $\mathbf{M}_{ij}[t] \geq \beta$ for all $j \in \{i\} \cup ((\mathcal{V}_F - F_x(i)) \cap N_i^-)$. Thus, Lemma 3 can be proved correct. □

6.4 Validity and Convergence of Algorithm 1

Correspondence between $\mathbf{M}[t]$ and a Reduced Graph: Let R_F denote the set of all the reduced graphs of $G(\mathcal{V}, \mathcal{E})$ corresponding to a feasible fault set F. Let $\tau = |R_F|$. τ depends on F and the underlying network, and is finite.

In discussion below, let us denote a reduced graph by an italic upper case letter, and the corresponding "adjacency matrix" (defined below) using the same letter in boldface upper case. Thus, $\mathbf{H}$ denotes the adjacency matrix for graph $H \in R_F$.

Non-zero elements of adjacency matrix $\mathbf{H}$ are defined as follows: (i) for $1 \leq i, j \leq n - \psi$, $\mathbf{H}_{ij} = 1$ if and only if $(j, i) \in H$, and (ii) $\mathbf{H}_{ii} = 1$ for $1 \leq i \leq n - \psi$. That is, non-zero elements of row $\mathbf{H}_i$ correspond to the incoming links at node i, and the self-loop at node i. Thus, the adjacency matrix for any reduced graph in R_F has a non-zero diagonal.

Based on the sufficient condition stated at the start of Section 6 and Lemma 3, we can show the following key lemmas.

Lemma 4. *For any $H \in R_F, \mathbf{H}^{\mathbf{n}-\psi}$ has at least one non-zero column.*

Proof. $G(\mathcal{V}, \mathcal{E})$ satisfies the sufficient condition stated at the start of Section 6. Therefore, there exists at least one non-faulty node k in the reduced graph H that has directed paths to all the nodes in H (consisting of the edges in H). Since

the length of the path from k to any other node in H is at most $n-\psi-1$, the k-th column of matrix $\mathbf{H}^{n-\psi}$ will be non-zero.[4] □

Definition 7. *For matrices $\boldsymbol{A}$ and $\boldsymbol{B}$ of identical size, and a scalar γ, $\gamma\boldsymbol{B} \leq \boldsymbol{A}$ provided that $\gamma\boldsymbol{B}_{ij} \leq \boldsymbol{A}_{ij}$ for all i, j.*

Lemma 5. *For any $t \geq 1$, there exists a graph $H \in R_F$ such that $\beta\mathbf{H} \leq \mathbf{M}[t]$.*

Proof. Observe that the i-th row of the transition matrix $\mathbf{M}[t]$ corresponds to the state update (in Algorithm 1) performed at fault-free node i. Recall from Lemma 3 that $\mathbf{M}_{ij}[t] \geq \beta$ for $j \in \{i\} \cup ((\mathcal{V}_F - F_x(i)) \cap N_i^-)$, where $F_x(i)$ is a feasible fault set.

Let us obtain a reduced graph H by choosing $F_x(i)$ for each i as defined in Lemma 3. Then from the definition of adjacency matrix $\mathbf{H}$, Lemma 5 then follows. □

Correctness of Algorithm 1: The rest of the proof below is inspired by related work on non-fault-tolerant consensus [6].

Let $\mathbf{H}[t]$ denote the matrix $\mathbf{H}$ corresponding to $\mathbf{M}[t]$ as defined in Lemma 5.

Lemma 6. *For any $z \geq 1$, in the product below of $\mathbf{H}[t]$ matrices for consecutive $\tau(n-\psi)$ iterations, at least one column is non-zero.*

$$\mathit{\Pi}_{t=z}^{z+\tau(n-\psi)-1} \mathbf{H}[t]$$

Proof. Since the above product consists of $\tau(n-\psi)$ adjacency matrices corresponding to graphs in $R_{\mathcal{F}}$, at least one of the adjacency matrices corresponding to the τ distinct graphs in $R_{\mathcal{F}}$, say matrix $\mathbf{H}_*$, will appear in the above product at least $n-\psi$ times.

Now observe that: (i) By Lemma 4, $\mathbf{H}_*^{n-\psi}$ contains a non-zero column, say the k-th column is non-zero, and (ii) all the $\mathbf{H}[t]$ matrices in the product contain a non-zero diagonal. These two observations together imply that the k-th column in the above product is non-zero. □

Let us now define a sequence of matrices $\mathbf{Q}(i)$, $i \geq 1$, such that each of these matrices is a product of $\tau(n-\psi)$ of the $\mathbf{M}[t]$ matrices. Specifically,

$$\mathbf{Q}(i) = \mathit{\Pi}_{t=(i-1)\tau(n-\psi)+1}^{i\tau(n-\psi)} \mathbf{M}[t] \tag{10}$$

From (6) and (10) observe that

$$v[k\tau(n-\psi)] = \left(\mathit{\Pi}_{i=1}^{k} \mathbf{Q}(i) \right) \; v[0] \tag{11}$$

[4] That is, all the elements of the column will be non-zero. Also, such a non-zero column will exist in $\mathbf{H}^{n-\psi-1}$, too. We use the loose bound of $n-\psi$ to simplify the presentation.

Lemma 7. *For $i \geq 1$, $\mathbf{Q}(i)$ is a scrambling row stochastic matrix, and*

$$\lambda(\mathbf{Q}(i)) \leq 1 - \beta^{\tau(n-\psi)}.$$

Proof. $\mathbf{Q}(i)$ is a product of row stochastic matrices ($\mathbf{M}[t]$); therefore, $\mathbf{Q}(i)$ is row stochastic. From Lemma 5, for each $t \geq 1$,

$$\beta\, \mathbf{H}[t] \;\leq\; \mathbf{M}[t]$$

Therefore,

$$\beta^{\tau(n-\psi)}\; \mathit{\Pi}_{t=(i-1)\tau(n-\psi)+1}^{i\tau(n-\psi)}\; \mathbf{H}[t] \;\leq\; \mathit{\Pi}_{t=(i-1)\tau(n-\psi)+1}^{i\tau(n-\psi)}\; \mathbf{M}[t] \;=\; \mathbf{Q}(i)$$

By using $z = (i-1)(n-\psi)+1$ in Lemma 6, we conclude that the matrix product on the left side of the above inequality contains a non-zero column. Therefore, $\mathbf{Q}(i)$ on the right side of the inequality also contains a non-zero column.

Observe that $\tau(n-\psi)$ is finite, and hence, $\beta^{\tau(n-\psi)}$ is non-zero. Since the non-zero terms in $\mathbf{H}[t]$ matrices are all 1, the non-zero elements in $\mathit{\Pi}_{t=(i-1)\tau(n-\psi)+1}^{i\tau(n-\psi)}\mathbf{H}[t]$ must each be ≥ 1. Therefore, there exists a non-zero column in $\mathbf{Q}(i)$ with all the elements in the column being $\geq \beta^{\tau(n-\psi)}$. Therefore, by Lemma 2, $\lambda(\mathbf{Q}(i)) \leq 1 - \beta^{\tau(n-\psi)}$, and $\mathbf{Q}(i)$ is a scrambling matrix. □

Theorem 2. *Suppose that $G(\mathcal{V}, \mathcal{E})$ satisfies the sufficient condition stated above. Algorithm 1 satisfies both the validity and convergence conditions.*

Proof. Since $v[t] = \mathbf{M}[t]\, v[t-1]$, and $\mathbf{M}[t]$ is a row stochastic matrix, it follows that Algorithm 1 satisfies the validity condition.

Using Lemma 1 and the definition of $\mathbf{Q}(i)$, and using the inequalities $\lambda(\mathbf{M}[t]) \leq 1$ and $\lambda(\mathbf{Q}(i)) \leq (1 - \beta^{\tau(n-\psi)}) < 1$, we get

$$\begin{aligned}\lim_{t\to\infty} \delta(\mathit{\Pi}_{i=1}^{t}\mathbf{M}[i]) \;&=\; \lim_{t\to\infty} \delta\left(\left(\mathit{\Pi}_{i=(\lfloor \frac{t}{\tau(n-\psi)} \rfloor)\tau(n-\psi)+1}^{t}\mathbf{M}[i]\right)\left(\mathit{\Pi}_{i=1}^{\lfloor \frac{t}{\tau(n-\psi)} \rfloor}\mathbf{Q}(i)\right)\right) \\ &\leq\; \lim_{t\to\infty} \mathit{\Pi}_{i=1}^{\lfloor \frac{t}{\tau(n-\psi)} \rfloor}\lambda(\mathbf{Q}(i)) \;=\; 0\end{aligned}$$

Thus, the rows of $\mathit{\Pi}_{i=1}^{t}\mathbf{M}[i]$ become identical in the limit. This observation, and the fact that $v[t] = (\mathit{\Pi}_{i=1}^{t}\mathbf{M}[i])v[0]$ together imply that the states of the fault-free nodes satisfy the convergence condition. □

7 Summary and Discussion

This paper considers a *generalized* fault model [7,9,5], which can be used to specify more complex failure patterns, such as correlated failures or non-uniform node reliabilities. Under this fault model, we prove a necessary condition for the existence of synchronous iterative approximate Byzantine consensus algorithms

in arbitrary directed graphs. Then, we show the condition is also sufficient by providing a new IABC algorithm.

We present a transition matrix-based proof to show the correctness of the proposed algorithm. While transition matrices have been used to prove correctness of non-fault-tolerant consensus [6], this paper is the first to extend the technique to Byzantine consensus.

There are two main open problems: (i) for arbitrary graph, finding the global fault-tolerance parameter f, and (ii)analyzing the communication and computation complexity of Algorithm 1.

References

1. V. Bhandari and N. H. Vaidya. On reliable broadcast in a radio network. In *Proc. of ACM symposium on Principles of distributed computing*, PODC '05.
2. S. Dasgupta, C. Papadimitriou, and U. Vazirani. *Algorithms*. McGraw-Hill Higher Education, 2006.
3. D. Dolev, N. A. Lynch, S. S. Pinter, E. W. Stark, and W. E. Weihl. Reaching approximate agreement in the presence of faults. *J. ACM*, 33:499–516, May 1986.
4. J. Hajnal and M. S. Bartlett. Weak ergodicity in non-homogeneous markov chains. In *Proceedings of the Cambridge Philosophical Society*, 1958.
5. M. Hirt and U. Maurer. Complete characterization of adversaries tolerable in secure multi-party computation (extended abstract). *PODC* '97.
6. A. Jadbabaie, J. Lin, and A. Morse. Coordination of groups of mobile autonomous agents using nearest neighbor rules. *Automatic Control, IEEE Transactions on*, 48(6):988 – 1001, june 2003.
7. F. P. Junqueira and K. Marzullo. Synchronous consensus for dependent process failures. ICDCS '03.
8. C.-Y. Koo. Broadcast in radio networks tolerating byzantine adversarial behavior. In *Proc. ACM Symp. on Principles of Distributed Computing (PODC' 04)*, 2004.
9. P. Kuznetsov. Understanding non-uniform failure models. *Bulletin of the European Association for Theoretical Computer Science (BEATCS)*, 106:53–77, 2012.
10. L. Lamport, R. Shostak, and M. Pease. The byzantine generals problem. *ACM Trans. on Programming Languages and Systems*, 1982.
11. H. LeBlanc, H. Zhang, S. Sundaram, and X. Koutsoukos. Consensus of multi-agent networks in the presence of adversaries using only local information. *HiCoNs* '12.
12. N. A. Lynch. *Distributed Algorithms*. Morgan Kaufmann, 1996.
13. L. Tseng and N. H. Vaidya. Iterative approximate byzantine consensus under a generalized fault model. Technical report, CSL, UIUC, 2012.
14. N. H. Vaidya, L. Tseng, and G. Liang. Iterative approximate byzantine consensus in arbitrary directed graphs. *PODC* '12.
15. J. Wolfowitz. Products of indecomposable, aperiodic, stochastic matrices. In *Proc. of the American Mathematical Society*, volume 14, pages 733–737, 1963.