Iterative Approximate Byzantine Consensus in Arbitrary Directed Graphs *

Nitin H. Vaidya
University of Illinois
Electrical and Computer Engineering
Urbana, Illinois
nhv@illinois.edu

Lewis Tseng
University of Illinois
Computer Science Department
Urbana, Illinois
ltseng3@illinois.edu

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

ABSTRACT

This paper proves a necessary and sufficient condition for the existence of *iterative* algorithms that achieve *approximate Byzantine consensus* in arbitrary directed graphs, where each directed edge represents a communication channel between a pair of nodes. The class of iterative algorithms considered in this paper ensures that, after each iteration of the algorithm, the state of each fault-free node remains in the *convex hull* of the states of the fault-free nodes at the end of the previous iteration. The following *convergence* requirement is imposed: for any $\epsilon > 0$, after a sufficiently large number of iterations, the states of the fault-free nodes are guaranteed to be within ϵ of each other.

To the best of our knowledge, *tight* necessary and sufficient conditions for the existence of such iterative consensus algorithms in synchronous *arbitrary* point-to-point networks in presence of *Byzantine faults* have not been developed previously.

The methodology and results presented in this paper can also be extended to asynchronous systems.

Categories and Subject Descriptors

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

General Terms

Algorithms

Keywords

Consensus, Byzantine faults, iterative algorithms

*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.

PODC'12, July 16–18, 2012, Madeira, Portugal.

1. INTRODUCTION

Dolev et al. [5] introduced the notion of *approximate Byzantine consensus* by relaxing the requirement of *exact* consensus [14]. 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). In presence of Byzantine faults, while *exact* consensus is impossible in *asynchronous* systems [8], approximate consensus is achievable [5]. The notion of approximate consensus is of interest in *synchronous* systems as well, since approximate consensus can be achieved using distributed algorithms that do not require complete knowledge of the network topology [3]. The rest of the discussion in this paper – with the exception of Section 8 – applies to synchronous systems.

We consider "iterative" algorithms for achieving approximate Byzantine consensus in synchronous point-to-point networks that are modeled by arbitrary *directed* graphs. The *iterative approximate Byzantine consensus* (IABC) algorithms of interest have the following properties, which we will soon state more formally:

- *Initial state* of each node is equal to a real-valued *input* provided to that node.
- *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 $\epsilon > 0$, after a sufficiently large number of iterations, the states of the fault-free nodes are guaranteed to be within ϵ of each other.

In this paper, for the existence of a correct IABC algorithm, we derive a necessary condition that must be satisfied by the underlying communication graph. For graphs that satisfy this necessary condition, we show the correctness of a specific IABC algorithm, proving that the necessary conditions are tight. The rest of the paper is organized as follows. Section 2 present our system and network models. Related work is discussed in Section 3. Section 4 describes the iterative algorithms of interest. The necessary condition is derived in Section 5. A specific IABC algorithm is described in Section 6, and its correctness is proved in Section 7. Section 8 extends our results to an iterative algorithm in asynchronous environments. Some recent results that build on the results presented in this paper are summarized in Section 9. The paper concludes with Section 10.

2. SYSTEM MODEL

Communication model: The system is assumed to be *synchronous* (except in Section 8). 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, $N_i^+ = \{ j \mid (i, j) \in \mathcal{E} \}$. 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.

Failure Model: We consider the Byzantine failure model, with up to f nodes becoming faulty. A faulty node may *misbehave* arbitrarily. Possible misbehavior includes sending incorrect and mismatching (or inconsistent) messages to different neighbors. The faulty nodes may potentially 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, contents of messages the other nodes send to each other, the algorithm specification, and the network topology.

3. RELATED WORK

As noted earlier, Dolev et al. presented the early results on Byzantine fault-tolerant iterative consensus [5]. The initial algorithms [5, 14] were proved correct in *fully connected* networks. Fekete [6] studied the convergence rate of approximate consensus algorithms.

There have been attempts at achieving approximate fault-tolerant consensus iteratively in *partially* connected graphs. Kieckhafer and Azadmanesh examined the necessary conditions in order to achieve "local convergence" in synchronous [10] and asynchronous [2] systems. [1] presents a specific class of networks in which convergence condition can be satisfied using iterative algorithms.

A restricted fault model – called "malicious" fault model – in which the faulty nodes are restricted to sending identical messages to their neighbors has also been explored recently [19, 11, 12, 13]. In contrast, our Byzantine model allows a faulty node to send different messages to different neighbors. Under the (restricted) malicious fault model, Zhang and Sundaram [19] develop sufficient conditions for iterative consensus algorithm assuming a "local" fault model (in their "local" model, a bounded number of each node's neighbors may be faulty).

LeBlanc and Koutsoukos [11] address a continuous time version of the consensus problem with malicious faults in complete graphs. Under both malicious and Byzantine fault models, LeBlanc and Koutsoukos [12] have identified some sufficient conditions under which the continuous time version of iterative consensus can be achieved with up to f faults in the network; however, these sufficient conditions are *not* tight.

For the malicious fault model, LeBlanc et al. [13] have independently obtained *tight* necessary and sufficient conditions for tolerating up to f total number of faults in the network. Under the malicious model, since a faulty node must send identical messages to all the neighbors, the necessary and sufficient conditions are weaker than those developed here for the Byzantine fault model. For instance, under the malicious model, iterative consensus is possible in a complete graph consisting of $2f + 1$ nodes, whereas at least $3f + 1$ nodes are necessary for consensus under the Byzantine fault model.

Iterative approximate consensus algorithms that do not tolerate faulty behavior have been studied extensively (e.g., [9, 3]). The proof technique used for proving *sufficiency* in this paper is inspired by the prior work on non-fault-tolerant iterative algorithms [3].

4. IABC ALGORITHMS

In this section, we describe the structure of the *iterative approximate Byzantine consensus* (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 (to nodes in N_i^+).

2. *Receive step:* Receive values on all incoming edges (from nodes in N_i^-). Denote by $r_i[t]$ the vector of values received by node i from its neighbors. The size of vector $r_i[t]$ is $|N_i^-|$.

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 as input the vector $r_i[t]$ and state $v_i[t-1]$.

$$v_i[t] \quad = \quad Z_i \left(r_i[t] , v_i[t-1] \right) \tag{1}$$

We now define $U[t]$ and $\mu[t]$, assuming that $\mathcal{F}$ is the set of Byzantine faulty nodes, with the nodes in $\mathcal{V} - \mathcal{F}$ being fault-free.[1]

- $U[t] = \max_{i \in \mathcal{V}-\mathcal{F}} v_i[t]$. $U[t]$ is the largest state among the fault-free nodes at the end of the t-th iteration. Since the initial state of each node is equal to its input, $U[0]$ is equal to the maximum value of the initial input at the fault-free nodes.

- $\mu[t] = \min_{i \in \mathcal{V}-\mathcal{F}} v_i[t]$. $\mu[t]$ is the smallest state among the fault-free nodes at the end of the t-th iteration. $\mu[0]$ is equal to the minimum value of the initial input at the fault-free nodes.

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

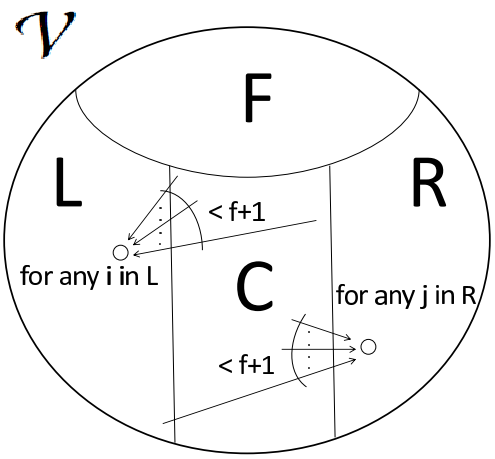

Figure 1: Illustration for the proof of Theorem 1. In this figure, $C \cup R \nRightarrow L$ and $L \cup C \nRightarrow R$.

The following conditions must be satisfied by an IABC algorithm in presence of up to f Byzantine faulty nodes:

- *Validity:* $\forall t > 0,\ \ \mu[t] \geq \mu[t-1]$ and $U[t] \leq U[t-1]$
- *Convergence:* $\lim_{t\to\infty} U[t] - \mu[t] = 0$

The objective in this paper is to identify the necessary and sufficient conditions for the existence of a *correct* IABC algorithm (i.e., an algorithm satisfying the above validity and convergence conditions) for a given $G(\mathcal{V}, \mathcal{E})$.

5. NECESSARY CONDITION

For a correct IABC algorithm to exist, the network graph $G(\mathcal{V}, \mathcal{E})$ must satisfy the necessary condition proved in this section. Theorems 1 and 2 below state equivalent necessary conditions. The form of the necessary condition in Theorem 2 is more intuitive, whereas the form in Theorem 1 is used later to prove sufficiency. We now define relations $\Rightarrow$ and $\nRightarrow$ that are used frequently in our discussion.

Definition 1. *For non-empty disjoint sets of nodes A and B,*

- *$A \Rightarrow B$ iff there exists a node $v \in B$ that has at least $f+1$ incoming edges from nodes in A, i.e., $|N_v^- \cap A| > f$.*
- *$A \nRightarrow B$ iff $A \Rightarrow B$ is* not *true.*

Theorem 1. *Suppose that a correct IABC algorithm exists for $G(\mathcal{V}, \mathcal{E})$. Let sets F, L, C, R form a partition[2] of $\mathcal{V}$, such that L and R are both non-empty, and $|F| \leq f$. Then, either $C \cup R \Rightarrow L$, or $L \cup C \Rightarrow R$.*

Proof. The proof is by contradiction. Let us assume that a correct iterative consensus algorithm exists, and $C \cup R \nRightarrow L$ and $L \cup C \nRightarrow R$. Thus, for any $i \in L$, $|N_i^- \cap (C \cup R)| < f+1$, and for any $j \in R$, $|N_j^- \cap (L \cup C)| < f+1$. Figure 1 illustrates the sets used in this proof.

[2]Sets $X_1, X_2, X_3, ..., X_p$ are said to form a partition of set X provided that (i) $\cup_{1 \leq i \leq p} X_i = X$, and (ii) $X_i \cap X_j = \Phi$ if $i \neq j$.

Also assume that the nodes in F (if F is non-empty) are all faulty, and the other nodes in sets L, C, R are fault-free. Note that the fault-free nodes are not aware of the identity of the faulty nodes.

Consider the case when (i) each node in L has initial input m, (ii) each node in R has initial input M, such that $M > m$, and (iii) each node in C, if C is non-empty, has an input in the interval $[m, M]$.

In the *Transmit Step* of iteration 1, suppose that the faulty nodes in F (if non-empty) send $m^- < m$ on outgoing links to nodes in L, send $M^+ > M$ on outgoing links to nodes in R, and send some arbitrary value in interval $[m, M]$ on outgoing links to the nodes in C (if C is non-empty). This behavior is possible since nodes in F are faulty. Note that $m^- < m < M < M^+$. Each fault-free node $k \in \mathcal{V} - F$, sends to nodes in N_k^+ value $v_k[0]$ in iteration 1.

Consider any node $i \in L$. Denote $N_i' = N_i^- \cap (C \cup R)$. Since $|F| \leq f$, $|N_i^- \cap F| \leq f$. Since $C \cup R \nRightarrow L$, $|N_i'| \leq f$. Node i will then receive m^- from the nodes in $N_i^- \cap F$, and values in $[m, M]$ from the nodes in N_i', and m from the nodes in $\{i\} \cup (N_i^- \cap L)$.

Consider the following two cases:

- Both $N_i^- \cap F$ and N_i' are non-empty: Now $|N_i^- \cap F| \leq f$ and $|N_i'| \leq f$. From node i's perspective, consider two possible scenarios: (a) nodes in $N_i^- \cap F$ are faulty, and the other nodes are fault-free, and (b) nodes in N_i' are faulty, and the other nodes are fault-free.

  In scenario (a), from node i's perspective, the fault-free nodes have sent values in interval $[m, M]$, whereas the faulty nodes have sent value m^-. According to the validity condition, $v_i[1] \geq m$. On the other hand, in scenario (b), the fault-free nodes have sent values m^- and m, where $m^- < m$; so $v_i[1] \leq m$, according to the validity condition. Since node i does not know whether the correct scenario is (a) or (b), it must update its state to satisfy the validity condition in both cases. Thus, it follows that $v_i[1] = m$.

- At most one of $N_i^- \cap F$ and N_i' is non-empty: Thus, $|(N_i^- \cap F) \cup N_i'| \leq f$. From node i's perspective, it is possible that the nodes in $(N_i^- \cap F) \cup N_i'$ are all faulty, and the rest of the nodes are fault-free. In this situation, the values sent to node i by the fault-free nodes (which are all in $\{i\} \cup (N_i^- \cap L)$) are all m, and therefore, $v_i[1]$ must be set to m as per the validity condition.

Thus, $v_i[1] = m$ for each node $i \in L$. Similarly, we can show that $v_j[1] = M$ for each node $j \in R$.

Now consider the nodes in set C, if C is non-empty. All the values received by the nodes in C are in $[m, M]$, therefore, their new state must also remain in $[m, M]$, as per the validity condition.

The above discussion implies that, at the end of iteration 1, the following conditions hold true: (i) state of each node in L is m, (ii) state of each node in R is M, and (iii) state of each node in C is in the interval $[m, M]$. These conditions are identical to the initial conditions listed previously. Then, by a repeated application of the above argument (proof by induction), it follows that for any $t \geq 0$, $v_i[t] = m$ for all $\forall i \in L$, $v_j[t] = M$ for all $j \in R$ and $v_k[t] \in [m, M]$ for all $k \in C$.

Since L and R both contain fault-free nodes, the convergence requirement is not satisfied. This is a contradiction to the assumption that a correct iterative algorithm exists. □

Corollary 1. *Suppose that a correct IABC algorithm exists for $G(\mathcal{V},\mathcal{E})$. Let $\{F,L,R\}$ be a partition of $\mathcal{V}$, such that L and R are both non-empty and $|F| \leq f$. Then, either $L \Rightarrow R$ or $R \Rightarrow L$.*

The proof follows by setting $C = \Phi$ in Theorem 1.

Corollary 2. *Suppose that a correct IABC algorithm exists for $G(\mathcal{V},\mathcal{E})$. Then n must be at least $3f+1$, and if $f > 0$, then each node must have at least $2f+1$ incoming edges.*

Proof. The necessary condition of $n \geq 3f+1$ has been shown previously [7]. We include a proof here for completeness. For $f = 0$, $n \geq 3f+1$ is trivially true. For $f > 0$, the proof is by contradiction. Suppose that $2 \leq n \leq 3f$. In this case, we can partition $\mathcal{V}$ into sets L, R, F such that $0 < |L| \leq f$, $0 < |R| \leq f$ and $0 \leq |F| \leq f$. Since $0 < |L| \leq f$ and $0 < |R| \leq f$, we have $L \nRightarrow R$ and $R \nRightarrow L$, respectively. This violates the necessary condition in Corollary 1. Thus, $n \geq 3f+1$.

The proof of the remaining corollary is also by contradiction. Suppose that $f > 0$, and for some node i, $|N_i^-| \leq 2f$. Define set $L = \{i\}$. Partition N_i^- into two sets F and H such that $|H| = \lfloor |N_i^-|/2 \rfloor \leq f$ and $|F| = \lceil |N_i^-|/2 \rceil \leq f$. Define $R = \mathcal{V} - F - L = \mathcal{V} - F - \{i\}$. Since $|\mathcal{V}| = n \geq 3f+1$, R is non-empty. Now, $N_i^- \cap R = H$, and $|N_i^- \cap R| \leq f$. Therefore, since $L = \{i\}$ and $|N_i^- \cap R| \leq f$, $R \nRightarrow L$. Also, since $|L| = 1 < f+1$, $L \nRightarrow R$. This violates Corollary 1 above. □

In Section 7, we prove that the condition stated in Theorem 1 is also sufficient for the existence of a correct IABC algorithm. The condition in Theorem 1 is not very intuitive. In Theorem 2 below, we state another necessary condition that is equivalent to the necessary condition in Theorem 1, and is somewhat easier to interpret. To facilitate the statement of Theorem 2, we now 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 non-empty 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.*

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.

It is known that the decomposition graph H^d is a directed *acyclic* graph [4].

Definition 3. **Source component***: Let H be a directed graph, and let H^d be its decomposition 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* <u>not</u> *reachable from any other vertex in H^d.*

Definition 4. **Reduced Graph:** *For a given graph $G(\mathcal{V},\mathcal{E})$ and $F \subset \mathcal{V}$, a graph $G_F(\mathcal{V}_F,\mathcal{E}_F)$ is said to be a* reduced graph*, if: (i) $\mathcal{V}_F = \mathcal{V} - F$, and (ii) $\mathcal{E}_F$ is obtained by first removing from $\mathcal{E}$ all the links incident on the nodes in F, and* then *removing up to f other incoming links at each node in $\mathcal{V}_F$.*

Note that for a given $G(\mathcal{V},\mathcal{E})$ and a given F, multiple reduced graphs G_F may exist.

Theorem 2. *Suppose that Theorem 1 holds for graph $G(\mathcal{V},\mathcal{E})$. Then, for any $F \subset \mathcal{V}$ such that $|F| < |\mathcal{V}|$ and $|F| \leq f$, every reduced graph G_F obtained as per Definition 4 must contain exactly one* source component.

Proof. Since $|F| < |\mathcal{V}|$, G_F contains at least one node; therefore, at least one source component must exist in G_F. We now prove that G_F cannot contain more than one source component. The proof is by contradiction. Suppose that there exists a set $F \subset \mathcal{V}$ with $|F| < |\mathcal{V}|$ and $|F| \leq f$, and a reduced graph $G_F(\mathcal{V}_F,\mathcal{E}_F)$ corresponding to F, such that the decomposition of G_F includes at least two source components.

Let the sets of nodes in two such source components of G_F be denoted L and R, respectively. Let $C = \mathcal{V} - F - L - R$. Observe that F, L, C, R form a partition of the nodes in $\mathcal{V}$. Since L is a source component in G_F it follows that there are no directed links in $\mathcal{E}_F$ from any node in $C \cup R$ to the nodes in L. Similarly, since R is a source component in G_F it follows that there are no directed links in $\mathcal{E}_F$ from any node in $L \cup C$ to the nodes in R. These observations, together with the manner in which $\mathcal{E}_F$ is defined, imply that (i) there are at most f links in $\mathcal{E}$ from the nodes in $C \cup R$ to each node in L, and (ii) there are at most f links in $\mathcal{E}$ from the nodes in $L \cup C$ to each node in R. Therefore, in graph $G(\mathcal{V},\mathcal{E})$, $C \cup R \nRightarrow L$ and $L \cup C \nRightarrow R$, violating Theorem 1. Thus, we have proved that G_F must contain exactly one source component. □

The above proof shows that Theorem 1 implies Theorem 2. Appendix A presents the proof that Theorem 2 implies Theorem 1. Thus, it follows that Theorems 1 and 2 specify equivalent conditions.[3]

Corollary 3. *Suppose that Theorem 1 holds true for graph $G(\mathcal{V},\mathcal{E})$. Then, for any $F \subset \mathcal{V}$ such that $|F| \leq f$, the unique source component in every reduced graph G_F must contain at least $f+1$ nodes.*

Proof. Since the source component is non-empty, the claim is trivially true for $f = 0$.

Now consider $f > 0$. The proof in this case is by contradiction. Suppose that there exists a set F with $|F| \leq f$, and a corresponding reduced graph $G_F(\mathcal{V}_F,\mathcal{E}_F)$, such that the decomposition of G_F contains a unique source component consisting of at most f nodes. Define L to be the set of nodes in this unique source component, and $R = \mathcal{V} - L - F$. Observe that F, L, R form a partition of $\mathcal{V}$. R must contain at least $f+1$ nodes, since $|L| \leq f$, $|F| \leq f$, and by Corollary 2, $n \geq 3f+1$.

Since $|L| \leq f$, it follows that in graph $G(\mathcal{V},\mathcal{E})$, $L \nRightarrow R$, Then Corollary 1 implies that, in graph $G(\mathcal{V},\mathcal{E})$, $R \Rightarrow L$. Thus, there must be a node in L, say node i, that has at least $f+1$ incoming links in $\mathcal{E}$ from the nodes in R. Since $i \in L$, it follows that $i \notin F$ (by definition of a reduced graph). Also, since i has at least $f+1$ incoming links in $\mathcal{E}$ from nodes in R, it follows that in $\mathcal{E}_F$, node i must have at least one incoming link from the nodes in R. This contradicts that assumption that set L containing node i is a source component of G_F. □

[3] An alternate interpretation of Theorem 2 is that in graph G_F non-fault-tolerant iterative consensus must be possible.

6. 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 4, and it is similar to algorithms that were analyzed in prior work as well [5, 14, 10] (although correctness of the algorithm under the necessary condition in Theorem 1 has not been proved previously).

Algorithm 1

Steps to be performed by node $i \in \mathcal{V}$ in t-th iteration, $t > 0$:

1. *Transmit step:* Transmit current state $v_i[t-1]$ on all outgoing edges.
2. Receive step: Receive values on all incoming edges. These values form vector $r_i[t]$ of size $|N_i^-|$. When a fault-free node expects to receive a message from a neighbor but does not receive the message, the message value is assumed to be equal to some *default value*.
3. *Update step:* Sort the values in $r_i[t]$ in an increasing order, and eliminate the smallest f values, and the largest f values (breaking ties arbitrarily). Let $N_i^*[t]$ denote the set of nodes from whom the remaining $N_i^- - 2f$ values were received, and let w_j denote the value received from node $j \in N_i^*[t]$. For convenience, define $w_i = v_i[t-1]$ to be the value node i "receives" from itself. Observe that if $j \in \{i\} \cup N_i^*[t]$ is fault-free, then $w_j = v_j[t-1]$.

   Define

$$v_i[t] \;=\; Z_i(r_i[t], v_i[t-1]) \;=\; \sum_{j \in \{i\} \cup N_i^*[t]} a_i \, w_j \qquad (2)$$

   where

$$a_i = \frac{1}{|N_i^-| + 1 - 2f}$$

   Note that $|N_i^*[t]| = |N_i^-| - 2f$, and $i \notin N_i^*[t]$ because $(i, i) \notin \mathcal{E}$. The "weight" of each term on the right-hand side of (2) is a_i, and these weights add to 1. Also, $0 < a_i \leq 1$. For future reference, let us define α as:

$$\alpha = \min_{i \in \mathcal{V}} a_i \qquad (3)$$

7. SUFFICIENCY (CORRECTNESS OF ALGORITHM 1)

In Theorems 3 and 4 in this section, we prove that Algorithm 1 satisfies *validity* and *convergence* conditions, respectively, provided that $G(\mathcal{V}, \mathcal{E})$ satisfies the condition below, which matches the necessary condition stated in Theorem 1.

Sufficient condition: For every partition F, L, C, R of $\mathcal{V}$, such that L and R are both non-empty, and $|F| \leq f$, either $C \cup R \Rightarrow L$, or $L \cup C \Rightarrow R$.

THEOREM 3. *Suppose that $\mathcal{F}$ is the set of Byzantine faulty nodes, and that $G(\mathcal{V}, \mathcal{E})$ satisfies the* sufficient *condition stated above. Then Algorithm 1 satisfies the* validity *condition.*

PROOF. Consider the t-th iteration, and any fault-free node $i \in \mathcal{V} - \mathcal{F}$. Consider two cases:

- $f = 0$: In this case, all nodes must be fault-free, and $\mathcal{F} = \Phi$. In (2) in Algorithm 1, note that $v_i[t]$ is computed using states from the previous iteration at node i and other nodes. By definition of $\mu[t-1]$ and $U[t-1]$, $v_j[t-1] \in [\mu[t-1], U[t-1]]$ for all fault-free nodes $j \in \mathcal{V} - \mathcal{F} = \mathcal{V}$. Thus, in this case, all the values used in computing $v_i[t]$ are in the interval $[\mu[t-1], U[t-1]]$. Since $v_i[t]$ is computed as a weighted average of these values, $v_i[t]$ is also within $[\mu[t-1], U[t-1]]$.
- $f > 0$: By Corollary 2, $|N_i^-| \geq 2f + 1$, and therefore, $|r_i[t]| \geq 2f + 1$. When computing set $N_i^*[t]$, the largest f and smallest f values from $r_i[t]$ are eliminated. Since at most f nodes are faulty, it follows that, either (i) the values received from the faulty nodes are all eliminated, or (ii) the values from the faulty nodes that still remain are between values received from two fault-free nodes. Thus, the remaining values in $r_i[t]$ are all in the interval $[\mu[t-1], U[t-1]]$. Also, $v_i[t-1]$ is in $[\mu[t-1], U[t-1]]$, as per the definition of $\mu[t-1]$ and $U[t-1]$. Thus $v_i[t]$ is computed as a weighted average of values in $[\mu[t-1], U[t-1]]$, and, therefore, it will also be in $[\mu[t-1], U[t-1]]$.

Since $\forall i \in \mathcal{V} - \mathcal{F}$, $v_i[t] \in [\mu[t-1], U[t-1]]$, the validity condition is satisfied. □

DEFINITION 5. *For disjoint sets A, B, $in(A \Rightarrow B)$ denotes the set of all the nodes in B that each have at least $f+1$ incoming edges from nodes in A. More formally,*

$$in(A \Rightarrow B) = \{\, v \mid v \in B \text{ and } \; f + 1 \leq |N_v^- \cap A| \,\}$$

With an abuse of notation, when $A \nRightarrow B$, define $in(A \Rightarrow B) = \Phi$.

DEFINITION 6. *For* non-empty disjoint *sets A and B, set A is said to* propagate to *set B in l steps, where $l > 0$, if there exist sequences of sets $A_0, A_1, A_2, \cdots, A_l$ and $B_0, B_1, B_2, \cdots, B_l$ (propagating sequences) such that*

- $A_0 = A, \quad B_0 = B, \quad A_l = A \cup B, \quad B_l = \Phi, \quad B_\tau \neq \Phi$ *for* $\tau < l$, *and*
- *for* $0 \leq \tau \leq l-1$,
  - $A_\tau \Rightarrow B_\tau$,
  - $A_{\tau+1} = A_\tau \cup in(A_\tau \Rightarrow B_\tau)$, *and*
  - $B_{\tau+1} = B_\tau - in(A_\tau \Rightarrow B_\tau)$

Observe that A_τ and B_τ form a partition of $A \cup B$, and for $\tau < l$, $in(A_\tau \Rightarrow B_\tau) \neq \Phi$. Also, when set A propagates to set B, the number of steps l in the above definition is upper bounded by $n - f - 1$, since set A must be of size at least $f + 1$ for it to propagate to B; otherwise, $A \nRightarrow B$.

LEMMA 1. *Assume that $G(\mathcal{V}, \mathcal{E})$ satisfies the* sufficient *condition stated above. For any partition A, B, F of $\mathcal{V}$, where A, B are both non-empty, and $|F| \leq f$, either A propagates to B, or B propagates to A.*

PROOF. Appendix B presents the proof. □

The lemma below states that the interval to which the states at all the fault-free nodes are confined shrinks after a finite number of iterations of Algorithm 1. Recall that $U[t]$ and $\mu[t]$ (defined in Section 4) are the maximum and minimum over the states at the fault-free nodes at the end of the t-th iteration.

LEMMA 2. *Suppose that $G(\mathcal{V},\mathcal{E})$ satisfies the* sufficient *condition stated above, and $\mathcal{F}$ is the set of Byzantine faulty nodes. Moreover, at the end of the s-th iteration of Algorithm 1, suppose that the fault-free nodes in $\mathcal{V}-\mathcal{F}$ can be partitioned into non-empty sets R and L such that (i) R propagates to L in l steps, and (ii) the states of nodes in R are confined to an interval of length $\leq \frac{U[s]-\mu[s]}{2}$. Then, with Algorithm 1,*

$$U[s+l] - \mu[s+l] \ \leq \ \left(1 - \frac{\alpha^l}{2}\right)(U[s] - \mu[s]) \tag{4}$$

where α is as defined in (3).

PROOF. Appendix C presents the proof. □

THEOREM 4. *Suppose that $\mathcal{F}$ is the set of Byzantine faulty nodes, and that $G(\mathcal{V},\mathcal{E})$ satisfies the* sufficient *condition stated above. Then Algorithm 1 satisfies the* convergence *condition.*

PROOF. Our goal is to prove that, given any $\epsilon > 0$, there exists τ such that

$$U[t] - \mu[t] \leq \epsilon \quad \forall t \geq \tau \tag{5}$$

Consider s-th iteration, for some $s \geq 0$. If $U[s] - \mu[s] = 0$, then the algorithm has already converged, and the proof is complete, with $\tau = s$ (recall that we have already proved that the algorithm satisfies the validity condition).

Now consider the case when $U[s] - \mu[s] > 0$. Partition $\mathcal{V} - \mathcal{F}$ into two subsets, A and B, such that, for each node $i \in A$, $v_i[s] \in \left[\mu[s], \frac{U[s]+\mu[s]}{2}\right)$, and for each node $j \in B$, $v_j[s] \in \left[\frac{U[s]+\mu[s]}{2}, U[s]\right]$. By definition of $\mu[s]$ and $U[s]$, there exist fault-free nodes i and j such that $v_i[s] = \mu[s]$ and $v_j[s] = U[s]$. Thus, sets A and B are both non-empty. By Lemma 1, one of the following two conditions must be true:

- Set A propagates to set B. Then, define $L = B$ and $R = A$. The states of all the nodes in $R = A$ are confined within an interval of length $< \frac{U[s]+\mu[s]}{2} - \mu[s] \leq \frac{U[s]-\mu[s]}{2}$.
- Set B propagates to set A. Then, define $L = A$ and $R = B$. In this case, states of all the nodes in $R = B$ are confined within an interval of length $\leq U[s] - \frac{U[s]+\mu[s]}{2} \leq \frac{U[s]-\mu[s]}{2}$.

In both cases above, we have found non-empty sets L and R such that (i) L, R is a partition of $\mathcal{V} - \mathcal{F}$, (ii) R propagates to L, and (iii) the states in R are confined to an interval of length $\leq \frac{U[s]-\mu[s]}{2}$. Suppose that R propagates to L in $l(s)$ steps, where $l(s) \geq 1$. Then by Lemma 2,

$$U[s+l(s)] - \mu[s+l(s)] \leq \left(1 - \frac{\alpha^{l(s)}}{2}\right)(U[s] - \mu[s]) \tag{6}$$

In Algorithm 1, observe that $a_i > 0$ for all i. Therefore, α defined in 3 in Algorithm 1 is > 0. Then, $n - f - 1 \geq l(s) \geq 1$ and $0 < \alpha \leq 1$; hence, $0 \leq \left(1 - \frac{\alpha^{l(s)}}{2}\right) < 1$.

Let us define the following sequence of iteration indices:

- $\tau_0 = 0$,
- for $i > 0$, $\tau_i = \tau_{i-1} + l(\tau_{i-1})$, where $l(s)$ for any given s was defined above.

If for some i, $U[\tau_i] - \mu[\tau_i] = 0$, then since the algorithm is already proved to satisfy the validity condition, we will have $U[t] - \mu[t] = 0$ for all $t \geq \tau_i$, and the proof of convergence is complete.

Now suppose that $U[\tau_i] - \mu[\tau_i] \neq 0$ for the values of i in the analysis below. By repeated application of the argument leading to (6), we can prove that, for $i \geq 0$,

$$U[\tau_i] - \mu[\tau_i] \leq \left(\Pi_{j=1}^{i}\left(1 - \frac{\alpha^{\tau_j - \tau_{j-1}}}{2}\right)\right)(U[0] - \mu[0]) \tag{7}$$

For a given ϵ, by choosing a large enough i, we can obtain

$$\left(\Pi_{j=1}^{i}\left(1 - \frac{\alpha^{\tau_j - \tau_{j-1}}}{2}\right)\right)(U[0] - \mu[0]) \leq \epsilon$$

and, therefore,

$$U[\tau_i] - \mu[\tau_i] \leq \epsilon \tag{8}$$

For $t \geq \tau_i$, by validity of Algorithm 1, it follows that

$$U[t] - \mu[t] \leq U[\tau_i] - \mu[\tau_i] \leq \epsilon$$

This concludes the proof. □

It should be easy to see that other correct IABC algorithms can be obtained by choosing "weights" differently than in Algorithm 1, and with other appropriate ways of eliminating values in the *Update step*. In recent work [18] we have developed an alternate proof of sufficiency, based on a transition matrix representation of the update step in Algorithm 1.

8. ASYNCHRONOUS NETWORKS

Dolev et al. [5] propose an iterative algorithm for asynchronous networks wherein message and processing delays may be arbitrary but finite. We extend their approach to arbitrary point-to-point networks. In particular, we consider the *Asynchronous IABC Algorithm* structure below, which is similar to the algorithm in [5]. This algorithm structure differs from the structure presented in Section 4 in two important ways: (i) the messages containing states are now tagged by the iteration index to which the states correspond, and (ii) each node i waits to receive only $|N_i^-| - f$ messages containing states from iteration $t-1$ before computing the new state in its t-th iteration. Due to the asynchronous nature of the system, different nodes may potentially perform their t-th iteration at very different real times.

Asynchronous IABC Algorithm

Steps to be performed by each node $i \in \mathcal{V}$ in its t-th iteration, $t > 0$:

1. *Transmit step:* Transmit current state $v_i[t-1]$ on all outgoing edges. The message is tagged by index $t-1$.
2. *Receive step:* Wait until $|N_i^-| - f$ messages tagged by index $t-1$ are received on the incoming edges. Values received in these messages form vector $r_i[t]$ of size $|N_i^-| - f$.
3. *Update step:* Node i updates its state using a transition function Z_i.

$$v_i[t] \quad = \quad Z_i\,(\,r_i[t]\,,\,v_i[t-1]\,) \tag{9}$$

We now introduce relation $\overset{a}{\Rightarrow}$ that is analogous to relation $\Rightarrow$ defined previously.

DEFINITION 7. *For non-empty disjoint sets of nodes A and B, $A \overset{a}{\Rightarrow} B$ iff there exists a node $v \in B$ that has at least $2f+1$ incoming edges from nodes in A, i.e., $|N_v^- \cap A| \geq 2f+1$.*

Theorem 5 states a necessary condition for asynchronous iterative algorithms with the above structure.

THEOREM 5. *If an* Asynchronous IABC Algorithm *satisfies* validity *and* convergence *conditions in graph* $G(\mathcal{V}, \mathcal{E})$*, then for any partition* F, L, C, R *of* $\mathcal{V}$*, such that* L *and* R *are both non-empty and* $|F| \leq f$*, then either* $C \cup R \stackrel{a}{\Rightarrow} L$*, or* $L \cup C \stackrel{a}{\Rightarrow} R$*.*

PROOF. The proof is similar to the proof of Theorem 1. More details can be found in [17]. □

The following corollary can be obtained from Theorem 5 [17].

COROLLARY 4. *If an* Asynchronous IABC Algorithm *satisfies* validity *and* convergence *conditions in graph* $G(\mathcal{V}, \mathcal{E})$*, then* $n > 5f$*, and when* $f > 0$*,* $|N_i^-| \geq 3f + 1$ *for all* $i \in \mathcal{V}$*.*

It can be shown that the necessary condition in Theorem 5 is tight. In particular, an *Asynchronous IABC Algorithm* with the structure above that performs the *Update step* shown below can be proved to satisfy the convergence and validity conditions [17]. Note that the *Update step* below, to be performed by each node $i \in \mathcal{V}$, is similar to that in *Algorithm 1* for the synchronous network.

- *Update step:* Sort the values in vector $r_i[t]$ in an increasing order, and eliminate the smallest f and the largest f values (breaking ties arbitrarily). Recall that $r_i[t]$ contains $|N_i^-| - f$ values. Let $N_i^*[t]$ denote the set of nodes from whom the remaining $|N_i^-| - 3f$ values were received, and let w_j denote the value received from node $j \in N_i^*[t]$. Define $w_i = v_i[t-1]$, and

$$v_i[t] \quad = \quad \sum_{j \in \{i\} \cup N_i^*[t]} a_i \, w_j \tag{10}$$

where

$$a_i = \frac{1}{|N_i^-| + 1 - 3f}.$$

9. OTHER RESULTS

The results presented in this paper have led to other related results described elsewhere. Here we summarize the other results. An alternate proof of correctness of Algorithm 1, using a transition matrix representation of the algorithm, is presented in [18]. Our necessary conditions are useful to examine whether IABC algorithms exist for specific graph families [16]. For instance, IABC is feasible in an undirected "core" network consisting of a clique of $2f + 1$ nodes, with the remaining nodes being connected to all the nodes in this clique [16]. Our results can also be extended to other system models, particularly, the *partially* asynchronous algorithmic model of [3], as shown in [17], and networks with time-varying topologies, as briefly discussed in [18]. Finally, the results can also be extended to a *generalized* Byzantine fault model [15] wherein possible faults are specified using a set of feasible fault sets. The generalized fault model can be used to capture correlated failures as well as different levels of reliabilities for different nodes in the system.

10. CONCLUSIONS

This paper proves a *tight* necessary and sufficient condition for the existence of a class of synchronous iterative approximate Byzantine consensus algorithms (IABC) that can tolerate up to f Byzantine fault in arbitrary directed graphs. These results can be extended to a class of iterative algorithms for asynchronous systems, as briefly discussed in Section 8. The work presented in this paper has led to further related results, as summarized in Section 9.

11. REFERENCES

[1] A. Azadmanesh and H. Bajwa. Global convergence in partially fully connected networks (PFCN) with limited relays. *Conf. of IEEE Industrial Electronics Soc. (IECON)*, 2001.

[2] M. H. Azadmanesh and R. Kieckhafer. Asynchronous approximate agreement in partially connected networks. *International Journal of Parallel and Distributed Systems and Networks*, 2002. `http://ahvaz.unomaha.edu/azad/pubs/ijpdsn.asyncpart.pdf`

[3] D. P. Bertsekas and J. N. Tsitsiklis. *Parallel and Distributed Computation: Numerical Methods*. Optimization and Neural Computation Series. Athena Scientific, 1997.

[4] S. Dasgupta, C. Papadimitriou, and U. Vazirani. *Algorithms*. McGraw-Hill Higher Education, 2006.

[5] 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.

[6] A. D. Fekete. Asymptotically optimal algorithms for approximate agreement. *ACM PODC*, 1986.

[7] M. J. Fischer, N. A. Lynch, and M. Merritt. Easy impossibility proofs for distributed consensus problems. *ACM PODC*, 1985.

[8] M. J. Fischer, N. A. Lynch, and M. S. Paterson. Impossibility of distributed consensus with one faulty process. *J. ACM*, 32:374–382, April 1985.

[9] 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.

[10] R. M. Kieckhafer and M. H. Azadmanesh. Low cost approximate agreement in partially connected networks. *J. of Computing and Information*, 1993. *http://ahvaz.ist.unomaha.edu/azad/pubs/jci.syncpart.pdf*

[11] H. LeBlanc and X. Koutsoukos. Consensus in networked multi-agent systems with adversaries. *14th International conference on Hybrid Systems: Computation and Control (HSCC)*, 2011.

[12] H. LeBlanc and X. Koutsoukos. Low complexity resilient consensus in networked multi-agent systems with adversaries. *Int. Conf. on Hybrid Systems: Computation and Control (HSCC)*, 2012.

[13] H. LeBlanc, H. Zhang, S. Sundaram, and X. Koutsoukos. Consensus of multi-agent networks in the presence of adversaries using only local information. *Conference on High Confidence Networked Systems* (HiCoNS), 2012.

[14] N. A. Lynch. *Distributed Algorithms*. Morgan Kaufmann, 1996.

[15] L. Tseng and N. H. Vaidya, "Iterative Approximate Byzantine Consensus under a Generalized Fault Model," report under preparation, May 2012.

[16] N. H. Vaidya, L. Tseng, and G. Liang. Iterative approximate Byzantine consensus in arbitrary directed

graphs. Tech. Rep., University of Illinois, January 2012. http://arxiv.org/abs/1201.4183

[17] N. H. Vaidya, L. Tseng, and G. Liang. Iterative approximate Byzantine consensus in arbitrary directed graphs – Part II: Synchronous and asynchronous systems. Tech. Rep., University of Illinois, February 2012. http://arxiv.org/abs/1202.6094

[18] N. H. Vaidya, "Matrix Representation of Iterative Approximate Byzantine Consensus in Directed Graphs," Tech. Rep., University of Illinois, March 2012. http://arxiv.org/abs/1203.1888

[19] H. Zhang and S. Sundaram. Robustness of information diffusion algorithms to locally bounded adversaries. http://arxiv.org/abs/1110.3843, October 2011. A version to appear at ACC 2012 as *Robustness of Distributed Algorithms to Locally Bounded Adversaries*.

APPENDIX

A. THEOREM 2 IMPLIES THEOREM 1

We now prove that Theorem 2 implies the correctness of Theorem 1. We achieve this by proving that, if the condition in Theorem 1 does not hold true for $G(\mathcal{V},\mathcal{E})$, then the condition in Theorem 2 also does not hold true.

Proof. Suppose that the condition stated in Theorem 1 does not hold for $G(\mathcal{V},\mathcal{E})$. Thus, there exists a partition F, L, C, R of $\mathcal{V}$ such that $|F| \leq f$, L and R are non-empty, and $C \cup R \nRightarrow L$ and $L \cup C \nRightarrow R$.

We now construct a reduced graph $G_F(\mathcal{V}_F, \mathcal{E}_F)$ corresponding to set F. First, remove all nodes in F from $\mathcal{V}$ to obtain $\mathcal{V}_F$. Remove all the edges incident on F from $\mathcal{E}$. Then because $C \cup R \nRightarrow L$, the number of incoming edges at each node in L from the nodes in $C \cup R$ is at most f; remove all these edges. Similarly, for every node $j \in R$, remove all incoming edges from $L \cup C$ (there are at most f such edges at each node $j \in R$). The resulting graph G_F is a reduced graph that satisfies the conditions in Definition 4.

In $\mathcal{E}_F$, there are no incoming edges to nodes in R from the nodes in $L \cup C$; similarly, in $\mathcal{E}_F$, there are no incoming edges to nodes in L from the nodes in $C \cup R$. It follows that no single node in $\mathcal{V}_F$ has paths in G_F (i.e., paths consisting of links in $\mathcal{E}_F$) to all the other nodes in $\mathcal{V}_F$. Thus, G_F must contain more than one source component. Thus, Theorem 2 does not hold for $G(\mathcal{V},\mathcal{E})$. □

B. PROOF OF LEMMA 1

To prove Lemma 1, we first prove the following Lemma.

Lemma 3. *Assume that $G(\mathcal{V},\mathcal{E})$ satisfies Theorem 1. Consider a partition A, B, F of $\mathcal{V}$ such that A and B are non-empty, and $|F| \leq f$. If $B \nRightarrow A$, then set A propagates to set B.*

Proof. Since A, B are non-empty, and $B \nRightarrow A$, by Corollary 1, we have $A \Rightarrow B$.

Define $A_0 = A$ and $B_0 = B$. Now, for a suitable $l > 0$, we will build propagating sequences $A_0, A_1, \cdots A_l$ and $B_0, B_1, \cdots B_l$ inductively.

- Recall that $A = A_0$ and $B = B_0 \neq \Phi$. Since $A \Rightarrow B$, $in(A_0 \Rightarrow B_0) \neq \Phi$. Define $A_1 = A_0 \cup in(A_0 \Rightarrow B_0)$ and $B_1 = B_0 - in(A_0 \Rightarrow B_0)$.

  If $B_1 = \Phi$, then $l = 1$, and we have found the propagating sequence already.

  If $B_1 \neq \Phi$, then define $L = A = A_0$, $R = B_1$ and $C = A_1 - A = B - B_1$. Since $B \nRightarrow A$, $R \cup C \nRightarrow L$. Therefore, by Theorem 1, $L \cup C \Rightarrow R$. That is, $A_1 \Rightarrow B_1$.
- For increasing values of $i \geq 0$, given A_i and B_i, where $B_i \neq \Phi$, by following steps similar to the previous item, we can obtain $A_{i+1} = A_0 \cup in(A_i \Rightarrow B_i)$ and $B_{i+1} = B_i - in(A_i \Rightarrow B_i)$, such that either $B_{i+1} = \Phi$ or $A_{i+1} \Rightarrow B_{i+1}$.

In the above construction, l is the smallest index such that $B_l = \Phi$. □

A more detailed proof of the above lemma is presented in [16].

Proof of Lemma 1.

Proof. Consider two cases:

- $A \nRightarrow B$: Then by Lemma 3 above, B propagates to A, completing the proof.
- $A \Rightarrow B$: In this case, consider two sub-cases:
  - *A propagates to B*: The proof in this case is complete.
  - *A does not propagate to B*: Recall that $A \Rightarrow B$. Since A does not propagate to B, propagating sequences defined in Definition 6 do not exist in this case. More precisely, there must exist $k > 0$, and sets $A_0, A_1, \cdots, A_k$ and $B_0, B_1, \cdots, B_k$, such that:
    - $A_0 = A$ and $B_0 = B$, and
    - for $0 \leq i \leq k-1$,
      - $A_i \Rightarrow B_i$,
      - $A_{i+1} = A_i \cup in(A_i \Rightarrow B_i)$, and
      - $B_{i+1} = B_i - in(A_i \Rightarrow B_i)$.
    - $B_k \neq \Phi$ and $A_k \nRightarrow B_k$.

    The last condition above violates the requirements for A to propagate to B.

    Now, $A_k \neq \Phi$, $B_k \neq \Phi$, and A_k, B_k, F form a partition of $\mathcal{V}$. Since $A_k \nRightarrow B_k$, by Lemma 3 above, B_k propagates to A_k.

    Given that $B_k \subseteq B_0 = B$, $A = A_0 \subseteq A_k$, and B_k propagates to A_k, now we prove that B propagates to A.

    Recall that A_i and B_i form a partition of $\mathcal{V} - F$.

    Let us define $P = P_0 = B_k$ and $Q = Q_0 = A_k$. Thus, P propagates to Q. Suppose that $P_0, P_1, ...P_m$ and $Q_0, Q_1, \cdots, Q_m$ are the propagating sequences in this case, with P_i and Q_i forming a partition of $P \cup Q = A_k \cup B_k = \mathcal{V} - F$.

    Let us define $R = R_0 = B$ and $S = S_0 = A$. Note that R, S form a partition of $A \cup B = \mathcal{V} - F$. Now, $P_0 = B_k \subseteq B = R_0$ and $S_0 = A \subseteq A_k = Q_0$. Also, $R_0 - P_0$ and S_0 form a partition of Q_0. Figure 2 illustrates some of the sets used in this proof.

    - Define $P_1 = P_0 \cup (in(P_0 \Rightarrow Q_0))$, and $Q_1 = \mathcal{V} - F - P_1 = Q_0 - (in(P_0 \Rightarrow Q_0))$. Also, $R_1 = R_0 \cup (in(R_0 \Rightarrow S_0))$, and $S_1 = \mathcal{V} - F - R_1 = S_0 - (in(R_0 \Rightarrow S_0))$. Since $R_0 - P_0$ and S_0 are a partition of Q_0, the nodes in $in(P_0 \Rightarrow Q_0)$ belong to one of these two sets. Note that $R_0 - P_0 \subseteq R_0$. Also, $S_0 \cap in(P_0 \Rightarrow Q_0) \subseteq in(R_0 \Rightarrow S_0)$. Therefore, it follows that $P_1 = P_0 \cup (in(P_0 \Rightarrow Q_0)) \subseteq R_0 \cup (in(R_0 \Rightarrow S_0)) = R_1$.

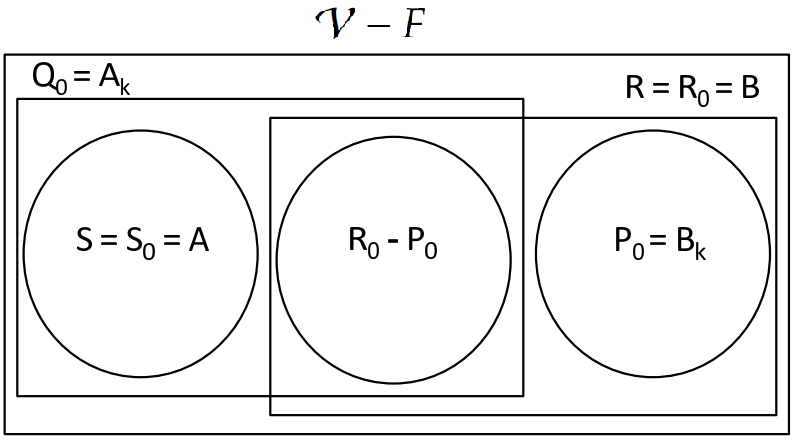

Figure 2: Illustration for the last part of the proof of Lemma 1. In this figure, $R_0 = P_0 \cup (R_0 - P_0)$ and $Q_0 = S_0 \cup (R_0 - P_0)$.

Thus, we have shown that, $P_1 \subseteq R_1$. Then it follows that $S_1 \subseteq Q_1$.

* For $0 \leq i < m$, let us define $R_{i+1} = R_i \cup in(R_i \Rightarrow S_i)$ and $S_{i+1} = S_i - in(R_i \Rightarrow S_i)$. Then following an argument similar to the above case, we can inductively show that, $P_i \subseteq R_i$ and $S_i \subseteq Q_i$. Due to the assumption on the length of the propagating sequence above, $P_m = P \cup Q = \mathcal{V} - F$ and $Q_m = \Phi$. Thus, there must exist $r \leq m$, such that for $i < r$, $R_i \neq \mathcal{V} - F$, and $R_r = \mathcal{V} - F$ and $S_r = \Phi$. The sequences $R_0, R_1, \cdots, R_r$ and $S_0, S_1, \cdots, S_r$ form propagating sequences, proving that $R = B$ propagates to $S = A$.

□

C. PROOF OF LEMMA 2

We first present two additional lemmas (using the notation in Algorithm 1).

Lemma 4. *Suppose that $\mathcal{F}$ is the set of faulty nodes, and that $G(\mathcal{V}, \mathcal{E})$ satisfies the "sufficient condition" stated in Section 7. Consider node $i \in \mathcal{V} - \mathcal{F}$. Let $\psi \leq \mu[t-1]$. Then, for $j \in \{i\} \cup N_i^*[t]$,*

$$v_i[t] - \psi \geq a_i (w_j - \psi)$$

Specifically, for fault-free $j \in \{i\} \cup N_i^*[t]$,

$$v_i[t] - \psi \geq a_i (v_j[t-1] - \psi)$$

Proof. In (2) in Algorithm 1, for each $j \in \{i\} \cup N_i^*[t]$, consider two cases:

- j is fault-free: Then, either $j = i$ or $j \in N_i^*[t] \cap (\mathcal{V} - \mathcal{F})$. In this case, $w_j = v_j[t-1]$. Therefore, $\mu[t-1] \leq w_j \leq U[t-1]$.
- j is faulty: In this case, f must be non-zero (otherwise, all nodes are fault-free). By Corollary 2, $|N_i^-| \geq 2f + 1$. Then it follows that, in step 2 of Algorithm 1, the smallest f values in $r_i[t]$ contain the state of at least one fault-free node, say k. This implies that $v_k[t-1] \leq w_j$. This, in turn, implies that $\mu[t-1] \leq w_j$.

Thus, for all $j \in \{i\} \cup N_i^*[t]$, we have $\mu[t-1] \leq w_j$. Therefore,

$$w_j - \psi \geq 0 \text{ for all } j \in \{i\} \cup N_i^*[t] \tag{11}$$

Since weights in (2) in Algorithm 1 add to 1, we can re-write that equation as,

$$\begin{aligned} v_i[t] - \psi &= \sum_{j \in \{i\} \cup N_i^*[t]} a_i (w_j - \psi) \\ &\geq a_i (w_j - \psi), \quad \forall j \in \{i\} \cup N_i^*[t] \quad \text{from (11)} \end{aligned} \tag{12}$$

For fault-free $j \in \{i\} \cup N_i^*[t]$, $w_j = v_j[t-1]$, therefore,

$$v_i[t] - \psi \geq a_i (v_j[t-1] - \psi) \tag{13}$$

□

Lemma 5. *Suppose that $\mathcal{F}$ is the set of faulty nodes, and that $G(\mathcal{V}, \mathcal{E})$ satisfies the "sufficient condition" stated in Section 7. Consider fault-free node $i \in \mathcal{V} - \mathcal{F}$. Let $\Psi \geq U[t-1]$. Then, for $j \in \{i\} \cup N_i^*[t]$,*

$$\Psi - v_i[t] \geq a_i (\Psi - w_j)$$

Specifically, for fault-free $j \in \{i\} \cup N_i^*[t]$,

$$\Psi - v_i[t] \geq a_i (\Psi - v_j[t-1])$$

Proof. The proof is similar to Lemma 4 proof. [16]. □

Proof of Lemma 2.

Proof. Since R propagates to L, as per Definition 6, there exist sequences of sets $R_0, R_1, \cdots, R_l$ and $L_0, L_1, \cdots, L_l$, where

- $R_0 = R, \quad L_0 = L, \quad R_l = R \cup L, \quad L_l = \Phi, \quad$ for $0 \leq \tau < l$, $L_\tau \neq \Phi$, and
- for $0 \leq \tau \leq l - 1$,

* $R_\tau \Rightarrow L_\tau$,
* $R_{\tau+1} = R_\tau \cup in(R_\tau \Rightarrow L_\tau)$, and
* $L_{\tau+1} = L_\tau - in(R_\tau \Rightarrow L_\tau)$

Let us define the following bounds on the states of the nodes in R at the end of the s-th iteration:

$$M = max_{j \in R}\, v_j[s] \quad (14)$$

$$m = min_{j \in R}\, v_j[s] \quad (15)$$

By the assumption in the statement of Lemma 2,

$$M - m \leq \frac{U[s] - \mu[s]}{2} \quad (16)$$

Also, $M \leq U[s]$ and $m \geq \mu[s]$. Therefore, $U[s] - M \geq 0$ and $m - \mu[s] \geq 0$.

The remaining proof of Lemma 2 relies on derivation of the three intermediate claims below.

Claim 1. *For $0 \leq \tau \leq l$, for each node $i \in R_\tau$,*

$$v_i[s+\tau] - \mu[s] \geq \alpha^\tau (m - \mu[s]) \quad (17)$$

Proof of Claim 1: The proof is by induction.
Induction basis: By definition of m, (17) holds true for $\tau = 0$.
Induction: Assume that (17) holds true for some τ, $0 \leq \tau < l$. Consider $R_{\tau+1}$. Observe that R_τ and $R_{\tau+1} - R_\tau$ form a partition of $R_{\tau+1}$; let us consider each of these sets separately.

- Set R_τ: By assumption, for each $i \in R_\tau$, (17) holds true. By validity of Algorithm 1 (proved in Theorem 3), $\mu[s] \leq \mu[s+\tau]$. Therefore, setting $\psi = \mu[s]$ and $t = s + \tau + 1$ in Lemma 4, we get,

$$\begin{aligned} v_i[s+\tau+1] - \mu[s] &\geq a_i\,(v_i[s+\tau] - \mu[s]) \\ &\geq a_i\,\alpha^\tau (m - \mu[s]) \quad \text{due to (17)} \\ &\geq \alpha^{\tau+1}(m - \mu[s]) \quad \text{due to (3)} \\ &\quad \text{and because} \quad m - \mu[s] \geq 0 \end{aligned}$$

- Set $R_{\tau+1} - R_\tau$: Consider a node $i \in R_{\tau+1} - R_\tau$. By definition of $R_{\tau+1}$, we have that $i \in in(R_\tau \Rightarrow L_\tau)$. Thus,

$$|N_i^- \cap R_\tau| \geq f + 1$$

  In Algorithm 1, $2f$ values (f smallest and f largest) received by node i are eliminated before $v_i[s+\tau+1]$ is computed at the end of $(s+\tau+1)$-th iteration. Consider two possibilities:

  - Value received from one of the nodes in $N_i^- \cap R_\tau$ is *not* eliminated. Suppose that this value is received from fault-free node $p \in N_i^- \cap R_\tau$. Then, by an argument similar to the previous case, we can set $\psi = \mu[s]$ in Lemma 4, to obtain,

$$\begin{aligned} v_i[s+\tau+1] - \mu[s] &\geq a_i\,(v_p[s+\tau] - \mu[s]) \\ &\geq a_i\,\alpha^\tau (m - \mu[s]) \quad \text{due to (17)} \\ &\geq \alpha^{\tau+1}(m - \mu[s]) \quad \text{due to (3)} \\ &\quad \text{and because} \quad m - \mu[s] \geq 0 \end{aligned}$$

  - Values received from *all* (there are at least $f+1$) nodes in $N_i^- \cap R_\tau$ are eliminated. Note that in this case f must be non-zero (for $f = 0$, no value is eliminated, as already considered in the previous case). By Corollary 2, we know that each node must have at least $2f+1$ incoming edges. Since at least $f+1$ values from nodes in $N_i^- \cap R_\tau$ are eliminated, and there are at least $2f+1$ values to choose from, it follows that the values that are *not* eliminated[4] are within the interval to which the values from $N_i^- \cap R_\tau$ belong. Thus, there exists a node k (possibly faulty) from whom node i receives some value w_k – which is not eliminated – and a fault-free node $p \in N_i^- \cap R_\tau$ such that

$$v_p[s+\tau] \leq w_k \quad (18)$$

  Then by setting $\psi = \mu[s]$ and $t = s+\tau+1$ in Lemma 4, we have

$$\begin{aligned} v_i[s+\tau+1] - \mu[s] &\geq a_i\,(w_k - \mu[s]) \\ &\geq a_i\,(v_p[s+\tau] - \mu[s]) \quad \text{by (18)} \\ &\geq a_i\,\alpha^\tau (m - \mu[s]) \quad \text{due to (17)} \\ &\geq \alpha^{\tau+1}(m - \mu[s]) \quad \text{due to (3)} \\ &\quad \text{and because} \quad m - \mu[s] \geq 0 \end{aligned}$$

Thus, we have shown that for all nodes in $R_{\tau+1}$,

$$v_i[s+\tau+1] - \mu[s] \geq \alpha^{\tau+1}(m - \mu[s])$$

This completes the proof of Claim 1.

Claim 2. *For each node $i \in \mathcal{V} - \mathcal{F}$,*

$$v_i[s+l] - \mu[s] \geq \alpha^l (m - \mu[s]) \quad (19)$$

Proof of Claim 2: Note that by definition, $R_l = \mathcal{V} - \mathcal{F}$. Then the proof follows by setting $\tau = l$ in the above Claim 1.

Claim 3. *For each node $i \in \mathcal{V} - \mathcal{F}$,*

$$U[s] - v_i[s+l] \geq \alpha^l (U[s] - M) \quad (20)$$

The proof of Claim 3 is similar to the proof of Claim 2 [16].

Now let us resume the proof of the Lemma 2. Note that $R_l = \mathcal{V} - \mathcal{F}$. Thus,

$$\begin{aligned} U[s+l] &= \max_{i \in \mathcal{V}-\mathcal{F}} v_i[s+l] \\ &\leq U[s] - \alpha^l (U[s] - M) \quad \text{by (20)} \end{aligned} \quad (21)$$

and

$$\begin{aligned} \mu[s+l] &= \min_{i \in \mathcal{V}-\mathcal{F}} v_i[s+l] \\ &\geq \mu[s] + \alpha^l (m - \mu[s]) \quad \text{by (19)} \end{aligned} \quad (22)$$

Subtracting (22) from (21),

$$\begin{aligned} & U[s+l] - \mu[s+l] \\ \leq\ & U[s] - \alpha^l (U[s] - M) - \mu[s] - \alpha^l (m - \mu[s]) \\ =\ & (1 - \alpha^l)(U[s] - \mu[s]) + \alpha^l (M - m) \\ \leq\ & (1 - \alpha^l)(U[s] - \mu[s]) + \alpha^l \frac{U[s] - \mu[s]}{2} \quad \text{by (16)} \\ \leq\ & (1 - \frac{\alpha^l}{2})(U[s] - \mu[s]) \end{aligned}$$

This concludes the proof of Lemma 2. □

[4] At least one value received from the nodes in N_i^- is not eliminated, since there are $2f+1$ incoming edges, and only $2f$ values are eliminated.