# Iterative Approximate Byzantine Consensus in Arbitrary Directed Graphs - Part II: Synchronous and Asynchronous Systems *

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

[1] Department of Electrical and Computer Engineering,
[2] Department of Computer Science, and
[3] Coordinated Science Laboratory
University of Illinois at Urbana-Champaign
Email: {nhv, ltseng3, gliang2}@illinois.edu

Technical Report

February 27, 2012

*This research is supported in part by National Science Foundation award CNS 1059540. Any opinions, findings, and conclusions or recommendations expressed here are those of the authors and do not necessarily reflect the views of the funding agencies or the U.S. government.

# 1 Introduction

This report contains two related sets of results with different assumptions on synchrony. The first part is about iterative algorithms in synchronous systems. Following our previous work on synchronous iterative approximate Byzantine consensus (IABC) algorithms [6], we provide a more intuitive *tight* necessary and sufficient condition for the existence of such algorithms in synchronous networks[1]. We believe this condition and the results in [6] also hold in partially asynchronous algorithmic model introduced in [2].

In the second part of the report, we explore the problem in asynchronous networks. While the traditional Byzantine consensus is not solvable in asynchronous systems [5], approximate Byzantine consensus can be solved using iterative algorithms [4].

# 2 Preliminaries

In this section, we present the network and failure models that are common to both parts.

## 2.1 Network Model

The 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 nodes in $\mathcal{V}$. With a slight abuse of terminology, we use the terms "edge" and "link" interchangeably. We assume that $n \geq \max(2, 3f + 1)$, since the consensus problem for $n = 1$ is trivial. If a directed edge $(i, j) \in \mathcal{E}$, then node $i$ can reliably transmit to node $j$. For convenience, we exclude self-loops from $\mathcal{E}$, although every node is allowed to send messages to itself. We also assume that all edges are authenticated, such that when a node $j$ receives a message from node $i$ (on edge $(i, j)$), it can correctly determine that the message was sent by node $i$. 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, $N_i^+ = \{\, j \mid (i, j) \in \mathcal{E} \,\}$. By definition, $i \notin N_i^-$ and $i \notin N_i^+$. However, we emphasize that each node can indeed send messages to itself.

## 2.2 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 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 state of the other nodes in the system and a complete knowledge of specification of the algorithm.

[1] With a slight abuse of terminology, we use "systems" and "networks" interchangeably in this report.

*Part I: Synchronous Networks*

## Synchronous Networks

The network is assumed to be synchronous. This report provides a more intuitive condition that is equivalent to our original necessary and sufficient condition introduced in Theorem 1 of [6]. Note that the discussion in this part is not self-contained, and relies heavily on the material and notations in [6].

# 3 More Intuitive Necessary and Sufficient Condition

For completeness, we state the tight condition from our previous report [6] here again:

**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 $F$ contains at most $f$ nodes. Then, at least one of these two conditions must be true: (i) $C \cup R \Rightarrow L$, or (ii) $L \cup C \Rightarrow R$.*[3]

This condition is not very intuitive. In Theorem 2 below, we state another tight necessary and sufficient 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 1 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 $H$.*

*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 $c_l$ 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 [3].

**Definition 2 Source component***: Let $H$ be a directed graph, and let $H^d$ be its decomposition as per Definition 1. 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 $H^d$.*

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

[3] Note that the notion of "$\Rightarrow$" and "$\overset{a}{\Rightarrow}$" (will be introduced in asynchronous networks part) is similar to "r-robust" graph presented in [7].

**Definition 3 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 3 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 $G(\mathcal{V}, \mathcal{E})$, $C \cup R \not\Rightarrow L$ and $L \cup C \not\Rightarrow R$, 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. Now, we prove that Theorem 2 implies Theorem 1.

**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 $C \cup R \not\Rightarrow L$ and $L \cup C \not\Rightarrow R$.

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 $C \cup R \not\Rightarrow L$, 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 3.

In $\mathcal{E}_F$, there are no incoming edges to nodes in $R$ from the nodes $L \cup C$; similarly, in $\mathcal{E}_F$, there are no incoming edges to nodes $L$ 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 $G(\mathcal{V}, \mathcal{E})$. □

By two results above, it follows that Theorems 1 and 2 specify equivalent conditions.[4]

Next, we present a weaker necessary conditions derived from Theorem 2 that implies the property of the source component.

**Corollary 1** *Suppose that Theorem 1 holds 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:** The proof 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. Also define $C = \Phi$ and $R = \mathcal{V} - L - F - C$. Observe that $F, L, C, R$ form a partition of $\mathcal{V}$.

Since $|L \cup C| = |L| \leq f$, it follows that in graph $G(\mathcal{V}, \mathcal{E})$, $L \cup C \not\Rightarrow R$, Then Theorem 1 implies that, in graph $G(\mathcal{V}, \mathcal{E})$, $C \cup R \Rightarrow L$. That is, since $C = \Phi$, $R \Rightarrow L$, and there must be a node in $L$, say node $i$, that has at least $f+1$ links in $\mathcal{E}$ from the nodes in $R$. Since $i \in L$, it follows that $i \notin F$ (by definition of $\Rightarrow$). Also, since $i$ has at least $f+1$ incoming edges in $\mathcal{E}$ from nodes in $R$, it follows that in $\mathcal{E}_F$, node $i$ must have at least one incoming edge from the nodes in $R$. This contradicts that assumption that set $L$ containing node $i$ is a source component of $G_F$. □

Note that this Corollary implies that for the correctness of IABC on the graph, the graph must have a component that acts as a source with at least $f+1$ nodes and thus outnumbers the faulty nodes.

For a "local" fault model under the constraint that fault nodes send *identical* messages to their outgoing neighbors, Zhang and Sundaram [7] showed *sufficiency* of a graph property similar to the condition above, although they do not prove that the sufficient condition is also necessary. Also, our fault model does not impose the above constraint on the faulty nodes.

# 4 Partially Asynchronous Algorithmic Model

[2] (Chapter 7) presents a *Partially Asynchronous Algorithmic Model*, in which an iterative algorithm analogous to Algorithm 1 [6] is used to solve iterative consensus with zero faults, with the following modifications:

- Each node may not necessarily update its state in each iteration. However, each node updates its state at least once in each set of consecutive $B$ iterations, where $B$ is a finite positive integer constant and is known to all nodes in advance.
- If node $i$ updates its state in iteration $t$, due to message delays, node $i$ may not necessarily be aware of the most recent state (i.e., at the end of the previous iteration) of its incoming neighbors. However, node $i$ will know the state of each incoming neighbor at the end of at

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

least one of the $B$ previous iterations[5]; the most recent state known is used in performing state update at node $i$.

We believe that the necessary and sufficient conditions for the IABC algorithm under partially asynchronous algorithmic model are identical to the necessary and sufficient conditions presented above and in [6] for the synchronous model. We expect that the proof is similar to the proof presented in [6].

[5]If node $i$ does not receive new values from some incoming neighbor $j$ in the past $B$ consecutive iterations, then by the model definition, node $i$ knows $j$ is faulty.

*Part II: Asynchronous Networks*

## Asynchronous Networks

In this part, we consider the iterative consensus problem in asynchronous networks. We will follow the definition of asynchronous system used in [4]. Each node operates at a completely arbitrary rate. Furthermore, the link between any pair of nodes suffers from an arbitrary but finite network delay[6] and out-of-order delivery.

Now, we introduce the class of algorithms that we will explore in this report.

# 5 Asynchronous Iterative Approximate Byzantine Consensus

**Algorithm Structure** By the definition of asynchronous systems, each node proceeds at different rate. Thus, Dolev et al. developed an algorithm based on "rounds" such that nodes update once in each round [4]. In particular, we consider the structure of *Async-IABC Algorithm* below, which has the same structure as the algorithm in [4]. This algorithm structure differs from the one for synchronous systems in [6] in two important ways: (i) the messages containing states are now tagged by the round index to which the states correspond, and (ii) each node $i$ waits to receive only $|N_i^-| - f$ messages containing states from round $t-1$ before computing the new state in round $t$.

Due to the asynchronous nature of the system, different nodes may potentially perform their $t$-th round at very different real times. Thus, the main difference between iteration and round is as following:

- Iteration is defined as fixed amount of real-time units. Hence, every node will be in the same iteration at any given real time.
- Round is defined as the time that each node updates its value[7]. Hence, every node may be in totally different rounds at any given real time in asynchronous systems.

In Async-IABC algorithm, each node $i$ maintains state $v_i$, with $v_i[t]$ denoting the state of node $i$ at the end of its $t$-th round. 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 round ($t > 0$), the state of node $i$ is $v_i[t-1]$. Now, we describe the steps that should be performed by each node $i \in \mathcal{V}$ in its $t$-th round.

---

**Async-IABC Algorithm**

---

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 the first $|N_i^-| - f$ messages tagged by index $t-1$ are received on the incoming edges (breaking ties arbitrarily). 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 $Z_i$.

   $Z_i$ is a part of the specification of the algorithm, and takes as input the vector $r_i[t]$ and state $v_i[t-1]$.

[6] The delay can also be variable.

[7] With a slight abuse of terminology, we will use "value" and "state" interchangeably in this report.

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

---

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 non-faulty.[8]

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

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

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

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

### 5.1 Notations

There are many notations used and will be introduced later in this part of the report. Here is a quick reference:

- $N_i^+, N_i^-$: set of outgoing neighbors and incoming neighbors of some node $i$, respectively.
- $U[t], \mu[t]$: maximum value and minimum value of all the fault-free nodes at the end of round $t$, respectively.
- $Z_i$: a function specifying how node $i$ updates its new value (algorithm specification).
- $N_i^@[t]$: set of incoming neighbors from whom node $i$ actually received values at round $t \geq 1$.
- $r_i[t]$: set of values sent by $N_i^@[t]$.
- $N_i^*[t]$: set of incoming neighbors from whom node $i$ actually used the values to update at round $t \geq 1$.

Note that by definition we have the following relationships: $N_i^*[t] \subset N_i^@[t] \subset N_i^-$. Moreover, $N_i^*[t]$ and $N_i^@[t]$ may change over the rounds, and $N_i^-$ is a constant. Lastly, $|N_i^@[t]| = |N_i^-| - 2f$ and $|N_i^*[t]| = |N_i^@[t]| - f$ for any round $t \geq 1$.

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

# 6 Necessary Condition

In asynchronous systems, for an Async-IABC algorithm satisfying the the *validity* and *convergence* conditions to exist, the underlying graph $G(\mathcal{V}, \mathcal{E})$ must satisfy a necessary condition proved in this section. We now define relations $\stackrel{a}{\Rightarrow}$ and $\stackrel{a}{\not\Rightarrow}$ that are used frequently in our proofs. Note that these definitions are analogous to the definitions of $\Rightarrow$ and $\not\Rightarrow$ in [6].

**Definition 4** *For non-empty disjoint sets of nodes $A$ and $B$,*

- $A \stackrel{a}{\Rightarrow} B$ *iff there exists a node $v \in B$ that has at least $2f+1$ incoming links from nodes in $A$, i.e., $|N_v^- \cap A| > 2f$.*
- $A \stackrel{a}{\not\Rightarrow} B$ *iff $A \stackrel{a}{\Rightarrow} B$ is* not *true.*

---

Now, we present the necessary condition for correctness of Async-IABC in asynchronous systems. Note that it is similar to that for synchronous systems [6], but with $\Rightarrow$ replaced by $\stackrel{a}{\Rightarrow}$.

**Theorem 3** *Let sets $F, L, C, R$ form a partition of $\mathcal{V}$, such that*

- $0 \leq |F| \leq f$,
- $0 < |L|$, *and*
- $0 < |R|$

*Then, at least one of the two conditions below must be true.*

- $C \cup R \stackrel{a}{\Rightarrow} L$
- $L \cup C \stackrel{a}{\Rightarrow} R$

**Proof:** The proof is by contradiction. Let us assume that a correct Async-IABC consensus algorithm exists, and $C \cup R \stackrel{a}{\not\Rightarrow} L$ and $L \cup C \stackrel{a}{\not\Rightarrow} R$. Thus, for any $i \in L$, $|N_i^- \cap (C \cup R)| < 2f+1$, and for any $j \in R$, $|N_j^- \cap (L \cup C)| < 2f+1$,

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

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

At the start of round 1, suppose that the faulty nodes in $F$ (if non-empty) send $m^- < m$ to outgoing neighbors in $L$, send $M^+ > M$ to outgoing neighbors in $R$, and send some arbitrary value in $[m, M]$ to outgoing neighbors 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 $k \in \mathcal{V} - \mathcal{F}$, sends to nodes in $N_k^+$ value $v_k[0]$ in round 1.

Consider any node $i \in L$. Denote $N_i' = N_i^- \cap (C \cup R)$. Since $C \cup R \stackrel{a}{\not\Rightarrow} L$, $|N_i'| \leq 2f$. Consider the situation where the delay between certain $w = \min(f, |N_i'|)$ nodes in $N_i'$ and node $i$ is arbitrarily large compared to all the other traffic (including messages from incoming neighbors

in $F$). Consequently, $r_i[1]$ includes $|N_i'| - w \leq f$ values from $N_i'$, since $w$ messages from $N_i'$ are delayed and thus ignored by node $i$. Recall that $N_i^@[1]$ is the set of nodes whose round 1 values are received by node $i$ in time (i.e., before $i$ finishes step 2 in Async-IABC). By the argument above, $N_i^@[1] \cap N_i' \leq f$.

Node $i$ receives $m^-$ from the nodes in $F \cap N_i^@[1]$, values in $[m, M]$ from the nodes in $N_i' \cap N_i^@[1]$, and $m$ from the nodes in $\{i\} \cup (L \cap N_i^@[1])$.

Consider four cases:

- $F \cap N_i^@[1]$ and $N_i' \cap N_i^@[1]$ are both empty: In this case, all the values that $i$ receives are from nodes in $\{i\} \cup (L \cap N_i^@[1])$, and are identical to $m$. By validity condition, node $i$ must set its new state, $v_i[1]$, to be $m$ as well.

- $F \cap N_i^@[1]$ is empty and $N_i' \cap N_i^@[1]$ is non-empty: In this case, since $|N_i' \cap N_i^@[1]| \leq f$, from $i$'s perspective, it is possible that all the nodes in $N_i^@[1] \cap N_i'$ are 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 $\{i\} \cup (L \cap N_i^@[1])$) are all $m$, and therefore, $v_i[1]$ must be set to $m$ as per the validity condition.

- $F \cap N_i^@[1]$ is non-empty and $N_i' \cap N_i^@[1]$ is empty: In this case, since $|F \cap N_i^@[1]| \leq f$, it is possible that all the nodes in $F \cap N_i^@[1]$ are 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 $\{i\} \cup (L \cap N_i^@[1])$) are all $m$, and therefore, $v_i[1]$ must be set to $m$ as per the validity condition.

- Both $F \cap N_i^@[1]$ and $N_i' \cap N_i^@[1]$ are non-empty: From node $i$'s perspective, consider two possible scenarios: (a) nodes in $F \cap N_i^@[1]$ are faulty, and the other nodes are fault-free, and (b) nodes in $N_i' \cap N_i^@[1]$ are faulty, and the other nodes are fault-free.

  In scenario (a), from node $i$'s perspective, the non-faulty nodes have values in $[m, M]$ whereas the faulty nodes have value $m^-$. According to the validity condition, $v_i[1] \geq m$. On the other hand, in scenario (b), the non-faulty nodes have 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 $v_i[1] = m$.

Observe that in each case above $v_i[1] = m$ 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 the first iteration, 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 $[m, M]$. These conditions are identical to the initial conditions listed previously. Then, by induction, it follows that for any $t \geq 0$, $v_i[t] = m, \forall i \in L$, and $v_j[t] = M, \forall j \in R$. Since $L$ and $R$ contain fault-free nodes, the convergence requirement is not satisfied. This is a contradiction to the assumption that a correct Async-IABC algorithm exists. □

**Corollary 2** *Let $\{F, L, R\}$ be a partition of $\mathcal{V}$, such that $0 \leq |F| \leq f$, and $L$ and $R$ are non-empty. Then, either $L \overset{a}{\Rightarrow} R$ or $R \overset{a}{\Rightarrow} L$.*

**Proof:** The proof follows by setting $C = \Phi$ in Theorem 3. □

**Corollary 3** *The number of nodes $n$ must exceed $5f$ for the existence of a correct Async-IABC algorithm that tolerates $f$ failures.*

**Proof:** The proof is by contradiction. Suppose that $2 \leq n \leq 5f$, and consider the following two cases:

- $2 \leq n \leq 4f$: Suppose that $L, R, F$ is a partition of $\mathcal{V}$ such that $|L| = \lceil n/2 \rceil \leq 2f$, $|R| = \lfloor n/2 \rfloor \leq 2f$ and $F = \Phi$. Note that $L$ and $R$ are non-empty, and $|L| + |R| = n$.
- $4f < n \leq 5f$:

  Suppose that $L, R, F$ is a partition of $\mathcal{V}$, such that $|L| = |R| = 2f$ and $|F| = n - 4f$. Note that $0 < |F| \leq f$.

In both cases above, Corollary 2 is applicable. Thus, either $L \overset{a}{\Rightarrow} R$ or $R \overset{a}{\Rightarrow} L$. For $L \overset{a}{\Rightarrow} R$ to be true, $L$ must contain at least $2f + 1$ nodes. Similarly, for $R \overset{a}{\Rightarrow} L$ to be true, $R$ must contain at least $2f + 1$ nodes. Therefore, at least one of the sets $L$ and $R$ must contain more than $2f$ nodes. This contradicts our choice of $L$ and $R$ above (in both cases, size of $L$ and $R$ is $\leq 2f$). Therefore, $n$ must be larger than $5f$. □

---

**Corollary 4** *For the existence of a correct Async-IABC algorithm, then for each node $i \in \mathcal{V}$, $|N_i^-| \geq 3f + 1$, i.e., each node $i$ has at least $3f + 1$ incoming links, when $f > 0$.*

**Proof:** The proof is by contradiction. Consider the following two cases for some node $i$:

- $|N_i^-| \leq 2f$: Define set $F = \Phi, L = \{i\}$ and $R = V - F - L = V - \{i\}$. Thus, $N_i^- \cap R = N_i^-$, and $|N_i^- \cap R| \leq 2f$ by assumption.
- $2f < |N_i^-| \leq 3f$: Define set $L = \{i\}$. Partition $N_i^-$ into two sets $F$ and $H$ such that $|F| = f$ and $|H| = |N_i^-| - f \leq 2f$. Define $R = V - F - L = V - F - \{i\}$. Thus, $N_i^- \cap R = H$, and $|N_i^- \cap R| \leq 2f$ by construction.

In both cases above, $L$ and $R$ are non-empty, so Corollary 2 is applicable. However, in each case, $L = \{i\}$ and $|L| = 1 < 2f + 1$; hence, $L \overset{a}{\not\Rightarrow} R$. Also, since $L = \{i\}$ and $|N_i^- \cap R| \leq 2f$, and hence $R \overset{a}{\not\Rightarrow} L$ by the definition of $\overset{a}{\Rightarrow}$. This leads to a contradiction. Hence, every node must have at least $3f + 1$ incoming neighbors.

□

# 7 Useful Lemmas

In this section, we introduce two lemmas that are used in our proof of convergence. Note that the proofs are similar to corresponding lemmas in [6] except for the adoption of $\overset{a}{\Rightarrow}$ and "rounds" instead of $\Rightarrow$ and "iterations."

**Definition 5** *For disjoint sets $A, B$, $in(A \overset{a}{\Rightarrow} B)$ denotes the set of all the nodes in $B$ that each have at least $2f+1$ incoming links from nodes in $A$. More formally,*

$$in(A \overset{a}{\Rightarrow} B) = \{\ v\ |v \in B \text{ and } \ 2f+1 \leq |N_v^- \cap A|\ \}$$

*With a slight abuse of notation, when $A \overset{a}{\not\Rightarrow} B$, define $in(A \overset{a}{\Rightarrow} B) = \Phi$.*

---

**Definition 6** *For* non-empty disjoint *sets $A$ and $B$, set $A$ is said to* **propagate to** *set $B$ in $l$ rounds, 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*

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

Observe that $A_\tau$ and $B_\tau$ form a partition of $A \cup B$, and for $\tau < l$, $in(A_\tau \overset{a}{\Rightarrow} B_\tau) \neq \Phi$. Also, when set $A$ propagates to set $B$, length $l$ above is necessarily finite. In particular, $l$ is upper bounded by $n - 2f - 1$, since set $A$ must be of size at least $2f+1$ for it to propagate to $B$.

---

**Lemma 1** *Assume that $G(\mathcal{V}, \mathcal{E})$ satisfies Theorem 3. Consider a partition $A, B, F$ of $\mathcal{V}$ such that $A$ and $B$ are non-empty, and $|F| \leq f$. If $B \overset{a}{\not\Rightarrow} A$, then set $A$ propagates to set $B$.*

**Proof:** Since $A, B$ are non-empty, and $B \overset{a}{\not\Rightarrow} A$, by Corollary 2, we have $A \overset{a}{\Rightarrow} B$.

The proof is by induction. Define $A_0 = A$ and $B_0 = B$. Thus $A_0 \overset{a}{\Rightarrow} B_0$ and $B_0 \overset{a}{\not\Rightarrow} A_0$. Note that $A_0$ and $B_0$ are non-empty.

*Induction basis*: For some $\tau \geq 0$,

- for $0 \leq k < \tau$, $A_k \overset{a}{\Rightarrow} B_k$, and $B_k \neq \Phi$,
- either $B_\tau = \Phi$ or $A_\tau \overset{a}{\Rightarrow} B_\tau$,
- for $0 \leq k < \tau$, $A_{k+1} = A_k \cup in(A_k \overset{a}{\Rightarrow} B_k)$, and $B_{k+1} = B_k - in(A_k \overset{a}{\Rightarrow} B_k)$

Since $A_0 \overset{a}{\Rightarrow} B_0$, the induction basis holds true for $\tau = 0$.

*Induction:* If $B_\tau = \Phi$, then the proof is complete, since all the conditions specified in Definition 6 are satisfied by the sequences of sets $A_0, A_1, \cdots, A_\tau$ and $B_0, B_1, \cdots, B_\tau$.

Now consider the case when $B_\tau \neq \Phi$. By assumption, $A_k \overset{a}{\Rightarrow} B_k$, for $0 \leq k \leq \tau$. Define $A_{\tau+1} = A_\tau \cup in(A_\tau \overset{a}{\Rightarrow} B_\tau)$ and $B_{\tau+1} = B_\tau - in(A_\tau \overset{a}{\Rightarrow} B_\tau)$. Our goal is to prove that either $B_{\tau+1} = \Phi$ or $A_{\tau+1} \overset{a}{\Rightarrow} B_{\tau+1}$. If $B_{\tau+1} = \Phi$, then the induction is complete. Therefore, now let us assume that $B_{\tau+1} \neq \Phi$ and prove that $A_{\tau+1} \overset{a}{\Rightarrow} B_{\tau+1}$. We will prove this by contradiction.

Suppose that $A_{\tau+1} \overset{a}{\not\Rightarrow} B_{\tau+1}$. Define subsets $L, C, R$ as follows: $L = A_0$, $C = A_{\tau+1} - A_0$ and $R = B_{\tau+1}$. Due to the manner in which $A_k$'s and $B_k$'s are defined, we also have $C = B_0 - B_{\tau+1}$. Observe that $L, C, R, F$ form a partition of $\mathcal{V}$, where $L, R$ are non-empty, and the following relationships hold:

- $C \cup R = B_0$, and
- $L \cup C = A_{\tau+1}$

Rewriting $B_0 \stackrel{a}{\not\Rightarrow} A_0$ and $A_{\tau+1} \stackrel{a}{\not\Rightarrow} B_{\tau+1}$, using the above relationships, we have, respectively,

$$C \cup R \stackrel{a}{\not\Rightarrow} L,$$

and

$$L \cup C \stackrel{a}{\not\Rightarrow} R$$

This violates the necessary condition in Theorem 3. This is a contradiction, completing the induction.

Thus, we have proved that, either (i) $B_{\tau+1} = \Phi$, or (ii) $A_{\tau+1} \stackrel{a}{\Rightarrow} B_{\tau+1}$. Eventually, for large enough $t$, $B_t$ will become $\Phi$, resulting in the propagating sequences $A_0, A_1, \cdots, A_t$ and $B_0, B_1, \cdots, B_t$, satisfying the conditions in Definition 6. Therefore, $A$ propagates to $B$. □

---

**Lemma 2** *Assume that $G(\mathcal{V}, \mathcal{E})$ satisfies Theorem 3. For any partition $A, B, F$ of $\mathcal{V}$, where $A, B$ are both non-empty, and $|F| \leq f$, at least one of the following conditions must be true:*

- *$A$ propagates to $B$, or*
- *$B$ propagates to $A$*

**Proof:** Consider two cases:

- $A \stackrel{a}{\not\Rightarrow} B$: Then by Lemma 1, $B$ propagates to $A$, completing the proof.
- $A \stackrel{a}{\Rightarrow} B$: In this case, consider two sub-cases:
  - $A$ propagates to $B$: The proof in this case is complete.
  - $A$ does not propagate to $B$: Thus, 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 $0 \leq i \leq k-1$,
      - $A_i \stackrel{a}{\Rightarrow} B_i$,
      - $A_{i+1} = A_i \cup in(A_i \stackrel{a}{\Rightarrow} B_i)$, and
      - $B_{i+1} = B_i - in(A_i \stackrel{a}{\Rightarrow} B_i)$.
    - $B_k \neq \Phi$ and $A_k \stackrel{a}{\not\Rightarrow} B_k$.

    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 $A_k \stackrel{a}{\not\Rightarrow} B_k$, by Lemma 1, $B_k$ propagates to $A_k$.

    Since $B_k \subseteq B_0 = B$, $A \subseteq A_k$, and $B_k$ propagates to $A_k$, it should be easy to see that $B$ propagates to $A$.

□

# 8 Sufficient Condition

## 8.1 Algorithm 2

We will prove that there exists an Async-IABC algorithm – particularly *Algorithm 2* below – that satisfies the *validity* and *convergence* conditions provided that the graph $G(\mathcal{V}, \mathcal{E})$ satisfies the necessary condition in Theorem 3. This implies that the necessary condition in Theorem 3 is also sufficient.

*Algorithm 2* has the three-step structure, and it is similar to algorithms that were analyzed in prior work as well [4, 1] (although correctness of the algorithm under the necessary condition in Theorem 3 has not been proved previously).

---

**Algorithm 2**

---

1. *Transmit step:* Transmit current state $v_i[t-1]$ on all outgoing edges.

2. Receive step: Wait until receiving values on all but $f$ incoming edges. These values form vector $r_i[t]$ of size $|N_i^-| - f$.[9]

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 identifiers of nodes from whom the remaining $N_i^- - 3f$ values were received, and let $w_j$ denote the value received from node $j \in N_i^*$. 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

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

   where

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

   Note that $|N_i^*[t]| = |N_i^-| - 3f$, 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 $\alpha$ as:

$$\alpha = \min_{i \in \mathcal{V}} \; a_i \tag{3}$$

---

## 8.2 Sufficiency

In Theorems 4 and 5 in this section, we prove that Algorithm 2 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 3.

**Sufficient condition:** *For every partition $F, L, C, R$ of $\mathcal{V}$, such that $L$ and $R$ are both non-empty, and $F$ contains at most $f$ nodes, at least one of these two conditions is true: (i) $C \cup R \overset{a}{\Rightarrow} L$, or (ii) $L \cup C \overset{a}{\Rightarrow} R$.*

[9] If more than $|N_i^-| - f$ values arrive at the same time, break ties arbitrarily.

Note that the proofs below are similar to the ones for synchronous systems in [6]. The main differences are the following:

- We need to consider only values in $N_i^{@}[t]$ not in $N_i^-$. This is due to different step 2 between Algorithm 1 [6] and Algorithm 2.
- We interpret $t$ as round index, rather than iteration index.

**Theorem 4** *Suppose that $G(\mathcal{V}, \mathcal{E})$ satisfies Theorem 3. Then Algorithm 2 satisfies the* validity *condition.*

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

- $f = 0$: In (2), note that $v_i[t]$ is computed using states from the previous round 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 $j \in \mathcal{V} - \mathcal{F}$. Thus, in this case, all the values used in computing $v_i[t]$ are in the range $[\mu[t-1], U[t-1]]$. 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 4, $|N_i^-| \geq 3f + 1$. Thus, $|N_i^{@}| \geq 2f + 1$, and $|r_i[t]| \geq 2f + 1$. 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 range $[\mu[t-1], U[t-1]]$. 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. □

---

Before proving the convergence of Algorithm 2, we first present three lemmas. In the discussion below, we assume that $G(\mathcal{V}, \mathcal{E})$ satisfies the sufficient condition.

**Lemma 3** *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), for each $j \in N_i^*[t]$, consider two cases:

- Either $j = i$ or $j \in N_i^*[t] \cap (\mathcal{V} - \mathcal{F})$: Thus, $j$ is fault-free. 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). From Corollary 4, $|N_i^-| \geq 3f+1$. Thus, $|N_i^@| \geq 2f+1$, and $|r_i[t]| \geq 2f+1$. Then it follows that the smallest $f$ values in $r_i[t]$ that are eliminated in step 2 of Algorithm 2 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,

$$w_j - \psi \geq 0 \text{ for all } j \in \{i\} \cup N_i^*[t] \tag{4}$$

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

$$\begin{aligned} v_i[t] - \psi &= \sum_{j \in \{i\} \cup N_i^*[t]} a_i \left(w_j - \psi\right) \\ &\geq a_i \left(w_j - \psi\right), \quad \forall j \in \{i\} \cup N_i^*[t] \qquad \text{from (4)} \end{aligned} \tag{5}$$

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

$$v_i[t] - \psi \geq a_i \left(v_j[t-1] - \psi\right) \tag{6}$$

□

---

Similar to the above result, we can also show the following lemma:

**Lemma 4** *Consider 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 $j \in \{i\} \cup N_i^*[t]$,*

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

Then we present the main lemma used in proof of convergence. Note that below, we use parameter $\alpha$ defined in (3). Recall that in (2) in Algorithm 2, $a_i > 0$ for all $i$, and thus, $\alpha > 0$.

**Lemma 5** *At the end of the $s$-th round, 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 $l$ rounds, and (ii) the states of nodes in $R$ are confined to an interval of length $\leq \frac{U[s]-\mu[s]}{2}$. Then,*

$$U[s+l] - \mu[s+l] \leq \left(1 - \frac{\alpha^l}{2}\right) \left(U[s] - \mu[s]\right) \tag{7}$$

**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

- $R_0 = R$, $L_0 = L$, $L_l = \Phi$, for $0 \leq \tau < l$, $L_\tau \neq \Phi$, and
- for $0 \leq \tau \leq l-1$,
  - $R_\tau \stackrel{a}{\Rightarrow} L_\tau$,
  - $R_{\tau+1} = R_\tau \cup in(R_\tau \stackrel{a}{\Rightarrow} L_\tau)$, and
  - $L_{\tau+1} = L_\tau - in(R_\tau \stackrel{a}{\Rightarrow} L_\tau)$

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

$$M = max_{j \in R}\ v_j[s] \tag{8}$$
$$m = min_{j \in R}\ v_j[s] \tag{9}$$

By the assumption in the statement of Lemma 5,

$$M - m \leq \frac{U[s] - \mu[s]}{2} \tag{10}$$

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 5 relies on derivation of the three intermediate claims below.

---

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

$$v_i[s+\tau] - \mu[s] \geq \alpha^\tau (m - \mu[s]) \tag{11}$$

*Proof of Claim 1:* The proof is by induction.

*Induction basis:* For some $\tau$, $0 \leq \tau < l$, for each node $i \in R_\tau$, (11) holds. By definition of $m$, the induction basis holds true for $\tau = 0$.

*Induction:* Assume that the induction basis holds true for some $\tau$, $0 \leq \tau < l$. Consider $R_{\tau+1}$. Observe that $R_\tau$ and $R_{\tau+1} - R_\tau$ form a partition of $R_{\tau+1}$; let us consider each of these sets separately.

- Set $R_\tau$: By assumption, for each $i \in R_\tau$, (11) holds true. By validity of Algorithm 2, $\mu[s] \leq \mu[s+\tau]$. Therefore, setting $\psi = \mu[s]$ in Lemma 3, we get,

$$\begin{aligned} v_i[s+\tau+1] - \mu[s] &\geq a_i\ (v_i[s+\tau] - \mu[s]) \\ &\geq a_i\ \alpha^\tau (m - \mu[s]) \qquad \text{due to (11)} \\ &\geq \alpha^{\tau+1}(m - \mu[s]) \qquad \text{due to (3)} \end{aligned}$$

- 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 $i \in in(R_\tau \stackrel{a}{\Rightarrow} L_\tau)$. Thus,

$$|N_i^- \cap R_\tau| \geq 2f + 1$$

It follows that

$$|N_i^{@}[s+\tau] \cap R_\tau| \geq f+1$$

In Algorithm 2, $2f$ 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 round. Consider two possibilities:

- Value received from one of the nodes in $N_i^{@}[s+\tau] \cap R_\tau$ is **not** eliminated. Suppose that this value is received from fault-free node $p \in N_i^{@}[s+\tau] \cap R_\tau$. Then, by an argument similar to the previous case, we can set $\psi = \mu[s]$ in Lemma 3, to obtain,

$$\begin{aligned} v_i[s+\tau+1] - \mu[s] &\geq a_i\ (v_p[s+\tau] - \mu[s]) \\ &\geq a_i\ \alpha^\tau (m - \mu[s]) \qquad \text{due to (11)} \\ &\geq \alpha^{\tau+1}(m - \mu[s]) \qquad \text{due to (3)} \end{aligned}$$

- Values received from **all** (there are at least $f+1$) nodes in $N_i^{@}[s+\tau] \cap R_\tau$ 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 4, we know that each node must have at least $3f+1$ incoming edges. Thus, $N_i^{@}[t+\tau] \geq 2f+1$. Since at least $f+1$ values from nodes in $N_i^{@}[t+\tau] \cap R_\tau$ are eliminated, and there are at least $2f+1$ values to choose from, it follows that the values that are **not** eliminated are within the interval to which the values from $N_i^{@}[s+\tau] \cap R_\tau$ 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 $p \in N_i^{@}[t+\tau] \cap R_\tau$ such that

$$v_p[s+\tau] \quad \leq \quad w_k \tag{12}$$

  Then by setting $\psi = \mu[s]$ in Lemma 3 we have

$$\begin{aligned} v_i[s+\tau+1] - \mu[s] &\geq a_i\ (w_k - \mu[s]) \\ &\geq a_i\ (v_p[s+\tau] - \mu[s]) \qquad \text{due to (12)} \\ &\geq a_i\ \alpha^\tau (m - \mu[s]) \qquad \text{due to (11)} \\ &\geq \alpha^{\tau+1}(m - \mu[s]) \qquad \text{due to (3)} \end{aligned}$$

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 $i \in \mathcal{V} - \mathcal{F}$,*

$$v_i[s+l] - \mu[s] \geq \alpha^l (m - \mu[s]) \tag{13}$$

*Proof of Claim 1:*

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

By a procedure similar to the derivation of Claim 2 above, we can also prove the claim below.

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

$$U[s] - v_i[s+l] \geq \alpha^l (U[s] - M) \tag{14}$$

Now let us resume the proof of the Lemma 5. Note that $R_l = \mathcal{V} - \mathcal{F}$. Thus,

$$\begin{aligned} U[s+l] &= \max_{i \in \mathcal{V}-\mathcal{F}} v_i[s+l] \\ &\leq U[s] - \alpha^l (U[s] - M) \qquad \text{by (14)} \end{aligned} \tag{15}$$

and

$$\begin{aligned} \mu[s+l] &= \min_{i \in \mathcal{V}-\mathcal{F}} v_i[s+l] \\ &\geq \mu[s] + \alpha^l (m - \mu[s]) \qquad \text{by (13)} \end{aligned} \tag{16}$$

Subtracting (16) from (15),

$$\begin{aligned} U[s+l] - \mu[s+l] &\leq U[s] - \alpha^l (U[s] - M) - \mu[s] - \alpha^l (m - \mu[s]) \\ &= (1 - \alpha^l)(U[s] - \mu[s]) + \alpha^l (M - m) && (17) \\ &\leq (1 - \alpha^l)(U[s] - \mu[s]) + \alpha^l \frac{U[s] - \mu[s]}{2} \qquad \text{by (10)} && (18) \\ &\leq (1 - \frac{\alpha^l}{2})(U[s] - \mu[s]) && (19) \end{aligned}$$

This concludes the proof of Lemma 5.

□

Now, we are able to prove the convergence of Algorithm 2. Note that this proof is essentially identical to the synchronous case [6].

**Theorem 5** *Suppose that $G(\mathcal{V}, \mathcal{E})$ satisfies Theorem 3. Then Algorithm 2 satisfies the* convergence *condition.*

**Proof:** Our goal is to prove that, given any $\epsilon > 0$, there exists $\tau$ such that

$$U[t] - \mu[t] \leq \epsilon \quad \forall t \geq \tau \tag{20}$$

Consider the $s$-th round, for some $s \geq 0$. If $U[s] - \mu[s] = 0$, then the algorithm has already converged, and the proof is complete, with $\tau = s$.

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 2, 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 $l(s) \geq 1$. By Lemma 5,

$$U[s + l(s)] - \mu[s + l(s)] \leq \left(1 - \frac{\alpha^{l(s)}}{2}\right)(U[s] - \mu[s]) \tag{21}$$

Since $n - f - 1 \geq l(s) \geq 1$ and $0 < \alpha \leq 1$, $0 \leq \left(1 - \frac{\alpha^{l(s)}}{2}\right) < 1$.

Let us define the following sequence of iteration indices[10]:

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

By repeated application of the argument leading to (21), we can prove that, for $i \geq 0$,

$$U[\tau_i] - \mu[\tau_i] \leq \left(\Pi_{j=1}^{i}\left(1 - \frac{\alpha^{\tau_j - \tau_{j-1}}}{2}\right)\right)\ (U[0] - \mu[0]) \tag{22}$$

For a given $\epsilon$, 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,

$$U[\tau_i] - \mu[\tau_i] \leq \epsilon \tag{23}$$

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. □

# 9 Conclusion

In this report, we present two sets of results. First, we prove another necessary and sufficient condition for the existence of synchronous IABC in arbitrary directed graphs. The condition is more intuitive than the one in [6]. We also believe that the results can be extended to partially asynchronous algorithmic model presented in [2]. In the second part, we extend our earlier results to asynchronous systems.

[10] Without loss of generality, we assume that $U[\tau_i] - \mu[\tau_i] > 0$. Otherwise, the statement is trivially true due to the validity shown in Theorem 4.

## References


[1] M. H. Azadmanesh and R. Kieckhafer. Asynchronous approximate agreement in partially connected networks. *International Journal of Parallel and Distributed Systems and Networks*, 5(1):26–34, 2002.

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

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

[4] 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.

[5] 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.

[6] N. H. Vaidya, L. Tseng, and G. Liang. Iterative approximate byzantine consensus in arbitrary directed graphs. *CoRR*, abs/1201.4183, 2012.

[7] H. Zhang and S. Sundaram. Robustness of information diffusion algorithms to locally bounded adversaries. *CoRR*, abs/1110.3843, 2011.