Reaching Approximate Byzantine Consensus with Multi-hop Communication

Lili Su(✉) and Nitin Vaidya

Department of Electrical and Computer Engineering,
University of Illinois at Urbana-Champaign, Urbana, USA
{lilisu3,nhv}@illinois.edu

Abstract. We address the problem of reaching approximate consensus in the presence of Byzantine faults in a synchronous system. We analyze *iterative* algorithms that maintain *minimal* state, and impose the constraint that in each iteration the nodes may only communicate with other nodes that are up to l hops away. For a given l, we prove a necessary and sufficient condition on the network structure for the existence of correct iterative algorithms that achieve *approximate* Byzantine consensus. We prove sufficiency of the condition by designing a correct algorithm, which uses a trim function based on a *minimal messages cover* property introduced in this paper. Our necessary and sufficient condition generalizes the tight condition identified in prior work for $l = 1$. For $l \geq l^*$, where l^* is the length of a longest cycle-free path in the given network, our condition is equivalent to the necessary and sufficient conditions for exact consensus in undirected and directed networks both.

Keywords: Approximate byzantine consensus · Iterative algorithm · Synchronous system · Incomplete network · Bounded length communication paths

1 Introduction

The Byzantine fault-tolerance problem was first introduced in [9] by Pease, Shostak and Lamport in 1980, and is one of the most fundamental problems in distributed computing. Fisher, Lynch and Paterson [7] showed that the fault-tolerant consensus problem cannot be solved in *asynchronous* system even in the presence of only one crash failure. As one way to circumvent this impossibility result, the notion of *approximate Byzantine consensus* was introduced by Dolev et al. in [4] by requiring that the agents agree with each other only approximately. The notion of approximate consensus is of interest in *synchronous* system as well [4,8,14]. The discussion in this paper applies to synchronous system.

This research is supported in part by National Science Foundation awards NSF 1329681 and 1421918. 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.

A. Pelc and A.A. Schwarzmann (Eds.): SSS 2015, LNCS 9212, pp. 21–35, 2015.
DOI: 10.1007/978-3-319-21741-3_2

Let n be the total number of nodes and f be the upper bound on the number of faulty nodes in the system. In networks with bidirectional links, approximate consensus is achievable if and only if the network node-connectivity is at least $2f+1$ and less than one third of nodes can be faulty, i.e., $n \geq 3f+1$ [6]. Relaxing the bidirectional communication assumption, a tight condition for directed graphs was presented in [11]. There has been increasing interest in designing iterative variants of approximate Byzantine consensus where only local knowledge of the network topology (and local communication) is needed, and agents carry minimal state across iterations [2,5,8,13,14]. Fekete [5] studied the convergence rate of approximate consensus algorithms over complete networks. [8,14] considered arbitrary directed networks and derived tight (necessary and sufficient) topological conditions on the communication network. While [14] investigated the Byzantine fault model, [8] considered a restricted fault model in which the faulty nodes are restricted to sending identical messages to their neighbors.

To the best of our knowledge, no attempts have been made on investigating the impact of each node's communication range on the network condition for a correct iterative approximate consensus algorithm to exist. In this paper, we model the network as a directed graph, and assume that in each iteration, any node may only communicate with nodes that are up to l hops away, by forwarding messages through intermediate nodes. The directed graph model is motivated by the presence of directed links in wireless networks. Our goal is to prove a necessary and sufficient condition on the network structure for the existence of correct iterative algorithms that achieve approximate Byzantine consensus for a given l with minimal memory (i.e., minimal amount of state carried across iterations).

Contributions: Our main contribution is to prove a necessary and sufficient condition on the network structure for a given l. Our sufficiency proof is shown by constructing a simple iterative algorithm, whose trim function is defined based on a *minimal messages cover* property that we introduce in this paper. The tight condition we found is consistent with the tight condition identified in [14] when only local communication is allowed, i.e., $l = 1$. For $l \geq l^*$, where l^* is the length of a longest cycle-free path in the given network, our condition is equivalent to the necessary and sufficient condition for consensus in undirected networks [6] as well as exact consensus in directed networks [12].

Organization: The rest of the paper is organized as follows. Section 2 presents our models and the structure of the iterative algorithms considered in our work. Our necessary condition is presented in Section 3, and its sufficiency is proved constructively in Section 4. The correspondence between our condition and the results in [4,6,12] is discussed in Section 5. Section 6 discusses possible relaxations of our fault model and concludes the paper.

2 System Model and Structure of Iterative Algorithms

Communication model: The system is assumed to be *synchronous*. The communication network is modeled as a simple *directed* graph G. Define $\mathcal{V}(G) = \{1, \ldots, n\}$ as the set of n nodes, where $n \geq 2$, and $\mathcal{E}(G)$ as the set of directed edges between nodes in $\mathcal{V}(G)$. Node i can send messages to node j if and only if there exists an i, j–path of length at most l in G, where l is a positive integer. In addition, we assume each node can send messages to itself as well, i.e., $(i, i) \in \mathcal{E}(G)$ for all $i \in \mathcal{V}(G)$. For each node i, let N_i^{l-} be the set of nodes that can reach node i via at most l hops. Similarly, denote the set of nodes that are reachable from node i via at most l hops by N_i^{l+}. Due to the existence of self-loops, $i \in N_i^{l-}$ and $i \in N_i^{l+}$. When $l = 1$, we write N_i^{1-} and N_i^{1+} as N_i^- and N_i^+, respectively, for simplicity. Each node i is assumed to be aware of the network topology within its l-hop neighborhood (i.e., node i knows all the paths of length at most l from the nodes in N_i^{l-}, and all the paths of length at most l to the nodes in N_i^{l+}). Node i may send a message to node j via different i, j–paths. To capture this distinction in transmission routes, we represent a message as a tuple $m = (w, P)$, where $w \in \mathbb{R}$ and P indicates the path via which message m should be transmitted. It is assumed that the network layer in the system delivers the messages along the specified paths. The intermediate nodes on the paths do not view the message values (i.e., the message values are not used by intermediate nodes in performing consensus). Four functions are defined over message m. For $m = (w, P)$, let function value be $\mathsf{value}(m) = w$ and let path be $\mathsf{path}(m) = P$, whose images are the first entry and the second entry, respectively, of message m. In addition, functions source and $\mathsf{destination}$ are defined by $\mathsf{source}(m) = i$ and $\mathsf{destination}(m) = j$ if P is an i, j–path, i.e., i and j are source and destination on path P. For a given path P, Let $\mathcal{V}(P)$ denote the set of nodes along the path, including the source and the destination.

Fault model: Let $\mathcal{F} \subseteq \mathcal{V}(G)$ be the collection of faulty nodes in the system. We consider the Byzantine fault model with up to f nodes becoming faulty, i.e., $|\mathcal{F}| \leq f$. A faulty node may tamper the message value arbitrarily. Possible misbehavior includes sending incorrect and mismatching (or inconsistent) messages to different neighbors. In addition, a faulty node $k \in \mathcal{F}$ may tamper message m if it is in the transmission path, i.e., $k \in \mathcal{V}(\mathsf{path}(m))$. However, faulty nodes may only tamper $\mathsf{value}(m)$, leaving $\mathsf{path}(m)$ unchanged. This constraint is placed for ease of exposition; later in Section 6 we relax this constraint. Faulty nodes are also assumed to have complete knowledge of the execution of the algorithm, including the states of all nodes, contents of messages that the other nodes send to each other, and the algorithm specification, so that they may potentially collaborate with each other adaptively.

Iterative approximate Byzantine consensus (IABC) algorithms: The iterative algorithms considered in this paper have the following structure: 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 messages of the form $(v_i[t-1], P)$ on each l–hop path P (including self-loops) to nodes in N_i^{l+}. As noted previously, the network layer of the system forwards each message to its destination along the path specified for the message.
2. *Receive step:* Receive messages from N_i^{l-} for which destination is i. When node i expects to receive a message from a path but does not receive the message, the message value is assumed to be equal to some default value. Let $\mathcal{M}_i[t]$ be the set of messages that node i received in this step.
3. *Update step:* Node i updates its state using a transition function Z_i, where Z_i is a part of the specification of the algorithm, and takes as input the set $\mathcal{M}_i[t]$.

$$v_i[t] = Z_i(\mathcal{M}_i[t]). \tag{1}$$

Algorithms with similar structure are considered in prior work as well [8,11,14]. The evolution of state $v_i[t]$ is governed by the update function defined in (1). Note that $v_i[t]$ only depends on $\mathcal{M}_i[t]$–the messages collected by node i at iteration t (which includes $v_i[t-1]$). No information collected/obtained during previous iterations will affect the update step in iteration t. Intuitively speaking, fault-free node i is assumed to have no memory across iterations other than its most recent state $v_i[t-1]$.

Let $U[t]$ be the largest state among the fault-free nodes at the end of the t-th iteration, i.e., $U[t] = \max_{i \in \mathcal{V}-\mathcal{F}} v_i[t]$. Since the initial state of each node is equal to its input, $U[0]$ is equal to the maximum value of the initial input of the fault-free nodes. Similarly, we define $\mu[t]$ to be the smallest state at the end of the t–th iteration and $\mu[0]$ to be the smallest initial input. For an IABC algorithm to be correct, the following two conditions must be satisfied:

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

Our goal is to identify a necessary and sufficient condition on graph G for the existence of a *correct* IABC algorithm (i.e., an algorithm satisfying the above validity and convergence conditions) for a given l.

3 Necessary Condition

For a correct IABC algorithm to exist, the underlying network G must satisfy the condition presented in this section. First, we introduce some definitions.

Definition 1. *Suppose $W \subseteq \mathcal{V}(G)$ and $x \in \mathcal{V}(G)$ such that $x \notin W$. A W, x–path is a path from some vertex $w \in W$ to vertex x. A set $S_l \subseteq \mathcal{V}(G)$ with $x \notin S_l$ is an*

*l–*restricted vertex cut *if the deletion of S_l disconnects all W, x–paths of length at most l. The l–*restricted W, x–connectivity*, denoted by $\kappa_l(W, x)$, is defined by*

$$\kappa_l(W, x) = \min_{S_l : S_l \text{ is an } l\text{–restricted } W, x\text{–cut}} |S_l|.$$

*A set of vertices S is a W, x–*vertex cut *if the removal of set S disconnects all W, x–paths. The W, x–*connectivity*, denoted by $\kappa(W, x)$, is defined by*

$$\kappa(W, x) = \min_{S : S \text{ is a } W, x\text{–cut}} |S|.$$

The second part of the above definition is the classic definition of node connectivity in graph theory [15], which is a global notion. In our communication model, we assume that each fault-free node only knows the local network topology up to its l–th hop neighborhood. Thus, we adapt node connectivity to our model by restricting the path length of interest. Note that $\kappa_l(W, x) = \kappa(W, x)$ for all $l \geq l^*$, and that $\kappa_1(W, x) = |W \cap N_x^-|$.

In general, $\kappa_l(W, x) \neq \kappa(W, x)$ and $\kappa_l(W, x) \leq \kappa_{l+1}(W, x)$ for all l. For instance for the system depicted in Figure 1, via enumeration it can be seen that

$$\kappa\left(\{p_2, p_3\}, p_1\right) = 2 \geq 1 = \kappa_1\left(\{p_2, p_3\}, p_1\right).$$

Intuitively speaking, the stronger the communication capability of each node is (the larger l is), the harder it is to prevent one node from being influenced by other nodes.

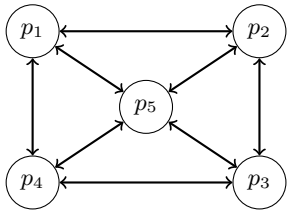

Fig. 1. In this system, there are five nodes p_1, p_2, p_3, p_4 and p_5; all communication links are bi-directional; and at most one node can be adversarial, i.e., $f = 1$

Definition 2. *For non-empty disjoint sets of nodes A and B in G, we say $A \Rightarrow_l B$ if and only if there exists a node $i \in B$ such that $\kappa_l(A, i) \geq f + 1$; $A \not\Rightarrow_l B$ otherwise.*

Informally speaking, the relation $A \Rightarrow_l B$ captures the existence of a node $i \in B$ that can be influenced by fault-free nodes in A despite the presence of Byzantine nodes.

Let $F \subseteq \mathcal{V}(G)$ be a set of vertices in G. Denote the subgraph of G induced by vertex set $\mathcal{V}(G) - F$ by G_F.[1] We describe a necessary and sufficient condition

[1] Subgraph of G induced by vertex set $S \subseteq \mathcal{V}(G)$ is the subgraph H with vertex set S such that $\mathcal{E}(H) = \{(u, v) \in \mathcal{E}(G) : u, v \in S\}$. Recall that $\mathcal{V}(\cdot)$ and $\mathcal{E}(\cdot)$ are the vertex set and edge set, respectively, of a given graph.

below, whose necessity is proved in Theorem 1 and sufficiency is shown constructively in Section 4. For ease of future reference, we termed the condition as *Condition NC*.

Condition NC: For any node partition L, C, R, F of G such that $L \neq \emptyset, R \neq \emptyset$ and $|F| \leq f$, in G_F, at least one of the two conditions below must be true: (i) $R \cup C \Rightarrow_l L$; (ii) $L \cup C \Rightarrow_l R$.

Intuitively, Condition NC requires that, for any node partition L, C, R, F, either the nodes in $R \cup C$ are able to collectively influence a node in L in G_F or vice versa. Note that when $l = 1$, Condition NC is equivalent to the following condition, which is shown to be both necessary and sufficient without message relay, i.e., $l = 1$, in [14].
" For any node partition L, C, R, F of G such that $L \neq \emptyset, R \neq \emptyset$ and $|F| \leq f$, in the induced subgraph G_F, at least one of the two conditions below must be true: (i) there exists a node $i \in L$ such that $\left|(R \cup C) \cap N_i^-\right| \geq f + 1$; (ii) there exists a node $j \in R$ such that $\left|(L \cup C) \cap N_j^-\right| \geq f + 1$."

Theorem 1. *Suppose that a correct IABC algorithm exists over G. Then G satisfies Condition NC.*

Our proof shares the structure of the proof of Theorem 1 in [14]. The basic idea is as follows: Suppose that the given graph G does not satisfy Condition NC and that there exists a correct IABC algorithm, say $\mathcal{A}$. Since G does not satisfy Condition NC, there exists a node partition L, R, C, F, where L, R are both non-empty and $|F| \leq f$ such that $L \cup C \not\Rightarrow_l R$ and $R \cup C \not\Rightarrow_l L$ in G_F. Consider the execution in which all the nodes in F are faulty and all the remaining nodes are fault-free. In addition, the input of each node in L is 0, the input of each node in R is 2ϵ, and the input of each node in C is an arbitrary value within the interval $[0, 2\epsilon]$. The faulty nodes in F can behave in such a way that each node $i \in L$ cannot determine whether nodes in F are faulty or nodes in the minimum l–restricted $(R \cup C, i)$–cut are faulty. This is possible, since $\kappa_l(R \cup C, i) \leq f$. Thus, to guarantee validity, node i will update its state $v_i[t] = 0$ for all t. Since i is an arbitrary node in L, we have $v_i[t] = 0$ for all $i \in L$ and all t. Similarly, we can show that $v_j[t] = 2\epsilon$ for all $j \in R$ and all t. Thus $|v_i[t] - v_j[t]| = 2\epsilon$ for all t, where $i \in L$ and $j \in R$, contradicting the assumption that $\mathcal{A}$ is a correct IABC algorithm. A formal proof of Theorem 1 can be found in the full version of the paper [10].

The above necessary condition is in general weaker than the necessary condition derived under single-hop message transmission model in [14], i.e., when $l = 1$. Consider the system depicted in Figure 1. The topology of this system does not satisfy the necessary condition derived in [14] for $l = 1$. Since in the node partition $L = \{p_1, p_4\}, R = \{p_2, p_3\}, C = \emptyset$ and $F = \{p_5\}$, neither $L \cup C \Rightarrow_l R$ in G_F nor $R \cup C \Rightarrow_l L$ in G_F holds for $l = 1$ and $f = 1$. However, via enumeration

it can be seen that the graph, depicted in Figure 1, satisfies Condition NC for $l \geq 2$ and $f = 1$.

It follows from the definition of Condition NC that if a graph G satisfies Condition NC for $l \in \{1, \ldots, n-1\}$, then G also satisfies Condition NC for all $l' \geq l$. Let l_0 be the smallest integer for which G satisfies Condition NC. In particular, if G does not satisfy Condition NC for any $l \in \{1, \ldots, n-1\}$, define $l_0 \triangleq n$ by convention. We observe that in general given a graph G, the diameter of G can be arbitrarily smaller than l_0. For instance, the diameter of the graph depicted in Figure 2 is 2. However, for the depicted graph, $l_0 \geq \frac{n+1}{4}$ when $n = 4k+3$, where k is a positive integer. So l_0 is much larger than 2 for large n. To see $l_0 \geq \frac{n+1}{4}$, consider the node partition $F = \{p_1\}, C = \emptyset, L = \{p_2, \ldots, p_{\frac{n+1}{2}}\}$ and $R = \{p_{\frac{n+3}{2}}, \ldots, p_n\}$. For $f = 1$, in order to have $L \cup C \Rightarrow_l R$ or $R \cup C \Rightarrow_l L$ hold in G_F for this particular node partition, it must be hold that $l \geq \frac{n+1}{4}$. Thus $l_0 \geq \frac{n+1}{4}$.

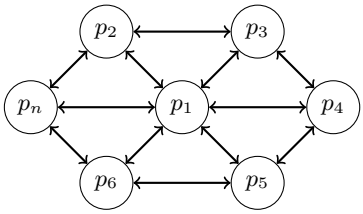

Fig. 2. In this system, there are n nodes $p_1, \ldots, p_n$; all communication links are bi-directional; and at most one node can be adversarial, i.e., $f = 1$. Nodes $p_2, \ldots, p_n$ form a cycle of length $n-1$ and these nodes are all connected to node p_1.

Similar to [14], as stated in our next corollary, our Condition NC for general l also implies a lower bound on n and a lower bound on each node's incoming degree. Moreover, these lower bounds are independent of l.

Corollary 1. *For $f > 0$, if G satisfies Condition NC, then n must be at least $3f + 1$, and each node must have at least $2f + 1$ incoming neighbors other than itself, i.e., $|N_i^- - \{i\}| \geq 2f + 1$.*

The proof of Corollary 1 is similar to the proof in [14], and can be found in [10]. Note that Corollary 1 also characterizes a lower bound on the density of G, that is $|\mathcal{E}(G)| \geq n(2f + 2)$, including self-loops, which is independent of the communication range l as well.

3.1 Equivalent Characterization of Condition NC

Informally speaking, Condition NC describes the information propagation property in terms of four set partitions. In this subsection, an equivalent condition of Condition NC is proposed, which is based on characterizing the structure

of a family of special subgraphs, termed as *reduced graphs*, of the power graph G^l. The new condition suggests that all fault-free nodes will be influenced by a collection of common fault-free nodes.

Definition 3. Meta-graph of SCCs: *Let $K_1, K_2, \ldots, K_k$ be the strongly connected components (i.e., SCCs) of G. The graph of SCCs of G, denoted by G^{SCC}, is defined as follows:*
(i) Nodes in G^{SCC} are $K_1, K_2, \ldots, K_k$; and
(ii) there is an edge (K_i, K_j) in G^{SCC} if there is some $u \in K_i$ and $v \in K_j$ such that (u, v) is an edge in G.
Strongly connected component K_h is said to be a source component *if the corresponding node in G^{SCC} is not reachable from any other node in G^{SCC}.*

It is known that the G^{SCC} is a directed acyclic graph, i.e., DAG [3]. It can be easily checked that due to the absence of directed cycles and finiteness, there exists at least one node in G^{SCC} that is not reachable from any other node. In particular, if G^{SCC} contains just one node, then that node is trivially the source. Thus, a graph G has at least one source component.

Definition 4. *The l–th power of a graph G, denoted by G^l, is a multigraph*[2] *with the same set of vertices as G and a directed edge between vertices u, v is defined by a path of length*[3] *l from u to v in G.*

The power graph G^l is a multigraph. There is a one-to-one correspondence between an edge e in G^l and a path of length l in G (including self-loops). A path of length 1 between vertices u and v in G exists if (u, v) is an edge in G. A path of length 2 between vertices u and v in G exists for every vertex w such that (u, w) and (w, v) are edges in G. Then for a given graph G with self-loop at each node, the $(u, v)^{th}$ element in the square of the adjacency matrix of G counts the number of paths of length at most 2 in G. Similarly, the $(u, v)^{th}$ element in the l–th power of the adjacency matrix of G gives the number of paths of length l between vertices u and v in G.

Let e be an edge in G^l, and let $P(e)$ be the corresponding path in G, we say an edge e in G^l is covered by node set S, if $\mathcal{V}(P(e)) \cap S \neq \emptyset$, i.e., path $P(e)$ passes through a node in S–recalling that $\mathcal{V}(P(e))$ is the vertex set of path $P(e)$.

Definition 5. *For a given graph G and $F \subseteq \mathcal{V}(G)$, let*

$$E = \{e \in \mathcal{E}(G^l) : \ \mathcal{V}(P(e)) \cap F \neq \emptyset\}$$

be the set of edges in G^l that are covered by node set F. For each node $i \in \mathcal{V}(G) - F$, choose $C_i \subseteq N_i^{l-} - \{i\}$ such that $|C_i| \leq f$. Let

$$E_i = \{e \in \mathcal{E}(G^l) : e \text{ is an incoming edge of node } i \text{ in } G^l \text{ and } \mathcal{V}(P(e)) \cap C_i \neq \emptyset\}$$

[2] A multigraph (or pseudograph) is a graph which is permitted to have multiple edges between each vertex pair, that is, edges that have the same end nodes. Thus two vertices may be connected by more than one edge.
[3] Recall that we assume that each node in G has a self-loop.

be the set of incoming edges of node i in G^l that are covered by node set C_i. A reduced graph of G^l, denoted by $\widetilde{G^l}_F$, is a subgraph of G^l whose node set and edge set are defined by (i) $\mathcal{V}(\widetilde{G^l_F}) = \mathcal{V}(G) - F$; and (ii) $\mathcal{E}(\widetilde{G^l_F}) = \mathcal{E}(G^l) - E - \cup_{i \in \mathcal{V}(G)-F} E_i$, respectively.

Note that for a given G and a given F, multiple reduced graphs may exist. Let us define set R_F to be the collection of all reduced graph of G^l for a given F, i.e.,

$$R_F = \{\widetilde{G^l}_F : \ \widetilde{G^l}_F \ \text{ is a reduced graph of } \ G^l\}. \tag{2}$$

G^l_F, the l–th power of the induced subgraph G_F, itself is a reduced graph of G^l, where we choose $C_i = \emptyset$ for each $i \in \mathcal{V}(G) - F$. Thus R_F is non-empty. In addition, $|R_F|$ is finite since the graph G is finite,

Theorem 2. *Graph G satisfies Condition NC if and only if every reduced graph $\widetilde{G^l}_F$ obtained as per Definition 5 contains exactly one* source component*.*

The proof of Theorem 2 is based on analogous proofs in [13,14], which can be found in [10].

4 Sufficiency: Algorithm 1

In this section we propose an algorithm, termed Algorithm 1 and show its correctness. First we introduce the definition of message cover that will be used frequently in this section.

Definition 6. *For a communication graph G, let $\mathcal{M}$ be a set of messages, and let $\mathcal{P}(\mathcal{M})$ be the set of paths corresponding to all the messages in $\mathcal{M}$, i.e., $\mathcal{P}(\mathcal{M}) = \{\mathsf{path}(m) | m \in \mathcal{M}\}$. A message cover of $\mathcal{M}$ is a set of nodes $\mathcal{T}(\mathcal{M}) \subseteq \mathcal{V}(G)$, such that for each path $P \in \mathcal{P}$, we have $\mathcal{V}(P) \cap \mathcal{T}(\mathcal{M}) \neq \emptyset$, i.e., each path P is covered by a node in $\mathcal{T}(\mathcal{M})$. In particular, a minimum message cover is defined by*

$$\mathcal{T}^*(\mathcal{M}) \in \arg \min_{\mathcal{T}(\mathcal{M}) \subseteq \mathcal{V}(G): \mathcal{T}(\mathcal{M}) \ \textit{is a cover of} \ \mathcal{M}} |\mathcal{T}(\mathcal{M})|.$$

Conversely, given a set of messages $\mathcal{M}_0$ and a set of nodes $\mathcal{T} \subseteq \mathcal{V}(G)$, a maximal set of messages $\mathcal{M} \subseteq \mathcal{M}_0$ that are covered by $\mathcal{T}$ is defined by,

$$\mathcal{M}^* \in \arg \max_{\mathcal{M} \subseteq \mathcal{M}_0: \mathcal{T} \ \textit{is a cover of} \ \mathcal{M}} |\mathcal{M}|.$$

Recall that $\mathcal{M}_i[t]$ is the collection of messages received by node i at iteration t. Let $\mathcal{M}'_i[t] = \mathcal{M}_i[t] - \{(v_i[t-1], (i,i))\}$. Sort messages in $\mathcal{M}'_i[t]$ in an increasing order, according to their message values, i.e., $\mathsf{value}(m)$ for $m \in \mathcal{M}'_i[t]$. Let $\mathcal{M}_{is}[t]$ be the largest sized subset of $\mathcal{M}'_i[t]$ such that (i) for all $m \in \mathcal{M}'_i[t] - \mathcal{M}_{is}[t]$ and $m' \in \mathcal{M}_{is}[t]$ we have $\mathsf{value}(m) \geq \mathsf{value}(m')$, and (ii) the cardinality of a minimum

cover of $\mathcal{M}_{is}[t]$ is exactly f, i.e., $|\mathcal{T}^*(\mathcal{M}_{is}[t])| = f$. Similarly, we define $\mathcal{M}_{il}[t]$ to be the largest sized subset of $\mathcal{M}'_i[t]$ as follows: (i) for all $m \in \mathcal{M}'_i[t] - \mathcal{M}_{il}[t]$ and $m'' \in \mathcal{M}_{il}[t]$ we have $\mathsf{value}(m) \leq \mathsf{value}(m'')$, and (ii) the cardinality of a minimum cover of $\mathcal{M}_{il}[t]$ is exactly f, i.e., $|\mathcal{T}^*(\mathcal{M}_{il}[t])| = f$. In addition, define $\mathcal{M}^*_i[t] = \mathcal{M}'_i[t] - \mathcal{M}_{is}[t] - \mathcal{M}_{il}[t]$.

Theorem 3. *Suppose that graph G satisfies Condition NC, then the sets of messages $\mathcal{M}_{is}[t]$, $\mathcal{M}_{il}[t]$ are well-defined and $\mathcal{M}^*_i[t]$ is non-empty for $f > 0$.*

This theorem is proved by construction, i.e., an algorithm is constructed to find the sets $\mathcal{M}_{is}[t]$, $\mathcal{M}_{il}[t]$ for a given $\mathcal{M}'_i[t]$. Details of the algorithm and its correctness proof can be found in [10]. 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 satisfies Condition NC. This implies that Condition NC is also sufficient. *Algorithm 1* has the three-step structure described in Section 2. With the exception of the update step (3) below, the algorithm is similar to the consensus algorithms in [8,14].

Algorithm 1

1. *Transmit step:* Transmit messages of the form $(v_i[t-1], P)$ on each l–hop path P (including self-loops) to nodes in N_i^{l+}. If node i is an intermediate node on path P for some message of the form $(\cdot, P)$, then node i forwards that to the next node on path P.
2. *Receive step:* Receive messages from N_i^{l-} for which destination is i. When node i expects to receive a message from a path but does not receive the message, the message value is assumed to be equal to some default value.[4]
3. *Update step:*
   Define

$$v_i[t] \;=\; Z_i(\mathcal{M}_i[t]) \;= a_i v_i[t-1] + \sum_{m \in \mathcal{M}^*_i[t]} a_i\, w_m. \tag{3}$$

   where $w_m = \mathsf{value}(m)$ and $a_i = \frac{1}{|\mathcal{M}^*_i[t]|+1}$.

Note that in Step 3, only messages in $\mathcal{M}^*_i[t]$ and the value $v_i[t-1]$ are used in updating v_i in (3). Messages in both $\mathcal{M}_{is}[t]$ and $\mathcal{M}_{il}[t]$ are trimmed away. This trimming strategy is motivated by the observation that the messages in $\mathcal{M}_{is}[t]$ (or $\mathcal{M}_{il}[t]$) may be tampered by nodes in $\mathcal{T}^*(\mathcal{M}_{is}[t])$ (or $\mathcal{T}^*(\mathcal{M}_{il}[t])$). These faulty behaviors are possible because of the fact that $|\mathcal{T}^*(\mathcal{M}_{is}[t])| = f$ and $|\mathcal{T}^*(\mathcal{M}_{il}[t])| = f$. Recall $\mathcal{M}^*_i[t] = \mathcal{M}'_i[t] - \mathcal{M}_{is}[t] - \mathcal{M}_{il}[t]$. The "weight"

[4] Note that node i does not read the message value if the message destination is not i.

of each term on the right-hand side of (3) is a_i, where $0 < a_i \leq 1$, and these weights add to 1. For future reference, let us define α, which is used in Theorem 4, as:

$$\alpha = \min_{i \in \mathcal{V}-\mathcal{F}} a_i. \tag{4}$$

In *Algorithm 1*, each fault-free node i's state, $v_i[t]$, is updated as a convex combination of all the *messages values* collected by node i at round t. In particular, for each message $m \in \mathcal{M}'[t]$, its coefficient is a_i if the message is in $\mathcal{M}_i^*[t]$ or the message is sent via self-loop of node i; otherwise, the coefficient of m is zero. The update step in *Algorithm 1* is a generalization of the update steps proposed in [8,13,14,16], where the update summation is over all the incoming neighbors of node i instead of over message routes. In [8,13,14,16], only single-hop communication is allowed, i.e., $l = 1$, and the fault-free node i can receive only one message from its incoming neighbor. With multi-hop communication, fault-free node can possibly receive messages from a node via multiple routes. Our trim function in *Algorithm 1* takes the possible multi-route messages into account.

4.1 Matrix Representation of Algorithm 1

With our trim function, the iterative update of the state of a fault-free node i admits a matrix representation of states evolution of fault-free nodes. We use boldface upper case letters to denote matrices, rows of matrices, and their entries. For instance, $\mathbf{A}$ denotes a matrix, $\mathbf{A}_i$ denotes the i-th row of matrix $\mathbf{A}$, and $\mathbf{A}_{ij}$ denotes the element at the intersection of the i-th row and the j-th column of matrix $\mathbf{A}$. Some useful concepts and theorems are reviewed briefly in [10].

Definition 7. *A vector is said to be* stochastic *if all the entries of the vector are* non-negative*, and the entries add up to 1. A matrix is said to be row stochastic if each row of the matrix is a stochastic vector.*

Recall that $\mathcal{F}$ is the set of faulty nodes. Let $|\mathcal{F}| = \phi$. Without loss of generality, suppose that nodes 1 through $(n - \phi)$ are fault-free, and if $\phi > 0$, nodes $(n - \phi + 1)$ through n are faulty. Denote by $\mathbf{v}[0] \in \mathbb{R}^{n-\phi}$ the column vector consisting of the initial states of all the *fault-free* nodes. Denote by $\mathbf{v}[t]$, where $t \geq 1$, the column vector consisting of the states of all the *fault-free* nodes at the end of the t-th iteration, $t \geq 1$, where the i-th element of vector $\mathbf{v}[t]$ is state $v_i[t]$.

Theorem 4. *We can express the iterative update of the state of a fault-free node* i $(1 \leq i \leq n - \phi)$ *performed in (3) using the matrix form in (5) below, where* $\mathbf{M}_i[t]$ *satisfies the four conditions listed below. In addition to* t*, the row vector* $\mathbf{M}_i[t]$ *may depend on the state vector* $\mathbf{v}[t-1]$ *as well as the behavior of the faulty nodes in* $\mathcal{F}$*. For simplicity, the notation* $\mathbf{M}_i[t]$ *does not explicitly represent this dependence.*

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

1. $\mathbf{M}_i[t]$ *is a* stochastic *row vector of size* $(n-\phi)$. *Thus,* $\mathbf{M}_{ij}[t] \geq 0$, *where* $1 \leq j \leq n-\phi$, *and*
$$\sum_{1 \leq j \leq n-\phi} \mathbf{M}_{ij}[t] \;=\; 1$$
2. $\mathbf{M}_{ii}[t] \geq a_i \geq \alpha$.
3. $\mathbf{M}_{ij}[t]$ *is non-zero only if there exists a message* $m \in \mathcal{M}_i[t]$ *such that* $\mathsf{source}(m) = j$ *and* $\mathsf{destination}(m) = i$.
4. *For any* $t \geq 1$, *there exists a reduced graph* $\widetilde{G^l}_{\mathcal{F}} \in R_{\mathcal{F}}$ *with adjacent matrix* $\mathbf{H}[t]$ *such that* $\beta\,\mathbf{H}[t] \;\leq\; \mathbf{M}[t]$, *where* $\beta = \frac{1}{16n^{2l}}$.

In the full version of the paper [10], we prove the correctness of Theorem 4 by constructing $\mathbf{M}_i[t]$ for $1 \leq i \leq n-\phi$. Our proof follows the same line of analysis as in the proof of Claim 2 in [13]. Due to the complexity (in particular, the dependency of message covers) brought up by messages relay, we divide the universe into six cases to consider.

Theorem 5. Algorithm 1 *satisfies the validity and the convergence conditions.*

From the code of *Algorithm 1*, we know that

$$v_i[t] = a_i v_i[t-1] + \sum_{m \in \mathcal{M}_i^*[t]} a_i\, w_m, \tag{6}$$

where $a_i = \frac{1}{|\mathcal{M}_i^*[t]|+1}$. Theorem 4 states that we can rewrite (6) as

$$\sum_{j \in \mathcal{V}-\mathcal{F}} \mathbf{M}_{ij}[t] v_j[t-1],$$

where $\mathbf{M}_{ij}[t]$s together satisfy the preceding four conditions. By "stacking" (5) for different i, $1 \leq i \leq n-\phi$, we can represent the state update for all the fault-free nodes together using (7) below, where $\mathbf{M}[t]$ is a $(n-\phi) \times (n-\phi)$ row stochastic matrix, with its i-th row being equal to $\mathbf{M}_i[t]$ in (5).

$$\mathbf{v}[t] = \mathbf{M}[t]\ \mathbf{v}[t-1]. \tag{7}$$

By repeated application of (7), we obtain:

$$\mathbf{v}[t] = \left(\Pi_{\tau=1}^{t} \mathbf{M}[\tau] \right)\ \mathbf{v}[0].$$

As the backward product $\Pi_{\tau=1}^{t} \mathbf{M}[\tau]$ is a row-stochastic matrix, it holds that $\mu[0] \leq v_i[t] \leq U[0]$ for all $i = 1, \ldots, n-\phi$ and all t. Thus Algorithm 1 satisfies validity condition.

The convergence of $v_i[t]$ depends on the convergence of the backward product $\Pi_{\tau=1}^{t} \mathbf{M}[\tau]$. As a result of this, our convergence proof uses toolkit of weak-ergodic theory that is also adopted in prior work (e.g., [1,2,8,14]). The last condition in Theorem 4 plays an important role in the proof of Theorem 5. A formal proof of Theorem 5 is presented in [10].

5 Unbounded Path Length

In this section, we show that Condition NC is equivalent to some existing results for undirected graphs and directed graphs when path lengths are not constrained.

5.1 Undirected Graph with Unbounded Path Length

If G is undirected, it has been shown in [6], that $n \geq 3f+1$ and node-connectivity $2f+1$ are both necessary and sufficient for achieving Byzantine approximate consensus. Recall that l^* is the length of a longest cycle-free path in G. We will show that when $l \geq l^*$, our Condition NC is equivalent to the above conditions.

Theorem 6. *When $l \geq l^*$, if G is undirected, then $n \geq 3f+1$ and node-connectivity of G is at least $2f+1$ if and only if G satisfies Condition NC.*

Informally, if the node-connectivity of G, denoted by $\kappa(G)$, is at most $2f$, then we are able to show that there exists a node partition L, R, C, F, where L, R are both non-empty and $|F| \leq f$, such that, in G_F, neither $L \cup C \Rightarrow_l R$ nor $R \cup C \Rightarrow_l L$ holds. Conversely, if $n \geq 3f+1$ and $\kappa(G) \geq 2f+1$, using Expansion Lemma [15] we are able to show Condition NC holds. Formal proof is given in [10].

5.2 Directed Graph with Unbounded Path Length

Synchronous exact Byzantine consensus is considered in [12].

Definition 8. *[12] Given disjoint subsets A, B, where B is non-empty:*
(i) We say $A \rightarrow B$ if and only if set A contains at least $f+1$ distinct incoming neighbors of B. That is, $|\{i|\ (i,j) \in \mathcal{E}, i \in A, j \in B\}| > f$.
(ii) We say $A \not\rightarrow B$ iff $A \rightarrow B$ is not true.

The following necessary and sufficient condition is obtained in [12].

Theorem 7. *[12] Given a graph G, exact Byzantine consensus is solvable if and only if for any partition L, C, R, F of G, such that both L and R are non-empty, and $|F| \leq f$, either $L \cup C \rightarrow R$ in G_F, or $R \cup C \rightarrow L$ in G_F.*

We term this condition as Condition 1. Note that in order for $A \rightarrow B$ to hold, we only require that there are at least $f+1$ incoming neighbors of set B in set A. As a result of this observation, our Condition NC with $l = 1$ is, in general, strictly stronger than Condition 1. However, we prove the following result in [10].

Theorem 8. *Condition NC is equivalent to Condition 1 when $l \geq l^*$.*

6 Summary and Discussion

In this paper, we assume that each node knows the topology within its l–hop neighborhood, and in each iteration it can send messages to nodes that are up to l hops away, where $l \geq 1$. We prove a necessary and sufficient condition for the existence of *iterative* algorithms that achieve *approximate Byzantine consensus* in directed graphs, while maintaining minimal memory across iterations. 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 within the range (or convex hull) of the initial inputs at the fault-free nodes.

Throughout the paper so far, we assumed that faulty nodes are only able to tamper message values, leaving message paths unchanged. However, this restriction of faulty behaviors of Byzantine nodes is not necessary. In fact, the above results still hold when both message value tampering and message path tampering are allowed, provided that (i) the number of faked messages is finite and there exists a constant C such that $\mathcal{M}_i[t] \leq C$ for all t (i.e., each faulty node $k \in \mathcal{F}$ cannot create too many non-existing messages), and that (ii) for each message m tampered/faked by a faulty node k, $\mathsf{path}(m)$ must satisfy $k \in \mathcal{V}(\mathsf{path}(m))$ (i.e., the faulty node k cannot conceal itself from the message path). The constraints (i) and (ii) can be implemented as follows. Recall that each fault-free node knows the network topology in its l–hop neighborhood. In each iteration, a fault-free node should accept any one message of the form (w, P) for any particular l–hop path P that is known to exist – if more than one such message is received, discard all but one such message (or discard all, and replace by a default value). Also, if node i receives the message (w, P) where path P is not known to exist, then node i should discard the message. These rules implement constraint (i) above. Suppose node i receives or relays a message $m = (w, P)$ from node j containing a path that does not have the form $\ldots ji \ldots$ then i will discard the message. This way, on any given l–hop path P, at least the very last faulty node will have to remain on the path (it may delete the earlier nodes on the path, but not itself). Thus the constraint (ii) is imposed. The necessity of Condition NC can be easily verified for the above behavior as well. It can also be seen that Algorithm 1 works under this relaxed model, proving the sufficiency of Condition NC.

Acknowledgements. The authors thank the referees and Lewis Tseng for providing constructive comments on the paper.

References

1. Ali, J., Jie, L., Morse, A.S.: Coordination of groups of mobile autonomous agents using nearest neighbor rules. IEEE Transactions on Automatic Control **48**(6), 988–1001 (2003)

2. Bnzit, F., Blondel, V., Thiran, P., Tsitsiklis, J., Vetterli, M.: Weighted gossip:Distributed averaging using non-doubly stochastic matrices. In: 2010 IEEE International Symposium on Information Theory Proceedings (ISIT), pp. 1753–1757, June 2010
3. Cormen, T.H., Leiserson, C.E., Rivest, R.L., Stein, C., et al.: Introduction toalgorithms, vol. 2. MIT Press Cambridge (2001)
4. Dolev, D., Lynch, N.A., Pinter, S.S., Stark, E.W., Weihl, W.E.: Reaching approximate agreement in the presence of faults. J. ACM **33**(3), 499–516 (1986)
5. Fekete, A.D.: Asymptotically optimal algorithms for approximate agreement. Distributed Computing **4**(1), 9–29 (1990)
6. Fischer, M.J., Lynch, N.A., Merritt, M.: Easy impossibility proofs for distributed consensus problems. In: Proceedings of the Fourth Annual ACM Symposium on Principles of Distributed Computing, PODC 1985, pp. 59–70. ACM, New York (1985)
7. Fischer, M.J., Lynch, N.A., Paterson, M.S.: Impossibility of distributed consensus with one faulty process. J. ACM **32**, 374–382 (1985)
8. LeBlanc, H.J., Zhang, H., Sundaram, S., Koutsoukos, X.: Consensus of multi-agent networks in the presence of adversaries using only local information. In: Proceedings of the 1st International Conference on High Confidence Networked Systems, HiCoNS 2012, pp. 1–10. ACM, New York (2012)
9. Pease, M., Shostak, R., Lamport, L.: Reaching agreement in the presence of faults. J. ACM **27**(2), 228–234 (1980)
10. Su, L., Vaidya, N.: Reaching approximate byzantine consensus with multi-hop communication (2014). arXiv preprint arXiv:1411.5282
11. Tseng, L., Vaidya, N.: Iterative approximate consensus in the presence of byzantine link failures. In: Noubir, G., Raynal, M. (eds.) NETYS 2014. LNCS, vol. 8593, pp. 84–98. Springer, Heidelberg (2014)
12. Tseng, L., Vaidya, N.H.: Fault-tolerant consensus in directed graphs. In: Proceedings of the 2015 ACM Symposium on Principles of Distributed Computing. ACM (to appear, 2015)
13. Vaidya, N.H.: Matrix representation of iterative approximate byzantine consensus in directed graphs. CoRR, arXiv:1203.1888 (2012)
14. Vaidya, N.H., Tseng, L., Liang, G.: Iterative approximate byzantine consensus in arbitrary directed graphs. In: Proceedings of the 2012 ACM Symposium on Principles of Distributed Computing, pp. 365–374. ACM (2012)
15. West, D.B., et al.: Introduction to graph theory, vol. 2. Prentice Hall, Upper Saddle River (2001)
16. Zhang, H., Sundaram, S.: Robustness of information diffusion algorithms to locally bounded adversaries. In: American Control Conference (ACC 2012), pp. 5855–5861 (2012)