Exact Byzantine Consensus in Directed Graphs*

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

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

Technical Report

August 24, 2012‡

Abstract

For synchronous point-to-point n-node networks of undirected links, it has been previously shown that, to achieve consensus in presence of up to f Byzantine faults, the following two conditions on the underlying communication graph are together necessary and sufficient: (i) $n \geq 3f+1$ and (ii) network connectivity greater than $2f$. The first condition, that is, $n \geq 3f+1$, is known to be necessary for directed graphs as well. So far, tight necessary and sufficient condition for Byzantine consensus in directed graphs has not been developed.

This paper presents tight necessary and sufficient condition for achieving Byzantine consensus in synchronous networks that can be represented as directed graphs. We provide a constructive proof of sufficiency by presenting a new Byzantine consensus algorithm for directed graphs.

Further work is needed to improve the message overhead of Byzantine consensus in directed graphs.

*This research is supported in part by Army Research Office grant W-911-NF-0710287. Any opinions, findings, and conclusions or recommendations expressed here are those of the authors and do not necessarily reflect the views of the funding agencies or the U.S. government.

†Revised September 4, 2012 to add Section 8 on example networks.

‡Revised October 10, 2012 to make minor improvements to the presentation.

1 Introduction

In a network of n nodes with up to f Byzantine faulty nodes, it is well-known that the following two conditions together are both necessary and sufficient for the existence of exact Byzantine consensus algorithms [5, 2] in networks of undirected links (i.e., undirected graphs).

- $n \geq 3f + 1$, and
- The connectivity of the underlying communication graph is at least $2f + 1$.

In this work, we consider algorithms for achieving exact Byzantine consensus in synchronous point-to-point networks that are modeled by arbitrary *directed* graphs, i.e., the communication between two neighboring nodes is not necessarily bi-directional. Consider a network of n nodes, of which at most f nodes may be Byzantine faulty. We assume that each node is given an initial input in $\{0, 1\}$. The Byzantine consensus algorithms of interest must satisfy the following three properties, where x_i denotes node i's input:

- **Termination**: every fault-free node i eventually decides on an output value y_i.
- **Agreement**: the output values of all the fault-free nodes are equal, i.e., there exists y such that, for every fault-free node i, $y_i = y$.
- **Validity**: for every fault-free node i, there exists a fault-free node k such that the output value $y_i = x_k$.

2 System Model

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

All the links (i.e., communication channels) are reliable, FIFO (first-in first-out) and deliver each transmitted message exactly once. When node i wants to send message M on link (i, j) to node j, it puts the message M in a send buffer for link (i, j). No further operations are needed at node i; the mechanisms for implementing reliable, FIFO and exactly-once semantics are transparent to the nodes. When a message is delivered on link (i, j), it becomes available to node j in a receive buffer for link (i, j). As stated earlier, the communication network is synchronous, and each message sent on link (i, j) is delivered to node j within a bounded interval of time.

Failure Model: We consider the Byzantine failure model, with up to f nodes becoming faulty. A faulty node may *misbehave* arbitrarily. Possible misbehavior includes sending incorrect and mismatching (or inconsistent) messages to different neighbors. The faulty nodes may potentially collaborate with each other. Moreover, the faulty nodes are assumed to have a complete knowledge

of the execution of the algorithm, including the states of all the nodes, contents of messages the other nodes send to each other, the algorithm specification, and the network topology.

3 Terminology

Upper case italic letters are used to name subsets of $\mathcal{V}$, and lower case italic letters are used to name nodes in $\mathcal{V}$.

Incoming neighbors:

- Node i is said to be an incoming neighbor of node j if $(i, j) \in \mathcal{E}$.
- For set $B \subseteq \mathcal{V}$, node i is said to be an incoming neighbor of set B if $i \notin B$, and there exists $j \in B$ such that $(i, j) \in \mathcal{E}$.
- Set B is said to have k incoming neighbors in set A if set A contains k distinct incoming neighbors of B.

Directed paths: All paths used in our discussion are directed paths.

- Paths from a node i to another node j:
  - An "(i, j)-path" is a directed path from node i to node j.
  - An "(i, j)-path excluding X" is a directed path from node i to node j that does not contain any node from set X.
  - Two paths from node i to node j are said to be "disjoint" if the two paths only have nodes i and j in common, with all remaining nodes being distinct.
  - The phrase "d disjoint (i, j)-paths" refers to d pairwise disjoint paths from node i to node j.
  - The phrase "d disjoint (i, j)-paths exluding X" refers to d pairwise disjoint (i, j)-paths that do not contain any nodes in set X.
- Every node i trivially has a path to itself. That is, for all $i \in \mathcal{V}$, (i, i)-path exists excluding $\mathcal{V} - \{i\}$.
- Paths from a set S to node $j \notin S$:
  - A path is said to be an "(S, j)-path" if it is an (i, j)-path for some $i \in S$.
  - An "(S, j)-path excluding X" is a (S, j)-path that does not contain any node from set X.
  - Two (S, j)-paths are said to be "disjoint" if the two paths only have node j in common, with all remaining nodes being distinct (including the first nodes on the paths).
  - The phrase "d disjoint (S, j)-paths" refers to d pairwise disjoint (S, j)-paths.
  - The phrase "d disjoint (S, j)-paths exluding X" refers to d pairwise disjoint (S, j)-paths that do not contain any nodes from set X.

Note that two disjoint (i, j)-paths are **not** disjoint $(\{i\}, j)$-paths. d disjoint (A, j)-paths can possibly exist for set A only if $|A| \geq d$.

For a directed path from node i to node j, node i will be called the "source" node on the path. Thus, for given d disjoint (A, b)-paths there are d distinct source nodes, all of which belong to A.

4 Necessary Condition

For a correct Byzantine consensus algorithm to exist, the networks graph $G(\mathcal{V}, \mathcal{E})$ must satisfy the necessary condition proved in this section. We state the necessary condition in two different forms in this section, and show that the two forms are equivalent. Later in Theorem 4 (in Section 5) we will state the necessary condition in a different form.

4.1 Necessary Condition: First Version

Relations $\stackrel{e}{\Rightarrow}$ and $\stackrel{e}{\not\Rightarrow}$ below are used frequently in our discussion.

Definition 1 *For disjoint sets*[1] *of nodes A and B, where B is non-empty:*

- $A \stackrel{e}{\Rightarrow} B$ *iff set A contains at least $f+1$ distinct incoming neighbors of B. That is,* $|\{i \mid (i,j) \in \mathcal{E},\ i \in A,\ j \in B\}| > f$.
- $A \stackrel{e}{\not\Rightarrow} B$ *iff* $A \stackrel{e}{\Rightarrow} B$ *is* not *true.*

Note that when $A = \Phi$, and $B \neq \Phi$, we have

$$A \stackrel{e}{\not\Rightarrow} B$$

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

Proof: The proof is presented in Appendix A. □

4.2 Necessary Condition: Second Version

Definition 2 *Given a partition A, B, F of $\mathcal{V}$ such that $|F| \leq f$, set A is said to propagate in $\mathcal{V} - F$ to set B if either (i) $B = \Phi$, or (ii) for each node $b \in B$, there exist at least $f+1$ disjoint (A, b)-paths excluding F.*

We will denote the fact that set A propagates in $\mathcal{V} - F$ to set B by the notation

$$A \stackrel{\mathcal{V}-F}{\rightsquigarrow} B$$

When it is not true that $A \stackrel{\mathcal{V}-F}{\rightsquigarrow} B$, we will denote that fact by

$$A \stackrel{\mathcal{V}-F}{\not\rightsquigarrow} B$$

[1] Sets A and B are said to be disjoint if $A \cap B = \Phi$. As per this definition, any set A is disjoint with empty set Φ.

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

Lemma 1 *Given a partition A, B, F of $\mathcal{V}$ such that B is non-empty, and $|F| \leq f$, if $A \overset{\mathcal{V}-F}{\rightsquigarrow} B$, then size of A must be at least $f+1$.*

Proof: By definition, there must be at least $f+1$ disjoint (A, b)-paths excluding F for each $b \in B$. Each of these $f+1$ disjoint paths will have a distinct *source* node in A. Therefore, such $f+1$ disjoint paths can only exist if A contains at least $f+1$ distinct nodes. □

We now state the second form of the necessary condition.

Theorem 2 *Suppose that a correct Byzantine consensus algorithm exists for $G(\mathcal{V}, \mathcal{E})$. Then for any partition A, B, F of $\mathcal{V}$, where A and B are both non-empty, and $|F| \leq f$, either $A \overset{\mathcal{V}-F}{\rightsquigarrow} B$ or $B \overset{\mathcal{V}-F}{\rightsquigarrow} A$.*

Proof: Suppose that a correct Byzantine consensus algorithm exists for $G(\mathcal{V}, \mathcal{E})$. Therefore, G must satisfy the condition in Theorem 1. Theorem 2 is proved below using Lemmas 2 through 4. Lemmas 2 through 4 together prove that the condition in Theorem 1 implies the condition in Theorem 2. □

Lemma 2 *Assume that the condition in Theorem 1 holds for $G(\mathcal{V}, \mathcal{E})$. For any partition A, B, F of $\mathcal{V}$, where A is non-empty, and $|F| \leq f$, if $B \overset{\mathcal{G}}{\not\Rightarrow} A$, then $A \overset{\mathcal{V}-F}{\rightsquigarrow} B$.*

Proof: If $B = \Phi$, then by Definition 2, the lemma is trivially true. In the rest of this proof, assume that $B \neq \Phi$.

Add a new (virtual) node v to graph G, such that, (i) v has no incoming edges, (ii) v has an outgoing edge to each node in A, and (iii) v has no outgoing edges to any node that is not in A. Let G_{+v} denote the graph resulting after the addition of v to $G(\mathcal{V}, \mathcal{E})$ as described above.

We want to prove that $A \overset{\mathcal{V}-F}{\rightsquigarrow} B$. Equivalently,[3] we want to prove that, in graph G_{+v}, for each $b \in B$, there exist $f+1$ disjoint (v, b)-paths excluding F. We will prove this claim by contradiction.

Suppose that $A \overset{\mathcal{V}-F}{\not\rightsquigarrow} B$, and therefore, there exists a node $b \in B$ such that there are at most f disjoint (v, b) paths excluding F in G_{+v}. By construction, there is no direct edge from v to b. Then Menger's theorem [8] implies that there exists a set $F_1 \subseteq (A \cup B) - \{b\}$ with $|F_1| \leq f$, such that, in graph G_{+v}, there is no (v, b)-path excluding $F \cup F_1$. In other words, all (v, b)-paths exluding F contain at least one node in F_1.

Let us define the following sets L, R, C:

- $L = A$.

  L is non-empty, because A is non-empty.

[3] *Justification*: Suppose that $A \overset{\mathcal{V}-F}{\rightsquigarrow} B$. By the definition of $A \overset{\mathcal{V}-F}{\rightsquigarrow} B$, for each $b \in B$, there exist at least $f+1$ disjoint (A, b)-paths excluding F – these paths only share node b. Since v has outgoing links to all the nodes in A, this implies that there exist $f+1$ disjoint (v, b)-paths excluding F in G_{+v} – these paths only share nodes v and b. Now, let us prove the converse. Suppose that there exist $f+1$ disjoint (v, b)-paths excluding F in G_{+v}. Node v has outgoing links only to the nodes in A, therefore, from the $(f+1)$ disjoint (v, b)-paths excluding F, if we delete node v and its outgoing links, then the shortened paths are disjoint (A, b)-paths excluding F.

- $R = \{ i \mid i \in B - F_1$ and there exists (i, b)-path excluding $F \cup F_1\}$.

  Thus, $R \subseteq B - F_1 \subseteq B$.
  Note that $b \in R$. Thus, R is non-empty.

- $C = B - R$.

  Thus, $C \subseteq B$. Since $R \subseteq B$, it follows that $R \cup C = B$.

Observe that L, R, C are disjoint sets, and $L \cup R \cup C = A \cup B$. Since set $F_1 \subseteq A \cup B$, $L = A$, and $R \cap F_1 = \Phi$, we have $F_1 \subseteq L \cup C$, and $F_1 \cap B \subseteq C$. Thus, set C can be partitioned into disjoint sets B_1 and B_2 such that

- $B_1 = C \cap F_1 = B \cap F_1 \subseteq C \subseteq B$, and
- $B_2 = C - B_1 \subseteq C \subseteq B$. Note that $B_2 \cap F_1 = \Phi$.

We make the following observations:

- For any $x \in A - F_1 = L - F_1$ and $y \in R$, $(x, y) \notin \mathcal{E}$

  Justification: Recall that virtual node v has a directed edge to x. If edge (x, y) were to exist then there would be a (v, b)-path via nodes x and y excluding $F \cup F_1$ (recall that y has a path to b excluding $F \cup F_1$). This contradicts the definition of set F_1.

- For any $p \in B_2$, and $q \in R$, $(p, q) \notin \mathcal{E}$

  Justification: If edge (p, q) were to exist, then there would be a (p, b)-path via node q excluding $F \cup F_1$, since q has a (q, b)-path excluding $F \cup F_1$. Then node p should have been in R by the definition of R. This is a contradiction to the assumption that $p \in B_2$, since $B_2 \cap R \subseteq C \cap R = \Phi$.

Thus, all the incoming neighbors of set R are contained in $F \cup F_1$ (note that $F_1 = (A \cap F_1) \cup B_1$). Recall that $F_1 \subseteq L \cup C$. Since $|F_1| \leq f$, it follows that

$$L \cup C \stackrel{\mathcal{E}}{\not\Rightarrow} R \tag{1}$$

By assumption in the lemma, $B \stackrel{\mathcal{E}}{\not\Rightarrow} A$. By definitions of L, R, C above, we have $A = L$ and $B = C \cup R$. Thus,

$$C \cup R \stackrel{\mathcal{E}}{\not\Rightarrow} L \tag{2}$$

(1) and (2) contradict the condition in Theorem 1. Thus, we have proved that $A \overset{\mathcal{V}-F}{\rightsquigarrow} B$. □

Lemma 3 *Assume that the condition in Theorem 1 holds for $G(\mathcal{V}, \mathcal{E})$. Consider a partition A, B, F of $\mathcal{V}$, where A, B are both non-empty, and $|F| \leq f$. If $B \overset{\mathcal{V}-F}{\not\rightsquigarrow} A$ then there exist A' and B' such*

- *A' and B' are both non-empty,*
- *A' and B' form a partition of $A \cup B$,*
- *$A' \subseteq A$ and $B \subseteq B'$, and*
- *$B' \stackrel{\mathcal{E}}{\not\Rightarrow} A'$.*

Proof: Suppose that $B \overset{\mathcal{V}-F}{\not\rightarrow} A$.

Add a new (virtual) node w to graph G, such that, (i) w has no incoming edges, (ii) w has an outgoing edge to each node in B, and (iii) w has no outgoing edges to any node that is not in B. Let G_{+w} denote the graph resulting after addition of w to $G(\mathcal{V}, \mathcal{E})$ as described above.

Since $B \overset{\mathcal{V}-F}{\not\rightarrow} A$, for some node $a \in A$ there exist at most f disjoint (B, a)-paths excluding F. Therefore, there exist at most f disjoint (w, a)-paths excluding F in G_{+w}.[4] Also by construction, $(w, a) \notin \mathcal{E}$. Then, by Menger's theorem [8], there must exist $F_1 \subseteq (A \cup B) - \{a\}$, $|F_1| \leq f$, such that, in graph G_{+w}, all (w, a)-paths excluding F contain at least one node in F_1.

Define the following sets (also recall that $\mathcal{V} - F = A \cup B$):

- $L = \{ i \mid i \in \mathcal{V} - F - F_1$ and there exists an (i, a)-path excluding $F \cup F_1 \}$
- $R = \{ j \mid j \in \mathcal{V} - F - F_1$ and there exists in G_{+w} a (w, j)-path excluding $F \cup F_1 \}$

  Set R contains $B - F_1$ since all nodes in B have edges from w.
- $C = \mathcal{V} - F - L - R = (A \cup B) - L - R$. Observe that $F_1 \subseteq C$ (because nodes of F_1 are not in $L \cup R$). Also, by definition of C, sets C and $L \cup R$ are disjoint.

Observe the following:

- Sets L and R are disjoint, and set $L \subseteq A - F_1 \subseteq A$.

  Justification: $F_1 \cap L = F_1 \cap R = \Phi$. By definition of F_1, all (w, a)-paths excluding F contain at least one node in F_1. If $L \cap R$ were to be non-empty, we can find a (w, a)-path excluding $F \cup F_1$, which is a contradiction.

  Note that $\mathcal{V} - F - F_1 = (A \cup B) - F_1$; therefore, $L \subseteq (A \cup B) - F_1$. $B - F_1 \subseteq R$, since all nodes in $B - F_1$ have links from w. Since L and R are disjoint, it follows that $(B - F_1) \cap L = \Phi$, and therefore, $(A - F_1) \cap L = L$; that is, $L \subseteq A - F_1 \subseteq A$.
- For any $x \in C - F_1$ and $y \in L$, $(x, y) \notin \mathcal{E}$.

  Justification: If such a link were to exist, then x should be in L, which is a contradiction (since C, L are disjoint).
- There are no links from nodes in R to nodes in L.

  Justification: If such a link were to exist, it would contradict the definition of F_1, since we can now find a (w, a)-path excluding $F \cup F_1$.

Thus, all the incoming neighbors of set L must be contained within $F \cup F_1$. Recall that $F_1 \subseteq C$ and $|F_1| \leq f$. Thus,

$$R \cup C \overset{\mathcal{G}}{\not\Rightarrow} L \tag{3}$$

Now define, $A' = L$, $B' = R \cup C$. Observe the following:

- A' and B' form a partition of $A \cup B$.

  Justification: L, R, C are disjoint sets, therefore $A' = L$ and $B' = R \cup C$ are disjoint. By the definition of sets L, R, C it follows that $A' \cup B' = L \cup (R \cup C) = \mathcal{V} - F = A \cup B$.

[4] See footnote 3.

- A' is non-empty and $A' \subseteq A$.

  Justification: By definition of set L, set L contains node a. Thus, $A' = L$ is non-empty. We have already argued that $L \subseteq A$.

- B' is non-empty and $B \subseteq B'$.

  Justification: Recall that L, R, C are disjoint. Thus, by definition of C, $R \cup C = (A \cup B) - L$. Since $L \subseteq A$, it follows that $B \subseteq R \cup C = B'$. Also, since B is non-empty, B' is also non-empty.

- $B' \overset{\mathcal{E}}{\not\Rightarrow} A'$

  Justification: Follows directly from (3), and the definition of A', B'.

This concludes the proof. □

Lemma 4 *The condition in Theorem 1 implies the condition in Theorem 2.*

Proof: Assume that the condition in Theorem 1 is satisfied by graph $G(\mathcal{V}, \mathcal{E})$. Consider a partition of A, B, F of $\mathcal{V}$ such that A, B are non-empty and $|F| \leq f$. Then, we must show that either $A \overset{\mathcal{V}-F}{\rightsquigarrow} B$ or $B \overset{\mathcal{V}-F}{\rightsquigarrow} A$.
Consider two possibilities:

- $B \overset{\mathcal{V}-F}{\rightsquigarrow} A$: In this case the proof is complete.
- $B \overset{\mathcal{V}-F}{\not\rightsquigarrow} A$: Then by Lemma 3 there exist non-empty sets A', B' that form a partition of $A \cup B$ such that $A' \subseteq A$, $B \subseteq B'$, and $B' \overset{\mathcal{E}}{\not\Rightarrow} A'$. Lemma 2 then implies that $A' \overset{\mathcal{V}-F}{\rightsquigarrow} B'$. Since $A' \subseteq A$ and $A \cup B = A' \cup B'$, it follows that $A \overset{\mathcal{V}-F}{\rightsquigarrow} B$.[5]

□

Lemma 5 *The condition in Theorem 2 implies the condition in Theorem 1.*

Proof: We will prove that the statement by proving that if the condition in Theorem 1 is violated, then the condition in Theorem 2 is violated as well.

Suppose that the condition in Theorem 1 is violated. Then there exists a partition L, R, C, F of $\mathcal{V}$ such that L, R are both non-empty, $|F| \leq f$,

$$L \cup C \overset{\mathcal{E}}{\not\Rightarrow} R$$

and

$$R \cup C \overset{\mathcal{E}}{\not\Rightarrow} L.$$

[5] *Explanation*: Since $A' \overset{\mathcal{V}-F}{\rightsquigarrow} B'$, for each $b \in B'$, there exist $f+1$ disjoint (A', b)-paths excluding F. Since $B \subseteq B'$, it then follows that, for each $b \in B \subseteq B'$ there exist $f+1$ disjoint (A', b)-paths excluding F. Since $A' \subseteq A$, and $F \cap A = \Phi$, each (A', b)-path excluding F is also a (A, b)-path excluding F. Thus, for each $b \in B$ there exist $f+1$ disjoint (A, b)-paths excluding F. This implies that $A \overset{\mathcal{V}-F}{\rightsquigarrow} B$.

Since $L \cup C \overset{\mathcal{G}}{\not\Rightarrow} R$, for any node $r \in R$, there exists a set F_r, $|F_r| \leq f$, such that all the $(L \cup C, r)$-paths excluding F contain at least one node in F_r. Since L is a subset of $L \cup C$, Menger's theorem [8] implies that there are at most f disjoint (L, r)-paths excluding F. Since $r \in R \cup C$,

$$L \overset{\mathcal{V}-F}{\not\rightarrow} R \cup C$$

Similarly, since $R \cup C \overset{\mathcal{G}}{\not\Rightarrow} L$, for any node $l \in L$, there exists a set F_l, $|F_l| \leq f$, such that all the $(R \cup C, l)$-paths excluding F contain at least one node in F_l. Menger's theorem [8] then implies that there are at most f disjoint $(R \cup C, l)$-paths excluding F. Thus,

$$R \cup C \overset{\mathcal{V}-F}{\not\rightarrow} L$$

Define $A = L$, and $B = R \cup C$. Thus, A, B, F is a partition of $\mathcal{V}$ such that $|F| \leq f$ and A, B are non-empty. The two conditions derived above imply that $A \overset{\mathcal{V}-F}{\not\rightarrow} B$ and $B \overset{\mathcal{V}-F}{\not\rightarrow} A$, violating the condition in Theorem 2. □

Lemmas 4 and 5 imply that the conditions in Theorems 1 and 2 are equivalent.

4.3 Corollaries

Corollary 1 *Suppose that a correct Byzantine consensus algorithm exists for $G(\mathcal{V}, \mathcal{E})$. Then size of set $\mathcal{V}$ (i.e., n) must be at least $3f + 1$.*

Since $n \geq 3f + 1$ is a necessary condition for Byzantine consensus for undirected graphs [5], it follows that $n \geq 3f + 1$ is also necessary for directed graphs. This necessary condition can also be derived from Theorem 1 as follows.

Proof: For $f = 0$, the corollary is trivially true. Now consider $f > 0$. The proof is by contradiction. Suppose that $n \leq 3f$. As stated previously, we assume $n \geq 2$, since consensus for $n = 1$ is trivial. Partition $\mathcal{V}$ into three subsets L, R, F such that $|F| \leq f$, $0 < |L| \leq f$, and $0 < |R| \leq f$. Such a partition can be found because $2 \leq |\mathcal{V}| \leq 3f$. Define $C = \Phi$. Since L, R are both non-empty, and contain at most f nodes each, we have $L \cup C \overset{\mathcal{G}}{\not\Rightarrow} R$ and $R \cup C \overset{\mathcal{G}}{\not\Rightarrow} L$, violating the condition in Theorem 1. □

Corollary 2 *For $f > 0$, suppose that a correct Byzantine consensus algorithm exists for $G(\mathcal{V}, \mathcal{E})$. Then each node must have at least $2f + 1$ incoming neighbors.*

Proof: The proof is by contradiction. Suppose that for some node i, the number of incoming neighbors of i is at most $2f$. Partition $\mathcal{V} - \{i\}$ into two sets L and F such that L is non-empty and contains at most f incoming neighbors of i, and $|F| \leq f$. It should be easy to see that such L, F can be found.

Define $C = \Phi$ and $R = \{i\}$. Then, since $f > 0$ and $|R| = 1$, it follows that

$$R \cup C \overset{\mathcal{G}}{\not\Rightarrow} L$$

Also, since L contains at most f incoming neighbors of node i, and set R contains only node i,

$$L \cup C \overset{\mathcal{G}}{\not\Rightarrow} R$$

The above two conditions violate the condition in Theorem 1. □

Corollary 3 *For $f > 0$, suppose that the graph $G(\mathcal{V}, \mathcal{E})$ satisfies the condition in Theorem 1, and $|\mathcal{V}| = n = 3f + 1$. Then for any pair of nodes $i, j \in \mathcal{V}$, either $(i, j) \in \mathcal{E}$, or there exist at least $2f + 1$ disjoint (i, j)-paths in $G(\mathcal{V}, \mathcal{E})$.*

Proof: The proof is by contradiction. Suppose that there exist two nodes i and j such that $(i, j) \notin \mathcal{E}$, and there are at most $2f$ node-disjoint paths from i to j in G. Then according to Menger's theorem [8], there must exist a set of nodes $P \subseteq \mathcal{V} - (\{i\} \cup \{j\})$ such that $|P| \leq 2f$, and all (i, j)-paths contain at least one node in P.

Define sets $X \subset \mathcal{V} - P$, $Y \subset \mathcal{V} - P$ and Z as follows:

- $k \in X$ iff there exists a (i, k)-path excluding P. $i \in X$; thus X is non-empty.
- $k \in Y$ iff there exists a (k, j)-path excluding P. $j \in Y$; thus Y is non-empty.
- $Z = \mathcal{V} - X - Y - P$.

Observe that P is disjoint from X, Y, Z by definition, and Z is disjoint from X, Y, P also by definition. Now, we show that X and Y are disjoint. Suppose they are not disjoint, then there exists a node $x \in X \cap Y$. Then, by definition of X and Y, there exists a (i, j)-path excluding P via x, violating the definition of P. Thus, $X \cap Y = \Phi$. Hence, X, Y, Z, P form a partition of $\mathcal{V}$. Observe that there are no links from nodes in X to nodes in Y,[6] no links from nodes in Z to nodes in Y,[7] and no links from X to nodes in Z.[8]

Consider the following cases:

- $|Y| \leq f$: In this case, define F to be a subset of $\mathcal{V}$ such that $|F| = f$, and if $|P| \geq f$ then $F \subseteq P$, else $P \subseteq F$. Define $R = Y$, $L = \mathcal{V} - F - R$, and $C = \Phi$. By definition of X, Y, P it follows that all the incoming neighbors of R are either in $F \cap P$ or in $L \cap P$. By definition of F and the constraint that $|P| \leq 2f$, it follows that $|L \cap P| \leq f$. Therefore, $L \cup C \overset{\mathcal{G}}{\not\Rightarrow} R$. Also, because $|R| = |Y| \leq f$, we have $R \cup C \overset{\mathcal{G}}{\not\Rightarrow} L$.
- $f < |Y| \leq 2f$ and $|P| \leq f$: Define F such that $|F| = f$ and $F \subseteq Y$. Define $R = Y - F$, $L = \mathcal{V} - F - R$, and $C = \Phi$. Observe that $|R| = |Y| - f \leq f$. Therefore, $R \cup C \overset{\mathcal{G}}{\not\Rightarrow} L$. Also, $Y = R \cup F$. Thus, $L = X \cup Z \cup P$. There are no links from the nodes in $X \cup Z$ to the nodes in Y, and therefore, no links from the nodes in $X \cup Z$ to the nodes in R. Thus, the only incoming neighbors of R that are also in L are in P. Since $|P| \leq f$, the number of incoming neighbors of R in L is at most f. Also, $C = \Phi$. Therefore, $L \cup C \overset{\mathcal{G}}{\not\Rightarrow} R$.
- $f < |Y| \leq 2f$ and $f < |P| \leq 2f$: Define F such that $|F| = f$ and contains $|P| - f$ nodes in P and $2f - |P|$ nodes in Y. Define $R = Y - F$, $L = \mathcal{V} - F - R$, and $C = \Phi$. Observe that

[6] Else there would be a (i, j)-path excluding P.
[7] Else there would be a path excluding P from a node in Z to node j, violating the definition of Y and Z.
[8] Else there would be a path excluding P from node i to a node in Z, violating the definition of X and Z.

$|Y| = |\mathcal{V}| - |P| - |X| - |Z|$, and thus, $|R| = |Y| - (2f - |P|) = 3f + 1 - |P| - |X| - |Z| - (2f - |P|) = f + 1 - |X| - |Z|$. Since $i \in X$, and is non-empty, $|R| \leq f$. Thus, $R \cup C \overset{\mathcal{G}}{\not\Rightarrow} L$. Also, $L = X \cup Z \cup (P - F)$. There are no links from the nodes in $X \cup Z$ to the nodes in Y. Since $R \subseteq Y$, there are no links from the nodes in $X \cup Z$ to the nodes in R. Thus, the only incoming neighbors of R that are also in L are in $P - F$. By the definition of the sets, $|P - F| = |P| - (|P| - f) = f$. Hence, $L \cup C \overset{\mathcal{G}}{\not\Rightarrow} R$.

- $f < |Y| \leq 2f$ and $|P| > 2f$: This case is not possible because $Y \cap P = \Phi$, and $n = 3f + 1$.
- $|Y| > 2f$ and $|P| \leq f$: Define F such that $|F| = f$ and $F \subset Y$. Define R such that $|R| = f$ and $R \subset (Y - F)$. This is possible, since $|Y| > 2f$. Then, define $L = \mathcal{V} - F - R$, and $C = \Phi$. By definition, $|R| = f$. Therefore, $R \cup C \overset{\mathcal{G}}{\not\Rightarrow} L$. Also, there are no links from the nodes in $X \cup Z$ to the nodes in R. Thus, the only incoming neighbors of R that are also in L are in $P \cup (Y - F - R)$. Note that $|P \cup Y| \leq 3f$, since $n = 3f + 1$ and X is non-empty. By the definition of the sets, $|P \cup (Y - F - R)| = |P \cup Y| - |F \cup R| \leq 3f - 2f = f$. Hence, $L \cup C \overset{\mathcal{G}}{\not\Rightarrow} R$.
- $|Y| > 2f$ and $|P| > f$: This case is not possible because $Y \cap P = \Phi$, and $n = 3f + 1$.

In each case above, we have found a partition of the graph that violates the necessary condition stated in Theorem 1. Thus, Corollary 3 must be true.

□

5 Sufficiency Proof: Preliminaries

When $f = 0$, all the nodes are fault-free, and the proof of sufficiency is trivial. The necessary condition for $f = 0$ implies that there must exist at least one node, say node i, that has directed paths to all the remaining nodes in the network. Then consensus can be achieved simply by node i routing its input to all the other nodes, and adopting node i's input as the output for consensus. In the rest of our discussion below, we will assume that $f > 0$. We will show that the necessary conditions in Theorems 1 and 2 are also sufficient by providing an algorithm that achieves exact consensus in any graph that satisfies those conditions. In the rest of the discussion, we assume that graph $G(\mathcal{V}, \mathcal{E})$ satisfies the conditions in Theorems 1 and 2, even if this is not stated explicitly elsewhere below (recall that the two necessary conditions are equivalent). Also, by Corollaries 1 and 2, $n > 3f$, and the number of incoming neighbors of each node is at least $2f + 1$.

In this section, we first introduce some definitions that are useful in the presentation of the algorithm.

Definition 3 Graph decomposition: *Let H be a subgraph of $G(\mathcal{V}, \mathcal{E})$. Partition graph H into non-empty strongly connected components, $H_1, H_2, \cdots, H_h$, where h is a non-zero integer dependent on graph H, such that nodes $i, j \in H_k$ if and only if there exist (i, j)- and (j, i)-paths both excluding nodes outside H_k.*

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

It is known that the decomposition graph H^d is a directed *acyclic* graph [1].

Definition 4 Source component: *Let H be a directed graph, and let H^d be its decomposition as per Definition 3. Strongly connected component H_k of H is said to be a* source component *if the corresponding vertex c_k in H^d is <u>not</u> reachable from any other vertex in H^d.*

Definition 5 Reduced Graph: *For a given graph $G(\mathcal{V}, \mathcal{E})$, and sets $F \subset \mathcal{V}$, $F_1 \subset \mathcal{V} - F$, such that $|F| \leq f$ and $|F_1| \leq f$, reduced graph $G_{F,F_1}(\mathcal{V}_{F,F_1}, \mathcal{E}_{F,F_1})$ is defined as follows: (i) $\mathcal{V}_{F,F_1} = \mathcal{V} - F$, and (ii) $\mathcal{E}_{F,F_1}$ is obtained by removing from $\mathcal{E}$ all the links incident on the nodes in F, and all the outgoing links from nodes in F_1. That is, $\mathcal{E}_{F,F_1} = \mathcal{E} - \{(i,j) \mid i \in F \text{ or } j \in F\} - \{(i,j) \mid i \in F_1\}$.*

Theorem 3 *Suppose that graph $G(\mathcal{V}, \mathcal{E})$ satisfies the condition in Theorem 1. For graph $G(\mathcal{V}, \mathcal{E})$, every reduced graph obtained as per Definition 5 must contain exactly one* source component*.*

Proof: Consider $F \subset \mathcal{V}$, $F_1 \subset \mathcal{V} - F$ such that $|F| \leq f$ and $|F_1| \leq f$, as specified in Definition 5. Since F is a strict subset of $\mathcal{V}$, the reduced graph G_{F,F_1} contains at least one node; therefore, at least one source component must exist in G_{F,F_1}. We now prove that G_{F,F_1} cannot contain more than one source component. The proof is by contradiction. Suppose that the reduced graph G_{F,F_1} includes at least two source components.

Let the sets of nodes in two such source components of G_{F,F_1} be denoted L and R, respectively. Let $C = \mathcal{V} - F - L - R$. Observe that L, R, C, F form a partition of the nodes in $\mathcal{V}$. Since L is a source component in G_{F,F_1} it follows that there are no directed links in $\mathcal{E}_{F,F_1}$ from any node in $C \cup R$ to the nodes in L. Similarly, since R is a source component in G_{F,F_1} it follows that there are no directed links in $\mathcal{E}_{F,F_1}$ from any node in $L \cup C$ to the nodes in R. These observations, together with the manner in which $\mathcal{E}_{F,F_1}$ is defined, imply that in $G(\mathcal{V}, \mathcal{E})$: (i) set L has at most f distinct incoming neighbors in $C \cup R$, and (ii) set R has at most f distinct incoming neighbors in $L \cup C$.

Therefore, in graph $G(\mathcal{V}, \mathcal{E})$, $C \cup R \overset{\mathcal{G}}{\not\Rightarrow} L$ and $L \cup C \overset{\mathcal{G}}{\not\Rightarrow} R$, contradicting the condition in Theorem 1. Thus, G_{F,F_1} must contain exactly one source component. □

Corollary 4 *Suppose that graph $G(\mathcal{V}, \mathcal{E})$ satisfies the condition in Theorem 1. For any $F \subset \mathcal{V}$ and $F_1 \subset \mathcal{V} - F$, such that $|F| \leq f$ and $|F_1| \leq f$, let S denote the set of nodes in the source component of G_{F,F_1}. Then,*

$$S \overset{\mathcal{V}-F}{\rightsquigarrow} \mathcal{V} - F - S$$

Proof: Since G_{F,F_1} contains non-zero number of nodes, its source component S must be non-empty. If $\mathcal{V} - F - S$ is empty, then the corollary follows trivially by Definition 2. Suppose that $\mathcal{V} - F - S$ is non-empty. Since S is a source component in G_{F,F_1}, it has no incoming neighbors in G_{F,F_1}; therefore, all of the incoming neighbors of S in $\mathcal{V} - F$ in graph $G(\mathcal{V}, \mathcal{E})$ must belong to F_1. Since $|F_1| \leq f$, we have,

$$(\mathcal{V} - S - F) \overset{\mathcal{G}}{\not\Rightarrow} S$$

Lemma 2 then implies that

$$S \overset{\mathcal{V}-F}{\rightsquigarrow} \mathcal{V} - F - S$$

□

Definition 6 *For $F \subset \mathcal{V}$, graph G_{-F} is obtained by removing from $G(\mathcal{V}, \mathcal{E})$ all the nodes in F, and all the links incident on nodes in F.*

Lemma 6 *For any $F \subset \mathcal{V}$, $F_1 \subset \mathcal{V} - F$, such that $|F| \leq f$, $|F_1| \leq f$:*

- *The source component of G_{F,F_1} is strongly connected in G_{-F}.*
- *The source component of G_{F,F_1} does not contain any nodes in F_1.*

Proof: By Definition 3, each pair of nodes i, j in the source component of graph G_{F,F_1} has at least one (i,j)-path and at least one (j,i)-path consisting of nodes only in G_{F,F_1}, i.e., excluding nodes in F.

Since $F_1 \subset \mathcal{V} - F$, G_{F,F_1} contains other nodes besides F_1. Although nodes of F_1 belong to graph G_{F,F_1}, the nodes in F_1 do not have any outgoing links in G_{F,F_1}. Thus, any node in F_1 cannot have paths to any other node in G_{F,F_1}. Then, due to the connetedness requirement of a source component, it follows that no nodes of F_1 can be in the source component.

□

The reader may skip the rest of this section without loss of continuity. The material is not used in specifying our algorithm for exact consensus, or in proving its correctness.

Theorem 4 *Suppose that a correct Byzantine consensus algorithm exists for $G(\mathcal{V}, \mathcal{E})$. Then the following condition must hold:*

For any $F \subset \mathcal{V}$ and $F_x \subset \mathcal{V} - F$ such that $|F| \leq f$ and $|F_x| \leq f$, let S be the source component in the reduced graph G_{F,F_x} as per Definition 5. For any $F' \subset \mathcal{V} - F$ with $|F'| \leq f$, for every node $i \in \mathcal{V} - F - F' - S$, there exists in $G(\mathcal{V}, \mathcal{E})$ a (S,i)-path excluding $F \cup F'$. Note that F' may or may not equal to F_x.

Proof: We prove this theorem by showing that the condition in Theorem 1 implies the condition in Theorem 4 [9]. The proof is by contradiction. Suppose that the condition in Theorem 4 is not true. Then there exist $F \subset \mathcal{V}$, $F_x \subset \mathcal{V} - F$, and $F' \subset \mathcal{V} - F$, with $|F| \leq f$, $|F_x| \leq f$, $|F'| \leq f$, and S being the the source component of G_{F,F_x}, such that in graph $G(\mathcal{V}, \mathcal{E})$ there is no (S,i)-path excluding $F \cup F'$ for some node $i \in \mathcal{V} - F - F' - S$. Now, let us define

- $L = S$.

  L is non-empty due to the definition of source component.
- $R = \{j \mid j \in \mathcal{V} - F - F'$ and there exists a (j,i)-path excluding $F \cup F'\}$.

  Node $i \in R$ by the definition of (i,i)-path, and thus, R is also non-empty.
- $C = \mathcal{V} - F - L - R$.

Observe that F, L, R, C are disjoint and together form a partition of G such that $|F| \leq f$, and L, R are non-empty.

Recall that by definition, the source component does not have any incoming neighbors in G_{F,F_x} from $\mathcal{V} - F - S = \mathcal{V} - F - L = C \cup R$. Therefore, in $G(\mathcal{V}, \mathcal{E})$, $C \cup R \not\Rightarrow L$.

Then, we make the following observations:

[9] Theorem 4 can also be proved using Corollary 4.

- $S \cap R = \Phi$; otherwise, there is a (S, i)-path excluding $F \cup F'$, violating the assumption.
- For any $s \in S - F'$ and $j \in R$, $(s, j) \notin \mathcal{E}$; otherwise, there is a (S, i)-path excluding $F \cup F'$, violating the assumption.
- For any pair of nodes $c \in C - F'$ and $r \in R$, $(c, r) \notin \mathcal{E}$; otherwise, there is a (c, i)-path excluding $F \cup F'$ via node r, violating the definition of R.

Therefore, all the incoming neighbors of set R in $\mathcal{V} - F$ are contained in F'. Since $|F'| \leq f$, $L \cup C \overset{\mathcal{G}}{\not\Rightarrow} R$.

Thus, the partition L, R, C, F contradicts the condition in Theorem 1. □

Lemma 7 *Assume that graph $G(\mathcal{V}, \mathcal{E})$ satisfies the condition in Theorem 4. Then it also satisfies the condition in Theorem 1.*

Proof: The proof is by contradiction. Suppose that $G(\mathcal{V}, \mathcal{E})$ does not satisfy the condition in Theorem 1. Then there exists a partition L, R, C, F of $\mathcal{V}$, where L, R are non-empty and $|F| \leq f$ such that $L \cup C \overset{\mathcal{G}}{\not\Rightarrow} R$, and $R \cup C \overset{\mathcal{G}}{\not\Rightarrow} L$. That is,

- There exists $F_R \subseteq L \cup C$ such that $|F_R| \leq f$, and there is no $(L \cup C, i)$-path excluding $F \cup F_R$ for all $i \in R$, and
- There exists $F_L \subseteq R \cup C$ such that $|F_L| \leq f$, and there is no $(R \cup C, j)$-path excluding $F \cup F_L$ for all $j \in L$.

Note that F_R may or may not overlap with F_L.

Now, consider a reduced graph G_{F,F_R}. Since there is no $(L \cup C, i)$-path excluding $F \cup F_R$ for all $i \in R$, the corresponding source component S is a subset of R. This observation and the definition of F_L imply that there is no (S, j)-path excluding $F \cup F_L$ for any $j \in L$. This contradicts the condition in Theorem 4. □

The necessary conditions in Theorems 1, 2 and 4 are thus equivalent. In the next section, we will prove that these conditions are sufficient as well.

6 Algorithm BC

When $f = 0$, all the nodes are fault-free, and the proof of sufficiency of the necessary condition derived earlier is trivial. The necessary condition for $f = 0$ implies that there must exist at least one node, say node i, that has directed paths to all the remaining nodes in the network. Then consensus can be achieved simply by node i routing its input to all the other nodes, and adopting node i's input as the output for consensus. In the rest of this section, we will assume that $f > 0$.

We now present a new algorithm, named Algorithm BC, and prove that it correctly achieves Byzantine consensus. As shown below in the pseudo-code of Algorithm BC, the algorithm consists of two loops, an OUTER loop, and an INNER loop. The OUTER loop of the algorithm considers

each subset F of $\mathcal{V}$ such that $|F| \leq f$.[10] For each such F, the INNER loop examines each partition A, B of $\mathcal{V} - F$ such that A, B are both non-empty. For each such partition A, B, a non-empty set S is identified such that $S \subseteq \mathcal{V} - F$, and

$$S \overset{\mathcal{V}-F}{\rightsquigarrow} \mathcal{V} - F - S$$

.[11] The INNER loop uses sub-algorithms `Propagate` and `Equality`. These sub-algorithms make use of some state maintained by the nodes. We first discuss the node state, followed by the sub-algorithms.

6.1 Node State

Each node i maintains two state variables that are explicitly used in our algorithm: v_i and t_i. Each node will have to maintain other states as well (such as the routes to other nodes); however, we do not introduce additional notation for that.

- Variable v_i: Initially, v_i at any node i is equal to the input at node i. During the course of the algorithm, v_i at node i may be updated several times. Value v_i at the end of the algorithm represents node i's decision (or output) for Algorithm BC. The output at each node is either 0 or 1.

  At any time during the execution of the algorithm, the value v_i at node i is said to be *valid* if either of the following two conditions is true:

  - $v_i = 0$, and at least one fault-free node has input equal to 0
  - $v_i = 1$, and at least one fault-free node has input equal to 1

  Initial value v_i at a fault-free node i is valid because it equals its own input. Algorithm BC ensures that v_i at a fault-free node i always remains valid throughout the execution of the algorithm.

- Variable t_i: Variable t_i at any node i may take a value in $\{0, 1, \perp\}$, where $\perp$ is distinguished from 0 and 1. The `Propagate` and `Equality` procedures take t_i at participating nodes i as input, and may also modify t_i. Under some circumstances, v_i at node i is set equal to t_i in order to update v_i. We will discuss this in detail below.

[10] It also suffices to perform the outer loop for $|F| = f$.

[11] We discuss how to choose such S in subsection 6.4.

Algorithm BC

(OUTER LOOP)
For each $F \subset \mathcal{V}$, where $|F| \leq f$ and $f > 0$:

(INNER LOOP)
For each partition A, B of $\mathcal{V} - F$ such that A, B are non-empty, and $A \overset{\mathcal{V}-F}{\rightsquigarrow} B$:

STEP 1 of INNER loop:

- **Case 1:** $A \overset{\mathcal{V}-F}{\rightsquigarrow} B$ and $B \overset{\mathcal{V}-F}{\not\rightsquigarrow} A$:
  Let non-empty set $S \subseteq A$ be a set such that $S \overset{\mathcal{V}-F}{\rightsquigarrow} \mathcal{V} - F - S$, and S is strongly connected in G_{-F}.
  (a) For all $i \in S$, $t_i := v_i$
  (b) `Equality`(S)
  (c) `Propagate`$(S, \mathcal{V} - F - S)$
  (d) At each $j \in \mathcal{V} - F - S$: if $t_j \neq \perp$, then $v_j := t_j$
- **Case 2:** $A \overset{\mathcal{V}-F}{\rightsquigarrow} B$ and $B \overset{\mathcal{V}-F}{\rightsquigarrow} A$:
  Let non-empty set $S \subseteq A \cup B$ be a set such that $S \overset{\mathcal{V}-F}{\rightsquigarrow} \mathcal{V} - F - S$, S is strongly connected in G_{-F}, and $A \overset{\mathcal{V}-F}{\rightsquigarrow} (S - A)$.
  (e) For all nodes $i \in A$: $t_i = v_i$
  (f) `Propagate`$(A, S - A)$
  (g) `Equality`(S)
  (h) `Propagate`$(S, \mathcal{V} - F - S)$
  (i) At each $j \in \mathcal{V} - F - (A \cap S)$: if $t_j \neq \perp$, then $v_j := t_j$

STEP 2 of INNER loop:

(j) Each node $k \in F$ receives v_j from each $j \in N_k$, where N_k is a set consisting of $f + 1$ of k's incoming neighbors in $\mathcal{V} - F$. If all the received values are identical, then v_k is set equal to this identical value; else v_k is unchanged.

Figure 1: Algorithm BC (for $f > 0$): In the pseudo-code, := denotes the assignment operator.

6.2 Procedure Propagate(S, B)

Propagate(S, B) assumes that $S \subseteq \mathcal{V} - F$, $B \subseteq \mathcal{V} - F$, $S \cap B = \Phi$ and $S \overset{\mathcal{V}-F}{\rightsquigarrow} B$.

Propagate(S, B)

---

(1) Since $S \overset{\mathcal{V}-F}{\rightsquigarrow} B$, for each $i \in B$, there exist $f+1$ disjoint (S, i)-paths that exclude F. The source of each of these paths is in S; on each path, the corresponding source node, say node s, sends[12] t_s to node i along the corresponding path. Intermediate nodes on these paths forward received messages as necessary.

When a node does not receive an expected message, the message content is assumed to be $\perp$.

(2) When any node $i \in B$ receives $f+1$ values along the $f+1$ disjoint paths above: if the $f+1$ values are all equal to 0, then $t_i := 0$; else if the $f+1$ values are all equal to 1, then $t_i := 1$; else $t_i := \perp$.
(Note that := denotes the assignment operator.)

For all $j \notin B$, t_j is not modified during Propagate(S, B). Also, for all $k \in \mathcal{V}$, v_k is not modified during Propagate(S, B).

6.3 Equality(A)

Equality(A) assumes that $A \subseteq \mathcal{V} - F$, $A \neq \Phi$, and that for each pair of nodes $i, j \in A$, an (i, j)-path excluding F exists. That is, A is strongly connected in G_{-F} (G_{-F} is defined in Definition 6).

Equality(A)

---

(1) Each node $i \in A$ sends t_i to all other nodes in A along paths excluding F.

(2) Each node $j \in A$ thus receives messages from all nodes in A. Node j checks whether values received from all the nodes in A and its own t_j are all equal, and also belong to $\{0, 1\}$. If these conditions are *not* satisfied, then $t_j := \perp$; otherwise t_j is not modified.

For any node $k \notin A$, t_k is not modified in Equality(A). For any node $k \in \mathcal{V}$, v_k is not modified in Equality(A).

6.4 INNER Loop

For each F chosen in the OUTER loop, the INNER loop of Algorithm BC examines each partition A, B of $\mathcal{V} - F$ such that A, B are both non-empty. From the condition in Theorem 2, we know that either $A \overset{\mathcal{V}-F}{\rightsquigarrow} B$ or $B \overset{\mathcal{V}-F}{\rightsquigarrow} A$. Therefore, with renaming of the partitions we can ensure that $A \overset{\mathcal{V}-F}{\rightsquigarrow} B$. Then, depending on the choice of A, B, F, two cases may occur:

[12] All the nodes are aware of the "schedule" used for such transmissions, which is considered a part of the algorithm specification.

- Case 1: $A \overset{\mathcal{V}-F}{\rightsquigarrow} B$ and $B \overset{\mathcal{V}-F}{\not\rightsquigarrow} A$
- Case 2: $A \overset{\mathcal{V}-F}{\rightsquigarrow} B$ and $B \overset{\mathcal{V}-F}{\rightsquigarrow} A$

Now we will show that a suitable set S as required in each case in Algorithm BC exists:

- Case 1: $A \overset{\mathcal{V}-F}{\rightsquigarrow} B$ and $B \overset{\mathcal{V}-F}{\not\rightsquigarrow} A$:

  Since $B \overset{\mathcal{V}-F}{\not\rightsquigarrow} A$, by Lemma 3, there exist non-empty sets A', B' that form a partition of $A \cup B = \mathcal{V} - F$ such that $A' \subseteq A$ and

  $$B' \overset{\mathcal{G}}{\not\Rightarrow} A'$$

  Let F_1 be the set of incoming neighbors of A' in B'. Since $B' \overset{\mathcal{G}}{\not\Rightarrow} A'$, $|F_1| \leq f$. Then A' has no incoming neighbors in G_{F,F_1}. Therefore, the source component of G_{F,F_1} must be contained within A'. Let S denote the set of nodes in this source component. Since S is the source component, by Corollary 4,

  $$S \overset{\mathcal{V}-F}{\rightsquigarrow} \mathcal{V} - S - F.$$

  Since $S \subseteq A'$ and $A' \subseteq A$, $S \subseteq A$. Then, $B \subseteq (A \cup B) - S = \mathcal{V} - S - F$; therefore, $\mathcal{V} - S - F$ is non-empty. Also, since $S \overset{\mathcal{V}-F}{\rightsquigarrow} \mathcal{V} - S - F$, set S must be non-empty (by Lemma 1). By Lemma 6, S is strongly connected in G_{-F}.

- Case 2: $A \overset{\mathcal{V}-F}{\rightsquigarrow} B$ and $B \overset{\mathcal{V}-F}{\rightsquigarrow} A$:

  Since $|\mathcal{V}| = n > 3f$, $|A \cup B| = |\mathcal{V} - F| > 2f$. In this case, we pick an arbitrary non-empty set $F_1 \subset A \cup B = \mathcal{V} - F$ such that $|F_1| = f$, and find the source component of G_{F,F_1}. Let the set of nodes in the source component be denoted as S. Since S is the source component, by Corollary 4,

  $$S \overset{\mathcal{V}-F}{\rightsquigarrow} \mathcal{V} - F - S$$

  Also, since $A \overset{\mathcal{V}-F}{\rightsquigarrow} B$, and $(S - A) \subseteq B$, we have $A \overset{\mathcal{V}-F}{\rightsquigarrow} (S - A)$. Also, since $\mathcal{V} - S - F$ contains F_1, $\mathcal{V} - S - F$ is non-empty, and since $S \overset{\mathcal{V}-F}{\rightsquigarrow} \mathcal{V} - S - F$, set S must be non-empty (by Lemma 1). By Lemma 6, S is strongly connected in G_{-F}.

Now consider nodes in set F. As shown in Corollary 2, when $f > 0$, each node in $\mathcal{V}$ has at least $2f + 1$ incoming neighbors. Since $|F| \leq f$, for each $k \in F$ there must exist at least $f + 2$ incoming neighbors in $\mathcal{V} - F$. This satisfies the requirement in step (j) of Algorithm BC.

6.5 Correctness of BC

In the discussion below, assume that F^* is the set of faulty nodes in the network ($0 \leq |F^*| \leq f$ and $f > 0$).

When discussing a certain iteration of the INNER loop, we sometimes add superscript *start* and *end* to v_i for node i below to indicate whether we are referring to v_i at the start of that iteration, or at the end of that iteration.

Lemma 8 states that the state v_j of any fault-free node j at the end of an iteration of the INNER loop equals the state of some fault-free node at the start of that iteration.

Lemma 8 *For any given iteration of the INNER loop, for each fault-free $j \in \mathcal{V}$, there exists a fault-free node s such that $v_j^{end} = v_s^{start}$.*

Proof: We will first consider fault-free nodes in $\mathcal{V} - F$ in each of the two cases in the INNER loop, and then consider the fault-free nodes in F.

Define set Z as the set of values for v_i at all fault-free $i \in \mathcal{V}$ at the start of the INNER loop iteration under consideration.

$$Z = \{ v_i^{start} \mid i \in \mathcal{V} - F^* \}$$

- Case 1:

  Observe that, in Case 1, v_i remains unchanged for all fault-free $i \in S$; hence, the claim of the lemma is trivially true for fault-free $i \in S$.

  We will now prove the claim for fault-free $j \in \mathcal{V} - F - S$.

  - step (a): Consider a fault-free node $i \in S$. At the end of step (a), t_i is equal to v_i^{start}; thus $t_i \in Z$.
  - step (b): In step (b), `Equality`(S) either keeps t_i unchanged at fault-free node $i \in S$ or modifies it to be $\perp$. Thus, now $t_i \in Z \cup \{\perp\}$.
  - step (c): Consider a fault-free node $j \in \mathcal{V} - F - S$. During `Propagate`$(S, \mathcal{V} - F - S)$, j receives $f + 1$ values along $f + 1$ disjoint paths originating at nodes in S. Therefore, at least one of the $f + 1$ values is received along a path that contains only fault-free nodes; suppose that the value received by node j along this fault-free path is equal to α. As observed above in step (b), t_i at all fault-free nodes $i \in S$ is in $Z \cup \{\perp\}$; thus, $\alpha \in Z \cup \{\perp\}$. Therefore, at fault-free node $j \in \mathcal{V} - F - S$, `Propagate`$(S, \mathcal{V} - F - S)$ will result in $t_j \in \{\alpha, \perp\} \subseteq Z \cup \{\perp\}$.
  - step (d): Then it follows that, in step (d), at fault-free $j \in \mathcal{V} - F - S$, if v_j is updated, then $v_j^{end} \in Z$. On the other hand, if v_j is not updated, then $v_j^{end} = v_j^{start} \in Z$.

- Case 2:

  Observe that, in Case 2, v_j remains unchanged for all fault-free $j \in A \cap S$.

  Now we prove the claim in the lemma for fault-free $j \in \mathcal{V} - F - (A \cap S)$.

  - step (e): For any fault-free node $i \in A$, at the end of step (e), $t_i \in Z$.
  - step (f): Consider a fault-free node $m \in S - A$. During `Propagate`$(A, S - A)$, m receives $f + 1$ values along $f + 1$ disjoint paths originating at nodes in A. Therefore, at least one of the $f + 1$ values is received along a path that contains only fault-free nodes; suppose that the value received by node m along this fault-free path is equal to $\alpha \in Z$. Therefore, at node $m \in S - A$, `Propagate`$(A, S - A)$ will result in t_m being set to a value in $\{\alpha, \perp\} \subseteq Z \cup \{\perp\}$.

    Now, for $m \in S \cap A$, t_m is not modified in step (f), and therefore, for $m \in S \cap A$, $t_m \in Z$.

    Thus, we can conclude that, at the end of step (f), for all fault-free nodes $m \in S$, $t_m \in Z \cup \{\perp\}$.
  - step (g): In step (g), at each $m \in S$, `Equality`(S) either keeps t_m unchanged, or modifies it to be $\perp$. Thus, at the end of step (g), for all fault-free $m \in S$, t_m is in $Z \cup \{\perp\}$.
  - step (h): Consider a fault-free node $j \in \mathcal{V} - F - S$. During `Propagate`$(S, \mathcal{V} - F - S)$, j receives $f + 1$ values along $f + 1$ disjoint paths originating at nodes in S. Therefore, at least one of the $f + 1$ values is received along a path that contains only fault-free nodes;

suppose that the value received by node j along this fault-free path is equal to β. As observed above, after step (g), for each fault-free node $m \in S$, $t_m \in Z \cup \{\perp\}$. Therefore, $\beta \in Z \cup \{\perp\}$, and at node $j \in \mathcal{V} - F - S$, `Propagate`$(S, \mathcal{V} - F - S)$ will result in t_j being set to a value in $\{\beta, \perp\} \subseteq Z \cup \{\perp\}$.

- step (i): From the discussion of steps (g) and (h) above, it follows that, in step (i), if v_j is updated at a fault-free $j \in \mathcal{V} - F - (S \cap A)$, then $v_j^{end} \in Z$; on the other hand, if v_j is not modified, then $v_j^{end} = v_j^{start} \in Z$.

Now consider a fault-free node $k \in F$. As shown above, at the start of step (j), $v_j^{end} \in Z$ at all fault-free $j \in \mathcal{V} - F$. Since $|N_k| = f + 1$, at least one of the nodes in N_k is fault-free. Thus, of the $f + 1$ values received by node k, at least one value must be in Z. Now, consider two cases: On one hand, if node k changes v_k in step (j), then the new value will also in Z. On the other hand, if node k does not change v_k, then it remains in Z by the definition of Z.

□

Lemma 9 *Algorithm BC satisfies the validity condition for Byzantine consensus.*

Proof: Observe that for each fault-free $i \in \mathcal{V}$, initially, v_i is valid, because it is equal to the input at node i. Lemma 8 implies that after each iteration of the INNER loop of Algorithm BC, v_i remains valid at each fault-free node i. Therefore, when Algorithm BC terminates, v_i at each fault-free node i will satisfy the validity condition for Byzantine consensus. □

Lemma 10 *Algorithm BC satisfies the termination condition for Byzantine consensus.*

Proof: Recall that we are assuming a synchronous system, and the graph $G(\mathcal{V}, \mathcal{E})$ is finite. Thus, Algorithm BC performs a finite number of iterations of the OUTER loop, and a finite number of iteration of the INNER loop for each choice of F in the OUTER loop, the number of iterations being a function of graph G. Hence, Algorithm BC will terminate after a bounded amount of time. □

Lemma 11 *Algorithm BC satisfies the agreement condition for Byzantine consensus.*

Proof: Recall that F^* denotes the set of faulty nodes in the network ($0 \leq |F^*| \leq f$).

Since the OUTER loop considers all possible $F \subseteq \mathcal{V}$ such that $|F| \leq f$, eventually, the OUTER loop will be performed with $F = F^*$.

In the INNER loop for $F = F^*$, different partitions A, B of $\mathcal{V} - F = \mathcal{V} - F^*$ will be considered. We will say that such a partition A, B is a "conformant" partition if $v_i = v_j$ for all $i, j \in A$, and $v_i = v_j$ for all $i, j \in B$. A partition A, B that is not conformant is said to be "non-conformant". Further, we will say that an iteration is a "deciding" iteration if one of the following condition is true.

C1 : The partition considered in this iteration is conformant.

In Case 1 with conformant partition, every node in S has the same value t after step (a). Hence, in the end of step (b), every node in S has the same value t. Now, consider Case 2

with conformant partition. Denote the value of all the nodes in A by α ($\alpha \in \{0,1\}$). Then, in step (e), each node i in $S \cap A$ sets t_i equal to α. In step (f), all the nodes in $S \cap B$ receive identical values α from nodes in A, and hence, they set value t equal to α. Therefore, every node in S has the same value t in the end of step (g).

C2 : The partition considered in this iteration is non-conformant; however, in the end of step (b) of Case 1, and in the end of step (g) of Case 2, every node in the corresponding source component S has the same value t. That is, for all $i, j \in S, t_i = t_j$.

In both C1 and C2, all the nodes in the corresponding source component S have the identical value t in the deciding iteration (in the end of step (b) of Case 1, and in the end of step (g) of Case 2). The iteration that is not deciding is said to be "non-deciding".

Claim 1 *In the INNER loop for $F = F^*$, value v_i for each fault-free node i will stay unchanged in every non-deciding iteration.*

Proof: Suppose that $F = F^*$, and the iteration under consideration is a non-deciding iteration of the INNER loop. Observe that since the paths used in `Equality` and `Propagate` exclude F, none of the faulty nodes can affect the outcome of any INNER loop iteration when $F = F^*$. Thus, during `Equality`(S) (step (b) of Case 1 or step (g) of Case 2), each node in S can receive the value from other nodes in S correctly. Then, every node in S will set value t to be $\perp$ in the end of `Equality`(S), since by the definition of non-deciding iteration, there is a pair of nodes $j, k \in S$ such that $t_j \neq t_k$. Hence, every node in $\mathcal{V} - F - S$ will receive $f + 1$ copies of $\perp$ after `Propagate`($S, \mathcal{V} - F - S$) (step (c) of Case 1 and step (h) of Case 2), and will set value t to $\perp$. Finally, in the end of the iteration, the value v at each node stays unchanged based on the following two observations:

- nodes in S (in Case 1) or in $A \cap S$ (in Case 2) will not change value v as specified by Algorithm BC, and
- $t_i = \perp$ for each node $i \in \mathcal{V} - F - S$ (in Case 1) or for each node $i \in \mathcal{V} - F - (A \cap S)$.

Note that by assumption, there is no fault-free node in F, and hence, we do not need to consider STEP 2. Therefore, the statement is proved. □

Let us divide the iterations of the INNER loop for $F = F^*$ into three phases:

- Phase 1: Iterations of the INNER loop before the first deciding iteration
- Phase 2: The first deciding iteration
- Phase 3: Remaining iterations of the INNER loop for $F = F^*$.

Claim 2 *The INNER loop for $F = F^*$ will eventually enter Phase 2.*

Proof: By Lemma 9, it is always true that $v_i \in \{0, 1\}$ for each fault-free node i. Thus, when we perform the OUTER iteration for $F = F^*$, a conformant partition exists (in particular, set A containing all fault-free nodes with v value 0, and set B containing the remaining fault-free nodes, or vice-versa.) By Claim 1, nodes in $\mathcal{V} - F$ will not change values during non-deciding iterations. Then, since the INNER loop considers all partitions, the INNER loop will eventually consider either a conformant partition, or a non-conformant partition such that every node in the corresponding source component S has the same value t. □

Now, let us consider each phase separately:

- Phase 1: By Claim 1, the v_i at each fault-free node $i \in \mathcal{V}$ stays unchanged.
- Phase 2: Now, consider the first deciding iteration of the INNER loop.

  Recall that all the nodes in $\mathcal{V} - F = \mathcal{V} - F^*$ are fault-free. Let S be the corresponding source components in this iteration. We will show that in this iteration, every node in S will have the same t value. Consider two scenarios:

  - The partition is non-conformant: Then by definition of deciding iteration, we can find an $\alpha \in \{0, 1\}$ such that $v_i = \alpha$ for all $i \in S$ after step (b) of Case 1, or after step (g) of Case 2.
  - The partition is conformant: Let $v_i = \alpha$ for all $i \in A$ for $\alpha \in \{0, 1\}$. Such an α exists because the partition is conformant.
    - Case 1: In this case, recall that $S \subseteq A$. Therefore, after steps (a) and (b) both, t_j at all $j \in S$ will be identical, and equal to α.
    - Case 2: This is similar to Case 1. At the end of step (e), for all nodes $i \in A$, $t_i = \alpha$. After step (f), for all nodes $i \in S \cup A$, $t_i = \alpha$. Therefore, after step (g), for all nodes $i \in S$, t_i will remain equal to α.

  Thus, in both scenarios, we found a source component S and α such that for all $i \in S$, $t_i = \alpha$ after step (b) of Case 1 or after step (g) of Case 2.

  Then, consider the remaining steps in the iteration.

  - Case 1: During `Propagate`$(S, \mathcal{V} - F - S)$, each node $k \in \mathcal{V} - F - S$ will receive $f + 1$ copies of α along $f + 1$ disjoint paths, and set $t_k = \alpha$ in step (c). Therefore, each node $k \in \mathcal{V} - F - S$ will update its v_k to be α in step (d).
  - Case 2: After step (h), $t_j = \alpha$ for all $j \in (\mathcal{V} - F - S) \cup S$. Thus, each node $k \in \mathcal{V} - F - (A \cap S)$ will update v_k to be α. Recall that each node $k \in A \cap S$ does not modify its v_k, which is already equal to α.

  Thus, in both cases, at the end of STEP 1 of the INNER loop, for all $k \in \mathcal{V} - F = \mathcal{V} - F^*$, $v_k = \alpha$.

  Since all nodes in F^* are faulty, agreement has been reached at this point. By Lemma 8, the agreed value is valid as well. Thus, the goal now is to show that the agreement and validity conditions are not violated by actions taken in any future iterations of the INNER loop.

- Phase 3: At the start of Phase 3, for each fault-free node $k \in \mathcal{V} - F^*$, we have $v_k = \alpha \in \{0, 1\}$. Then by Lemma 8, all future iterations of the INNER loop cannot assign any value other than α to any node $k \in \mathcal{V} - F^*$.

After Phase 3 with $F = F^*$, Algorithm BC may perform iterations for other choices of set F. However, due to Lemma 8, the value v_i at each $i \in \mathcal{V} - F^*$ (i.e., all fault-free nodes) continues being equal to α. □

Theorem 5 *Algorithm BC satisfies validity, agreement, and termination properties for Byzantine consensus.*

Proof: The theorem follows from Lemmas 9, 10 and 11. □

7 Generalized Fault Model

In this section, we briefly discuss how to extend the above results to exact consensus under generalized fault model. The generalized fault model [6] is characterized using *fault domain* $\mathcal{F} \subseteq 2^{\mathcal{V}}$ as follows: Nodes in set F may fail during an execution of the algorithm only if there exists set $F^* \in \mathcal{F}$ such that $F \subseteq F^*$. Set F is then said to be a *feasible* fault set.

Definition 7 *Set $F \subseteq \mathcal{V}$ is said to be a feasible fault set, if there exists $F^* \in \mathcal{F}$ such that $F \subseteq F^*$.*

Please refer to our previous work [6] for more discussion on generalized fault model.

For a set of nodes B, define $N^-(B) = \{i \mid (i, j) \in \mathcal{E},\ i \notin B,\ j \in B\}$, the set of incoming neighbors of B.

Definition 8 *Given $\mathcal{F}$, for disjoint sets of nodes A and B, where B is non-empty.*

- *$A \overset{g}{\Rightarrow} B$ iff for every $F^* \in \mathcal{F}$, $N^-(B) \cap A \not\subseteq F^*$, i.e., the set of incoming neighbors of B in A is not a feasible fault set.*
- *$A \overset{g}{\not\Rightarrow} B$ iff $A \overset{g}{\Rightarrow} B$ is* not *true.*

With the replacement of $\overset{e}{\Rightarrow}$ by $\overset{g}{\Rightarrow}$, Theorem 1 and 4 will hold for the generalized fault model.

For the generalized fault model, the definition of propagation from A to B should be modified as follows:

Definition 9 *For any partition A, B, F of $\mathcal{V}$ such that A, B are non-empty and F is a feasible fault set, $A \overset{\mathcal{V}-F}{\rightsquigarrow} B$ if for any feasible fault set $F' \subset \mathcal{V} - F$, for every node $i \in B - F'$, there exists in $G(\mathcal{V}, \mathcal{E})$ a (A, i)-path excluding $F \cup F'$.*

Then the correctness of Algorithm BC can be proved with the following changes to the algorithm:

- Whenever Algorithm BC uses $f + 1$ (S, i)-paths in `Propagate`(S, B), the new algorithm uses all possible (S, i)-paths excluding F.

- Whenever a node i in Algorithm BC compares $f+1$ values received in `Propagate`(S, B), in the new algorithm node i uses all values received along all the paths excluding F to decide how to update value t.

Note that by the condition in Theorem 4, it should be easy to see that the paths used to propagate messages contains at least one fault-free path, i.e., every node on the path is fault-free. Therefore, the new algorithm can be shown to achieve termination, agreement, and validity similarly.

8 Example Networks

In this section, we introduce two different graphs, and use the results in the previous sections to show that exact Byzantine consensus can be reached in these graphs.

8.1 1-Core Network

Definition 10 *A graph $G(\mathcal{V}, \mathcal{E})$ consisting of $n > 3f$ nodes is said to be a* 1-core network *if the following two properties are satisfied:*

- *It includes a clique formed by nodes in $K \subseteq \mathcal{V}$, such that $|K| = 3f+1$, as a subgraph. That is, $\forall i, j \in K, i \neq j, (i,j) \in \mathcal{E}$.*
- *Each node $i \notin K$ has incoming links from arbitrary $2f+1$ nodes in K. That is, for each $v \in \mathcal{V} - K$, there exists $K_v \subseteq K$ such that $|K_v| = 2f+1$, and $\forall u \in K_v, (u,v) \in \mathcal{E}$.*

It is easy to show that a core network satisfies the condition in Theorem 1.

There is a simple consensus algorithm for the 1-core network: first solve consensus in the $(3f+1)$-node clique using any existing Byzantine consensus algorithm for cliques; then, all the nodes in the clique transmit their decision value on all the outgoing links to the nodes outside the clique; every node outside the clique decides on the majority of $2f+1$ values received from the nodes in the clique.

8.2 2-Core Network

Definition 11 *A graph $G(\mathcal{V}, \mathcal{E})$ consisting of $n = 6f+2$ nodes, where f is a positive non-zero even integer, is said to be a* 2-core network *if all the following properties are satisfied:*

- *It includes two disjoint cliques, each consisting of $3f+1$ nodes. Suppose that the nodes in the two cliques are specified by sets K_1, K_2, respectively, where $K_1 = \{u_1, u_2, \cdots, u_{3f+1}\} \subset \mathcal{V}$, and $K_2 = \mathcal{V} - K_1 = \{w_1, w_2, \cdots, w_{3f+1}\}$. Thus, $(u_i, u_j) \in \mathcal{E}$ and $(w_i, w_j) \in \mathcal{E}$, for $1 \leq i, j \leq 3f+1, i \neq j$.*
- *$(u_i, w_i) \in \mathcal{E}$, for $1 \leq i \leq \frac{3f}{2}$ and $i = 3f+1$.*
- *$(w_i, u_i) \in \mathcal{E}$, for $\frac{3f}{2} + 1 \leq i \leq 3f$ and $i = 3f+1$.*

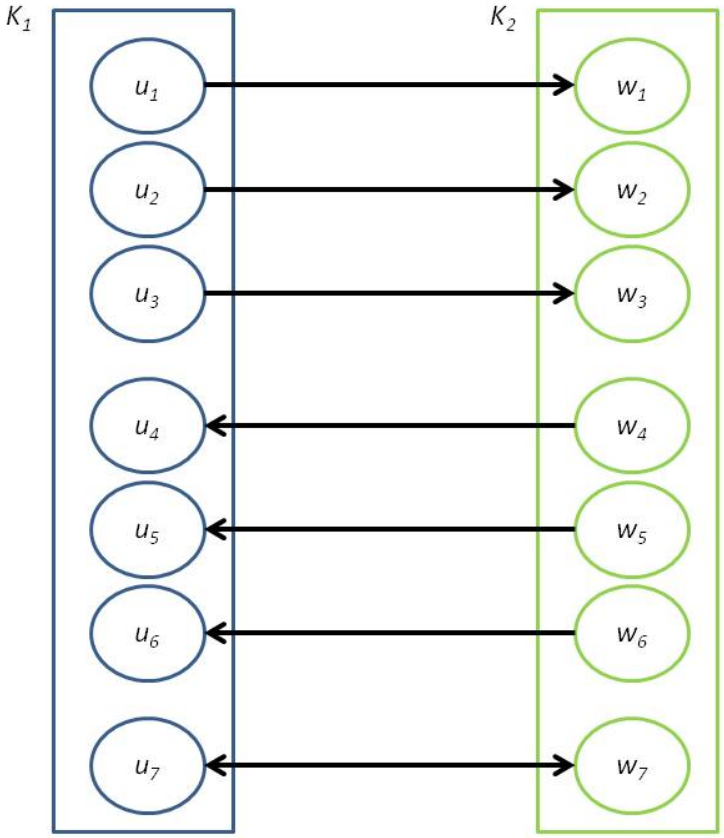

Figure 2: A 2-core network for $f = 2$. For simplicity, the edges in each core, K_1 and K_2, are not presented in this figure. Note that each core is a clique.

Figure 2 illustrates the 2-core network for $f = 2$. We will show that the 2-core network satisfies the condition in Theorem 2. We first prove the following lemma.

Lemma 12 *Let A, B, C, F be disjoint subsets of $\mathcal{V}$ such that $|F| \leq f$ and A, B, C are non-empty. Suppose that $A \overset{\mathcal{V}-F}{\rightsquigarrow} B$ and $A \cup B \overset{\mathcal{V}-F}{\rightsquigarrow} C$. Then, $A \overset{\mathcal{V}-F}{\rightsquigarrow} B \cup C$.*

Proof: The proof is by contradiction. Suppose that

- $A \overset{\mathcal{V}-F}{\rightsquigarrow} B$,
- $A \cup B \overset{\mathcal{V}-F}{\rightsquigarrow} C$, and
- $A \overset{\mathcal{V}-F}{\not\rightsquigarrow} B \cup C$.

The first condition above implies that $|A| \geq f + 1$. By Definition 2 and Menger's Theorem [8], the third condition implies that there exists a node $v \in B \cup C$ and a set of nodes $P \subseteq \mathcal{V} - F - \{v\}$ such that $|P| \leq f$ and all (A, v)-paths excluding F contain at least one node in P. In other words, there is no (A, v)-path excluding $F \cup P$. Observe that, because $A \overset{\mathcal{V}-F}{\rightsquigarrow} B$, v cannot be in B; therefore v must belong to set C.

Let us define the sets X and Y as follows:

- Node $x \in X$ if and only if $x \in \mathcal{V} - F - P$ and there exists an (A, x)-path excluding $F \cup P$. It is possible that $P \cap A \neq \Phi$; thus, the (A, x)-path cannot contain any nodes in $P \cap A$.
- Node $y \in Y$ if and only if $y \in \mathcal{V} - F - P$ and there exists an (y, v)-path excluding $F \cup P$.

By the definition of X and Y, it follows that for any $x \in X, y \in Y$, there cannot be any (x, y)-path excluding $F \cup P$. Also, since $A \overset{\mathcal{V}-F}{\rightsquigarrow} B$, for each $b \in B - P$, there must exist an (A, b)-path excluding $F \cup P$; thus, $B - P \subseteq X$, and $B \subseteq X \cup P$. Similarly, $A \subseteq X \cup P$, and therefore, $A \cup B \subseteq X \cup P$.

By definition of X, there are no $(X \cup P, v)$-paths excluding $F \cup P$. Therefore, because $A \cup B \subseteq X \cup P$, there are no $(A \cup B, v)$-paths excluding $F \cup P$. Therefore, since $v \in C$, $A \cup B \overset{\mathcal{V}-F}{\not\rightsquigarrow} C$. This is a contradiction to the second condition above. □

Lemma 13 *Suppose that $G(\mathcal{V}, \mathcal{E})$ is a 2-core network. Then G satisfies the condition in Theorem 2.*

Proof: Consider a partition A, B, F of $\mathcal{V}$, where A and B are both non-empty, and $|F| \leq f$. Recall from Definition 11 that K_1, K_2 also form a partition of $\mathcal{V}$.

Define $A_1 = A \cap K_1, A_2 = A \cap K_2, B_1 = B \cap K_1, B_2 = B \cap K_2, F_1 = F \cap K_1$ and $F_2 = F \cap K_2$.

Define $\mathcal{E}'$ to be the set of directed links from the nodes in K_1 to the nodes in K_2, or vice-versa. Thus, there are $\frac{3f}{2} + 1$ directed links in $\mathcal{E}'$ from the nodes in K_1 to the nodes in K_2, and the same number of links from the nodes in K_2 to the nodes in K_1. Each pair of links in $\mathcal{E}'$, with the exception of the link pair between a_{3f+1} and b_{3f+1}, is node disjoint. Since $|F| \leq f$, it should be easy to see that, at least one of the two conditions below is true:

(a) There are at least $f+1$ directed links from the nodes in $K_1 - F$ to the nodes in $K_2 - F$.

(b) There are at least $f+1$ directed links from the nodes in $K_2 - F$ to nodes the in $K_1 - F$.

Without loss of generality, suppose that condition (a) is true. Therefore, since $|K_1 - F| \geq 2f+1$ and the nodes in $K_2 - F$ form a clique, it follows that $K_1 - F \overset{\mathcal{V}-F}{\rightsquigarrow} K_2 - F$. Then, because $K_1 - F = A_1 \cup B_1$ and $K_2 - F = A_2 \cup B_2$, we have

$$A_1 \cup B_1 \overset{\mathcal{V}-F}{\rightsquigarrow} A_2 \cup B_2. \tag{4}$$

$|K_1 - F| \geq 2f+1$ also implies that either $|A_1| \geq f+1$ or $|B_1| \geq f+1$. Without loss of generality, suppose that $|A_1| \geq f+1$. Then, since the nodes in $A_1 \cup B_1$ form a clique, it follows that $A_1 \overset{\mathcal{V}-F_1-K_2}{\rightsquigarrow} B_1$ (recall that $\mathcal{V} - F_1 - K_2 = A_1 \cup B_1$). Since $\mathcal{V} - F_1 - K_2 \subset \mathcal{V} - F$, we have

$$A_1 \overset{\mathcal{V}-F}{\rightsquigarrow} B_1 \tag{5}$$

(4) and (5), along with Lemma 12 imply that $A_1 \overset{\mathcal{V}-F}{\rightsquigarrow} B_1 \cup A_2 \cup B_2$. Therefore, $A_1 \overset{\mathcal{V}-F}{\rightsquigarrow} B_1 \cup B_2$, and $A_1 \cup A_2 \overset{\mathcal{V}-F}{\rightsquigarrow} B_1 \cup B_2$. Since $A = A_1 \cup A_2$ and $B = B_1 \cup B_2$, $A \overset{\mathcal{V}-F}{\rightsquigarrow} B$. □

Interestingly, the 2-core network satisfies the necessary condition despite the fact that $2f+1$ links are not available in either direction between the nodes in K_1 and K_2.

9 Conclusion

This paper presents tight necessary and sufficient conditions for achieving Byzantine consensus in synchronous networks that can be represented as directed graphs. We provide a constructive proof of sufficiency by presenting a new Byzantine consensus algorithm for directed graphs. As briefly stated in Section 7, the necessary condition in Theorem 4 and Algorithm BC can also be applied with the generalized fault model in [6]. In Section 8, we also introduce two families of graphs that satisfy the necessary and sufficient condition in Theorem 2.

References

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

[2] M. J. Fischer, N. A. Lynch, and M. Merritt. Easy impossibility proofs for distributed consensus problems. In *Proceedings of the fourth annual ACM symposium on Principles of distributed computing*, PODC '85, pages 59–70, New York, NY, USA, 1985. ACM.

[3] G. Liang and N. Vaidya. Capacity of byzantine agreement with finite link capacity. In *INFOCOM, 2011 Proceedings IEEE*, pages 739 –747, april 2011.

[4] G. Liang and N. H. Vaidya. Capacity of byzantine agreement: Complete characterization of the four node network. Technical report, CSL, UIUC, 2010.

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

[6] L. Tseng and N. H. Vaidya. Iterative approximate byzantine consensus under a generalized fault model. Technical report, CSL, UIUC, 2012.

[7] N. H. Vaidya, L. Tseng, and G. Liang. Iterative approximate byzantine consensus in arbitrary directed graphs. In *Proceedings of the thirty-first annual ACM symposium on Principles of distributed computing*, PODC '12. ACM, 2012.

[8] D. B. West. *Introduction To Graph Theory*. Prentice Hall, 2001.

Appendix

A Proof of Theorem 1

We first describe the intuition behind the proof, followed by a formal proof. Intuitively, if the graph does not satisfy the condition in Theorem 1, then the faulty nodes can force the fault-free nodes to disagree with each other, as follows. Suppose that there exists partition L, R, C, F where L, R are non-empty and $|F| \leq f$ such that $C \cup R \overset{g}{\not\Rightarrow} L$, and $L \cup C \overset{g}{\not\Rightarrow} R$. Now, suppose that all the nodes in L have input m, and all the nodes in $R \cup C$ have input M, where $m \neq M$.

Suppose that the nodes in F are faulty. Then the the faulty nodes can behave to nodes in L as if nodes in $R \cup C \cup F$ have input m, while behaving to nodes in R as if nodes in $L \cup C \cup F$ have input M. Since the graph does not satisfy the condition in Theorem 1, nodes in L cannot distinguish between the following two scenarios, where N_L denotes the set of incoming neighbors of L in $C \cup R$:

- All the nodes in N_L are faulty, rest of the nodes are fault-free, and all the fault-free nodes have input m.
- All the nodes in F are faulty, rest of the nodes are fault-free, and fault-free nodes have input m or M.

In the first scenario, for validity, the output at nodes in L must be m. Therefore, in the second scenario as well, the output at the nodes in L must be m.

Similarly, nodes in R cannot distinguish between the following two scenarios, where N_R denotes the set of incoming neighbors of R in $C \cup L$:

- All the nodes in N_R are faulty, rest of the nodes are fault-free, and all the fault-free nodes have input M.
- All the nodes in F are faulty, rest of the nodes are fault-free, and fault-free nodes have input m or M.

In the first scenario, for validity, the output at nodes in R must be M. Therefore, in the second scenario as well, the output at the nodes in R must be M.

Thus, in the case when the nodes in F are faulty, nodes in L and R can be forced to decide on distinct values, violating the agreement requirement.

Now we present a formal proof of Theorem 1.

Proof: The proof is by contradiction. Suppose that a correct Byzantine consensus algorithm (say ALGO) exists, and there exists a partition such that $C \cup R \overset{g}{\not\Rightarrow} L$ and $L \cup C \overset{g}{\not\Rightarrow} R$. Thus, L has at most f incoming neighbors in $R \cup C$, and R has at most f incoming neighbors in $L \cup C$. We further assume that the nodes in F (if F is non-empty) are all faulty, and the remaining nodes (in L, C, R) are all fault-free.

Let us assume that the behavior of each node $i \in \mathcal{V}$ when using ALGO can be modeled by a state machine. We construct an augmented network $\mathcal{N}$ with the following properties:[13]

[13] We use italic letters for entities in $G(\mathcal{V}, \mathcal{E})$, and non-italic letters for entities in $\mathcal{N}$.

- For each node r in R, there are two copies in $\mathcal{N}$. The two copies are named r and r2. The two nodes r and r2 in $\mathcal{N}$ are copies of r in the sense that the corresponding two nodes have identical state machine as r.
- For each node $l \in L$, there are two copies in $\mathcal{N}$. The two copies are named l and l1.
- For each node $k \in F$, there are two copies in $\mathcal{N}$. The two copies are named k1 and k2.
- For each node $c \in C$, there are three copies in $\mathcal{N}$. The three copies are named c, c1 and c2.

The communication links in $\mathcal{N}$ are derived using the communication graph $G(\mathcal{V}, \mathcal{E})$. In particular, if node i has a link to node j in G, then a copy of node j in $\mathcal{N}$ will have a link from one copy of node i in $\mathcal{N}$.

On the other hand, if link $(i, j) \in \mathcal{E}$ then one copy of node i in $\mathcal{N}$ may have links to multiple copies of node j in $\mathcal{N}$. This should be viewed as a "broadcast" operation that is being simulated unbeknownst to the state machines for the corresponding nodes in $\mathcal{N}$. The same technique of broadcast operation has also been used in [3, 4]. Exactly which copy of node i has link to a copy of node j is represented with the edges shown in Figure 3, as described next.

- Vertices in Figure 3 represent sets of vertices in $\mathcal{N}$.
  Vertex R represents a set containing node r in $\mathcal{N}$ corresponding to each node $r \in R$.
  Vertex R2 represents a set containing node r2 in $\mathcal{N}$ corresponding to each node $r \in R$.
  Vertex F1 represents a set containing node k1 in $\mathcal{N}$ corresponding to each node $k \in F$.
  Vertex F2 represents a set containing node k2 in $\mathcal{N}$ corresponding to each node $k \in F$.
  Vertices C, C1, C2, L, and L1 analogously represent copies of appropriate nodes in G.
- The directed edge from vertex R to vertex F1 in Figure 3 indicates that, **if** for $r \in R$ and $k \in F$, link $(r, k) \in \mathcal{E}$, then link (r,k1) is in $\mathcal{N}$. Similarly, the directed edge from vertex F2 to vertex L in Figure 3 indicates that, **if** for $k \in F$ and $l \in L$, link $(k, l) \in \mathcal{E}$, then link (k2,l) is in $\mathcal{N}$. Other solid edges in Figure 3 represent other communication links in $\mathcal{N}$ similarly.

  The dotted arrows are also communication links in $\mathcal{N}$, but we use dots to emphasize that the links are broadcast links in the sense discussed above. There are four such "broadcast edges" in the figure. The broadcast edge from L to R and R1 implies that **if** for $l \in L$ and $r \in R$, link $(l, r) \in \mathcal{E}$, then messages from node l in $\mathcal{N}$ being sent to the state machine r are sent to r and r1 both in $\mathcal{N}$.
- Five of the edges do not terminate at any vertex in Figure 3 (one such edge at each of the vertices C1, L1, R2, C2, and C). This signifies that the corresponding transmissions are discarded silently without the knowledge of the sender. For instance, transmissions from L1 to R are discarded. More specifically, for $l \in L$ and $r \in R$, if there is a link $(r, l) \in \mathcal{E}$, then transmissions by node l1 (in $\mathcal{N}$) intended for state machine r are silently discarded *without the knowledge of node* l1[14].

Each node in $G(\mathcal{V}, \mathcal{E})$ has an input as discussed previously. An input is also available to each node in $\mathcal{N}$. In our discussion, we will assume that the fault-free nodes represented by any single

[14]These edges may alternatively be modeled using additional copies of state machines in L, R, and F that do not have outgoing edges. For instance, in Figure 3, we can replace L1's outgoing neighbor set R by set R3, which contains node r3 (in $\mathcal{N}$) corresponding to each $r \in R$. Now, in our example above, instead of the message from $l1$ to r being silently discarded, the message will be sent to $r3$.

vertex in Figure 3 all have the identical input. Specifically, the input at the nodes represented by vertex L is m, and the input is shown in a rectangle next to vertex L in Figure 3. Similarly, input at nodes represented by the other vertices is also shown in the figure.

Let us define:

$$\begin{aligned} N_L &= \text{set of incoming neighbors of } L \text{ in } R \cup C \\ N_R &= \text{set of incoming neighbors of } R \text{ in } L \cup C \end{aligned}$$

By assumption, $|N_L| \leq f$ and $|N_R| \leq f$.

We now show how the behavior of a certain subset of vertices in $\mathcal{N}$ is identical to the behavior of corresponding nodes in the original network G. In each case, we consider partition L, R, C, F of $\mathcal{V}$.

- *Case 1*: Nodes in N_R are faulty, and the other nodes in $\mathcal{V}$ are fault-free: We can model the fault-free nodes by the corresponding nodes in L1, R, C1 and F1 in $\mathcal{N}$. An instance of the behavior of faulty nodes in N_R is modeled by corresponding nodes in L, L1, C and C1. Since the fault-free nodes in L, C, R, F must agree on value M in G, the nodes represented by R in $\mathcal{N}$ will also terminate with output M.

- *Case 2*: Nodes in N_L are faulty, and the other nodes in $\mathcal{V}$ are fault-free: We can model the fault-free nodes by the corresponding nodes in L, R2, C2 and F2 in $\mathcal{N}$. An instance of the behavior of faulty nodes in N_L is modeled by corresponding nodes in R, R2, C and C2. Since the fault-free nodes in L, C, R, F must agree on value m in G, the nodes represented by L in $\mathcal{N}$ will also terminate with output m.

- *Case 3:* Nodes in set F are faulty, and the other nodes in $\mathcal{V}$ are fault-free: We can model the fault-free nodes in $\mathcal{V} - F = L \cup C \cup R$ by the corresponding nodes in L, C, R in $\mathcal{N}$. An instance of the behavior of faulty nodes in F is modeled by the behavior of F1 and F2. Since the fault-free nodes in L, C, R must agree on a common value in G, nodes represented by L and R will also terminate with agreement on an identical value. However, this contradicts with Cases 1 and 2, which conclude that nodes in R and L output M and m, respectively.

The above contradiction proves that the condition in Theorem 1 is necessary. □

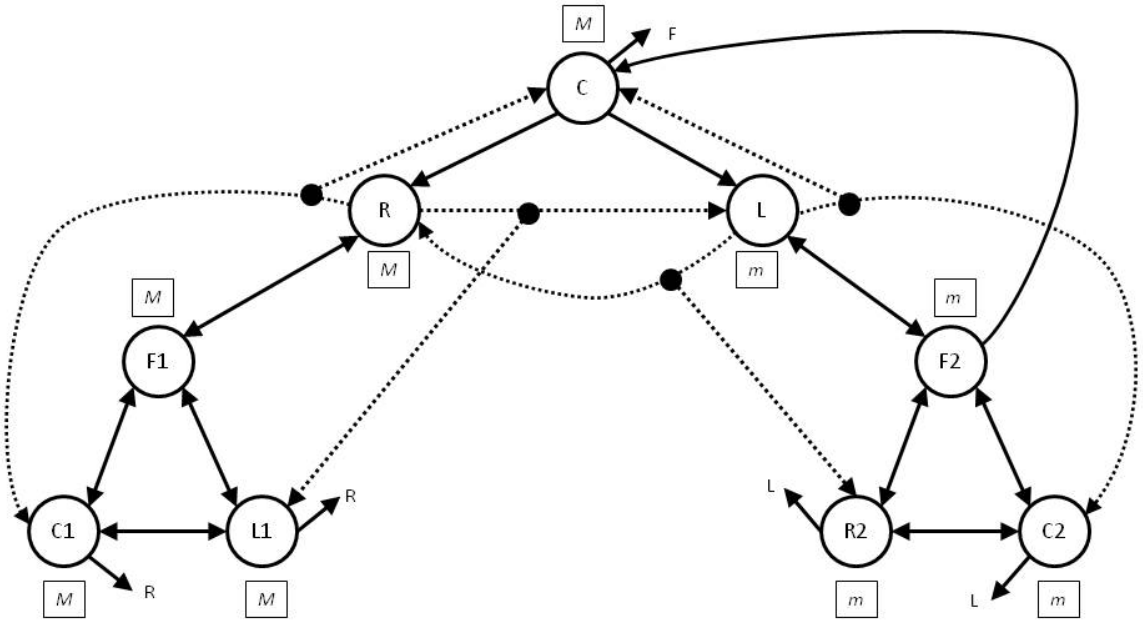

Figure 3: Augmented Network $\mathcal{N}$