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
Phone: +1 217-244-6024, +1 217-265-5414

February 10, 2013†

Abstract

Consider a synchronous point-to-point network of n nodes connected by *directed* links, wherein each node has a binary input. This paper proves a *tight* necessary and sufficient condition for achieving Byzantine consensus among these nodes in the presence of up to f Byzantine faults. We derive two forms of the necessary condition. We provide a constructive proof of sufficiency by presenting a Byzantine consensus algorithm for directed graphs that satisfy the necessary condition.

Prior work has developed analogous necessary and sufficient conditions for *undirected* graphs. It is known that, for undirected graphs, the following two conditions are together necessary and sufficient [4, 2]: (i) $n \geq 3f+1$, and (ii) network connectivity greater than $2f$. However, these conditions are not adequate to completely characterize 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.

†This is a revised version of *Exact Byzantine Consensus in Directed Graphs* (Oct. 2012) of the previous report and version 3 of that report (on arxiv.org/abs/1208.5075).

1 Introduction

In this work, we explore algorithms for achieving Byzantine consensus [3] in a synchronous point-to-point network of n nodes, in presence of up to f Byzantine faulty nodes. The network is modeled as a *directed* graph, i.e., the communication links between neighboring nodes are not necessarily bi-directional. Our work is motivated by the presence of directed links in wireless networks. However, we believe that the results here are of independent interest as well.

The Byzantine consensus problem [3] considers n nodes, of which at most f nodes may be faulty. The faulty nodes may deviate from the algorithm in arbitrary fashion. Each node has an *input* in $\{0, 1\}$. A Byzantine consensus algorithm is *correct* if it satisfies the following three properties:

- **Agreement**: the output (i.e., decision) at all the fault-free nodes must be identical.
- **Validity**: the output of every fault-free node equals the input of a fault-free node.
- **Termination**: every fault-free node eventually decides on an output.

In networks with undirected links (i.e., in undirected graphs), it is well-known that the following two conditions together are both necessary and sufficient for the existence of Byzantine consensus algorithms [4, 2]: (i) $n \geq 3f + 1$, and (ii) node connectivity greater than $2f$. The first condition, that is, $n \geq 3f+1$, is necessary for directed graphs as well. Tight necessary and sufficient conditions for Byzantine consensus in *directed* graphs have not been developed previously. In this paper, we develop such conditions. We provide a constructive proof of sufficiency by presenting a Byzantine consensus algorithm for directed graphs satisfying the necessary condition.

Network 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.

Communication Model: All the communication links 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 thus, each message sent on link (i, j) is delivered to node j within a bounded interval of time.

2 Terminology

We now describe terminologies that are used frequently in our presentation. 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:
  - For a directed path from node i to node j, node i is said to be the "source node" for the path.
  - 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 excluding X" refers to d pairwise disjoint (i, j)-paths that do not contain any node from set X.
- Every node i trivially has a path to itself. That is, for all $i \in \mathcal{V}$, an (i, i)-path excluding $\mathcal{V} - \{i\}$ exists.
- 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 source 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 excluding X" refers to d pairwise disjoint (S, j)-paths that do not contain any node from set X.

3 Necessary Condition

For a correct Byzantine consensus algorithm to exist in presence of up to f faulty nodes, the network graph $G(\mathcal{V}, \mathcal{E})$ must satisfy the necessary condition developed in this section. In Sections 3.1 and 3.2, we present two equivalent forms of the necessary condition.

3.1 Necessary Condition: First Form

We first define relations $\rightarrow$ and $\not\rightarrow$ that are used frequently in the paper. These relations are defined for disjoint sets. Two sets are disjoint if their intersection is empty. For convenience of presentation, we adopt the convention that sets A and B are disjoint if either one of them is empty. More than two sets are disjoint if they are pairwise disjoint.

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

- *$A \rightarrow B$ iff set A contains at least $f+1$ distinct incoming neighbors of B.*

  That is, $| \{i \mid (i,j) \in \mathcal{E},\ i \in A,\ j \in B\} | > f$.

- *$A \not\rightarrow B$ iff $A \rightarrow B$ is* not *true.*

The theorem below states the first form of our necessary condition.

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

Proof Sketch: The complete proof is presented in Appendix A using a state-machine approach [4]. Here we sketch an intuitive proof. The proof is by contradiction.

Suppose that there exists a partition L, C, R, F where L, R are non-empty and $|F| \leq f$ such that $C \cup R \not\rightarrow L$, and $L \cup C \not\rightarrow R$. Assume that the nodes in F are faulty, and the nodes in sets L, C, R are fault-free. Note that fault-free nodes are not aware of the identity of the faulty nodes.

Consider the case when 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 (if non-empty) 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. This behavior by nodes in F is possible, since the nodes in F are all assumed to be faulty here.

Consider nodes in L. Let N_L denote the set of incoming neighbors of L in $R \cup C$. Since $R \cup C \not\rightarrow L$, $|N_L| \leq f$. Therefore, nodes in L cannot distinguish between the following two scenarios: (i) all the nodes in N_L (if non-empty) are faulty, rest of the nodes are fault-free, and all the fault-free nodes have input m, and (ii) all the nodes in F (if non-empty) are faulty, rest of the nodes are fault-free, and fault-free nodes have input either 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. We can similarly show that the output at the nodes in R must be M. Thus, if the condition in Theorem 1 is not satisfied, nodes in L and R can be forced to decide on distinct values, violating the agreement property. $\square$

3.2 Necessary Condition: Second Form

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

We will denote the fact that set A propagates in $\mathcal{V} - F$ to set B by the notation $A \overset{\mathcal{V}-F}{\rightsquigarrow} B$. When it is not true that $A \overset{\mathcal{V}-F}{\rightsquigarrow} B$, we will denote that fact by $A \overset{\mathcal{V}-F}{\not\rightsquigarrow} B$.

We now state the second form of the necessary condition. This form provides insight on how to "propagate" values from one set of nodes to the other set.

[1]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) $X_i \cap X_j = \emptyset$ if $i \neq j$.

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: Appendix B proves Theorem 2 by showing that the condition stated in Theorem 2 is implied by the condition stated in Theorem 1. $\square$

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

Proof: From the prior work [3, 4], the condition $n \geq 3f + 1$ is known to be necessary. As elaborated in Appendix C, $n \geq 3f + 1$ can be shown to be necessary using Theorem 1 as well.

Now, for $f > 0$, we show that it is necessary for each node to have at least $2f + 1$ incoming neighbors. The proof is by contradiction. Suppose that for some node $i \in \mathcal{V}$, the number of incoming neighbors 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, since node i has at most $2f$ incoming neighbors.

Define $C = \emptyset$ and $R = \{i\}$. Thus, L, C, R, F for a partition of $\mathcal{V}$. Then, since $f > 0$ and $|R \cup C| = 1$, it follows that $R \cup C \not\rightarrow L$. Also, since L contains at most f incoming neighbors of node i, $C = \emptyset$, and set R contains only node i, $L \cup C \not\rightarrow R$. The above two conditions violate the necessary condition stated in Theorem 1. $\square$

3.3 The Equivalence of Two Forms of the Condition

The proof of Theorem 2 shows that the condition in Theorem 1 implies the condition in Theorem 2. That fact, and Lemma 1 below, together prove that the two forms of the condition are equivalent.

Lemma 1 *The condition stated in Theorem 2 (i.e., the second form of the necessary condition) implies the condition stated in Theorem 1 (i.e., the first form of the necessary condition).*

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

Suppose that the condition in Theorem 1 is violated. Then there exists a partition L, C, R, F of $\mathcal{V}$ such that L, R are both non-empty, $|F| \leq f$, $L \cup C \not\rightarrow R$ and $R \cup C \not\rightarrow L$.

Since $L \cup C \not\rightarrow R$, 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 $L \subseteq L \cup C$, Menger's theorem [5] implies that there are at most f disjoint (L, r)-paths excluding F. Thus, because $r \in R \cup C$, $L \overset{\mathcal{V}-F}{\not\rightsquigarrow} R \cup C$.

Similarly, since $R \cup C \not\rightarrow L$, 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 [5] then implies that there are at most f disjoint $(R \cup C, l)$-paths excluding F. Thus, $R \cup C \overset{\mathcal{V}-F}{\not\rightsquigarrow} L$.

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 $A \overset{\mathcal{V}-F}{\not\rightsquigarrow} B$ and $B \overset{\mathcal{V}-F}{\not\rightsquigarrow} A$, violating the condition stated in Theorem 2. $\square$

4 Sufficiency: Algorithm BC and Correctness Proof

This section proves that the necessary condition stated in Theorems 1 and 2 is also sufficient by proving the correctness of Algorithm BC below in graphs that satisfy the necessary condition. Hereafter, assume that graph $G(\mathcal{V}, \mathcal{E})$ satisfies the condition stated in Theorems 1 and 2, even if this is not stated explicitly again (recall that the conditions in Theorems 1 and 2 are equivalent).

When $f = 0$, all the nodes are fault-free, and as shown in Appendix E, the proof of sufficiency is trivial. In the rest of our discussion below, we will assume that $f > 0$.

The proposed Algorithm BC uses Definition 3 below. The INNER loop of Algorithm BC uses procedures `Propagate` and `Equality`. These procedures make use of state variables t and v maintained by the nodes. We discuss the node state in Section 4.1, and the two procedures in Sections 4.2 and 4.3.

Definition 3 *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.*

Algorithm BC

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

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

4.1 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 state 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 binary 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 it equals some fault-free node's input. Initial value v_i at a fault-free node i is valid because it equals its own input. Lemma 2 proved later in Section 4.5 implies that v_i at a fault-free node i always remains valid throughout the execution of Algorithm BC.

- 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, in steps (d) and (i) of Algorithm BC.

4.2 Procedure `Propagate`(P, D)

`Propagate`(P, D) assumes that $P \subseteq \mathcal{V} - F$, $D \subseteq \mathcal{V} - F$, $P \cap D = \emptyset$ and $P \overset{\mathcal{V}-F}{\rightsquigarrow} D$. Recall that set F is the set chosen in each OUTER loop as specified by Algorithm BC.

`Propagate`(P, D)

---

(1) Since $P \overset{\mathcal{V}-F}{\rightsquigarrow} D$, for each $i \in D$, there exist $f + 1$ disjoint (P, i)-paths that exclude F. The source node of each of these paths is in P. On each of these $f + 1$ disjoint paths, the source node for that path, say s, sends t_s to node i. 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 $i \in D$ 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 any node $j \notin D$, t_j is not modified during `Propagate`(P, D). Also, for any node $k \in \mathcal{V}$, v_k is not modified during `Propagate`(P, D).

4.3 Procedure `Equality`(D)

`Equality`(D) assumes that $D \subseteq \mathcal{V} - F$, $D \neq \emptyset$, and for each pair of nodes $i, j \in D$, an (i, j)-path excluding F exists, i.e., D is strongly connected in G_{-F} (G_{-F} is defined in Definition 3).

`Equality`(D)

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

(2) Each node $j \in D$ thus receives messages from all nodes in D. Node j checks whether values received from all the nodes in D 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 $k \notin D$, t_k is not modified in `Equality`(D). Also, for any node $k \in \mathcal{V}$, v_k is not modified in `Equality`(D).

4.4 INNER Loop of Algorithm BC for $f > 0$

Assume that $f > 0$. 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 sets we can ensure that $A \overset{\mathcal{V}-F}{\rightsquigarrow} B$. Then, depending on the choice of A, B, F, two cases may occur: (Case 1) $A \overset{\mathcal{V}-F}{\rightsquigarrow} B$ and $B \overset{\mathcal{V}-F}{\not\rightsquigarrow} A$, and (Case 2) $A \overset{\mathcal{V}-F}{\rightsquigarrow} B$ and $B \overset{\mathcal{V}-F}{\rightsquigarrow} A$.

In Case 1 in the INNER loop of Algorithm BC, we need to find a non-empty set $S \subseteq A$ such that $S \overset{\mathcal{V}-F}{\rightsquigarrow} \mathcal{V} - F - S$, and S is strongly connected in G_{-F}. In Case 2, we need to find a non-empty set $S \subseteq A \cup B$ such that $S \overset{\mathcal{V}-F}{\rightsquigarrow} \mathcal{V} - F - S$, S is strongly connected in G_{-F}, and $A \overset{\mathcal{V}-F}{\rightsquigarrow} (S - A)$. Appendix F shows that (i) the required set S exists in both the cases, and (ii) each node in set F has enough incoming neighbors in $\mathcal{V} - F$ to perform step (j) of Algorithm BC with $f > 0$.

4.5 Correctness of Algorithm BC for $f > 0$

In the discussion below, assume that F^* is the actual set of faulty nodes in the network ($0 \leq |F^*| \leq f$). Thus, the set of fault-free nodes is $\mathcal{V} - F^*$. When discussing a certain INNER loop iteration, we sometimes add superscripts *start* and *end* to v_i for node i to indicate whether we are referring to v_i at the start, or at the end, of that INNER loop iteration, respectively.

Lemma 2 *For any given INNER loop iteration, for each fault-free node $j \in \mathcal{V} - F^*$, there exists a fault-free node $s \in \mathcal{V} - F^*$ such that $v_j^{end} = v_s^{start}$.*

Proof: To avoid cluttering the notation, for a set of nodes X, we use the phrase

a fault-free node $j \in X$

as being equivalent to

a fault-free node $j \in X - F^*$

because all the fault-free nodes in any set X must also be in $X - F^*$.

Define set Z as the set of values of v_i at all fault-free $i \in \mathcal{V}$ at the start of the INNER loop iteration under consideration, i.e., $Z = \{v_i^{start} \mid i \in \mathcal{V} - F^* \}$.

We first prove the claim in the lemma for the fault-free nodes in $\in \mathcal{V} - F$, and then for the fault-free nodes in F. Consider the following two cases in the INNER loop iteration.

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

  Observe that, in Case 1, v_i remains unchanged for all fault-free $i \in S$. Thus, $v_i^{end} = v_i^{start}$ for $i \in S$, and hence, the claim of the lemma is trivially true for these nodes. 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), step 2 of `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 $j \in \mathcal{V} - F - S$, step 2 of `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:** $A \overset{\mathcal{V}-F}{\rightsquigarrow} B$ and $B \overset{\mathcal{V}-F}{\rightsquigarrow} A$:

  Observe that, in Case 2, v_j remains unchanged for all fault-free $j \in A \cap S$; thus $v_j^{end} = v_j^{start}$ for these nodes. 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 $\gamma \in Z$. Therefore, at node $m \in S - A$, `Propagate`$(A, S - A)$ will result in t_m being set to a value in $\{\gamma, \perp\} \subseteq Z \cup \{\perp\}$. Now, for $m \in S \cap A$, t_m is not modified in step (f), and therefore, for fault-free $m \in S \cap A$, $t_m \in Z$. 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 remains 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 $k \in F$. Step (j) uses set $N_k \subset \mathcal{V} - F$ such that $|N_k| = f + 1$. 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. It follows that 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 equal to $v_k^{start} \in Z$. $\square$

Lemma 3 *Algorithm BC satisfies the validity property for Byzantine consensus.*

Proof: By the definition of *valid* in Section 4.1, the state v_i of a fault-free node i is *valid* if it equals the input at a fault-free node. For each fault-free $i \in \mathcal{V}$, initially, v_i is valid. Lemma 2 implies that after each INNER loop iteration, v_i remains valid at each fault-free node i. Thus, when Algorithm BC terminates, v_i at each fault-free node i will satisfy the *validity* property for Byzantine consensus, as stated in Section 1. $\square$

Lemma 4 *Algorithm BC satisfies the termination property 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 OUTER loop iterations, and a finite number of INNER loop iterations for each choice of F in the OUTER loop, the number of iterations being a function of graph $G(\mathcal{V}, \mathcal{E})$. Hence, the termination property is satisfied. $\square$

Lemma 5 *Algorithm BC satisfies the agreement property for Byzantine consensus.*

Proof Sketch: The complete proof is in Appendix G. Recall that F^* denotes the actual set of faulty nodes in the network ($0 \le |F^*| \le f$). Since the OUTER loop considers all possible $F \subset \mathcal{V}$ such that $|F| \le f$, eventually, the OUTER loop will be performed with $F = F^*$. We will show that when OUTER loop is performed with $F = F^*$, *agreement* is achieved. After agreement is reached when $F = F^*$, Algorithm BC may perform the OUTER loop with other choices of set F. However, due to Lemma 2, the *agreement* among fault-free nodes is still preserved. (Also, due to Lemma 2, before the OUTER loop with $F = F^*$ is performed, v_i at each fault-free node remains valid.)

Now, consider the OUTER loop with $F = F^*$. We will say that an INNER loop iteration with $F = F^*$ is "deciding" if one of the following conditions is true: (i) in Case 1 of the INNER loop iteration, after step (b) is performed, all the nodes in set S have an identical value for variable t, or (ii) in Case 2 of the INNER loop iteration, after step (g) is performed, all the nodes in set S have an identical value for variable t. As elaborated in Appendix G, when $F = F^*$, at least one of the INNER loop iterations must be a *deciding* iteration. Let us partition the INNER loop iterations when $F = F^*$ into three phases:

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

From the pseudo-code for `Propagate` and `Equality`, observe that when $F = F^*$, all paths used in the INNER loop iterations **exclude** $F = F^*$. That is, all these paths contain only fault-free nodes, since F^* is the actual set of faulty nodes. In each INNER loop iteration in Phase 1, we can show that value v_i for each fault-free node i remains unchanged from previous INNER loop iteration. As elaborated in Appendix G, this ensures that a *deciding* INNER loop iteration is eventually performed when $F = F^*$. In Phase 2, Algorithm BC achieves agreement among fault-free nodes due to the fact that nodes in set S reliably propagate an identical value to all the other nodes. Finally, in Phase 3, due to Lemma 2, agreement achieved in the previous phase is preserved. Therefore, at the end of the OUTER loop with $F = F^*$, agreement is achieved. □

Theorem 3 *Algorithm BC satisfies the agreement, validity, and termination conditions.*

Proof: The theorem follows from Lemmas 3, 4 and 5. □

5 2-Clique Network

In this section, we introduce an example network, named 2-clique network.

Definition 4 *A graph $G(\mathcal{V}, \mathcal{E})$ consisting of $n = 6f + 2$ nodes, where f is a positive even integer, is said to be a* 2-clique 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 and 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 $1 \leq i, j \leq 3f + 1$ and $i \neq j$,*
- *$(u_i, w_i) \in \mathcal{E}$, for $1 \leq i \leq \frac{3f}{2}$ and $i = 3f + 1$, and*
- *$(w_i, u_i) \in \mathcal{E}$, for $\frac{3f}{2} + 1 \leq i \leq 3f$ and $i = 3f + 1$.*

Figure 1 illustrates the 2-clique network for $f = 2$. As proved in Appendix H, the 2-clique network satisfies the condition in Theorem 2, and thus, Algorithm BC achieves consensus in this network. Note that only $\frac{3f}{2} + 1$ links exist from clique K_1 to clique K_2, and vice-versa. Thus, when $f > 0$, reliable communication is *not* possible from a node in clique K_1 to any node in clique K_2, and vice-versa. Yet, Byzantine consensus is possible in the 2-clique network with $f > 0$.

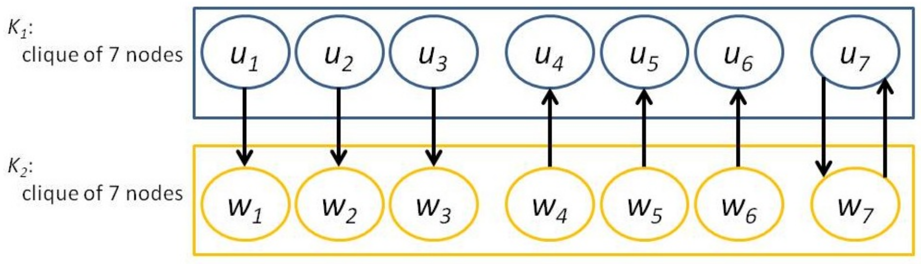

Figure 1: A 2-clique network for $f = 2$. Edges inside cliques K_1 and K_2 are not shown.

6 Conclusion

For nodes with binary inputs, we present a *tight* necessary and sufficient condition for achieving Byzantine consensus in synchronous *directed* networks. We provide a constructive proof of sufficiency by presenting a Byzantine consensus algorithm for graphs satisfying the necessary condition.

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] L. Lamport, R. Shostak, and M. Pease. The Byzantine generals problem. *ACM Trans. on Programming Languages and Systems*, 1982.

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

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

Appendices

A Proof of Theorem 1

In Section 3.1, we presented an intuitive proof of Theorem 1. Here we present a formal proof using the state-machine approach [4].

Proof: The proof is by contradiction. Suppose that a correct Byzantine consensus algorithm, say ALGO, exists in $G(\mathcal{V}, \mathcal{E})$, and there exists a partition F, L, C, R of $\mathcal{V}$ such that $C \cup R \not\rightarrow L$ and $L \cup C \not\rightarrow R$. Thus, L has at most f incoming neighbors in $R \cup C$, and R has at most f incoming neighbors in $L \cup C$. Let us define:

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

Then,

$$|N_L| \leq f \tag{1}$$

$$|N_R| \leq f \tag{2}$$

The behavior of each node $i \in \mathcal{V}$ when using ALGO can be modeled by a state machine that characterizes the behavior of each node $i \in \mathcal{V}$.

We construct a new network called $\mathcal{N}$, as illustrated in Figure 2. In $\mathcal{N}$, there are three copies of each node in C, and two copies of each node in $L \cup R \cup F$. In particular, C0 represents one copy of the nodes in C, C1 represents the second copy of the nodes in C, and C2 represents the third copy of the nodes in C. Similarly, R0 and R2 represent the two copies of the nodes in R, L0 and L1 represent the two copies of the nodes in L, and F1 and F2 represent the two copies of the nodes in F. Even though the figure shows just one vertex for C1, it represents all the nodes in C (each node in C has a counterpart in the nodes represented by C1). Same correspondence holds for other vertices in Figure 2.

The communication links in $\mathcal{N}$ are derived using the communication graph $G(\mathcal{V}, \mathcal{E})$. The figure shows solid edges and dotted edges, and also edges that do not terminate on one end. We describe all three types of edges below.

- *Solid edges:* If a node i has a link to node j in $G(\mathcal{V}, \mathcal{E})$, i.e., $(i, j) \in \mathcal{E}$, then each copy of node j in $\mathcal{N}$ will have a link from **one of the copies** of node i in $\mathcal{N}$. Exactly which copy of node i has link to a copy of node j is represented with the edges shown in Figure 2. For instance, the directed edge from vertex R0 to vertex F1 in Figure 2 indicates that, **if** for $r \in R$ and $k \in F$, link $(r, k) \in \mathcal{E}$, then there is a link in $\mathcal{N}$ from the copy of r in R0 to the copy of k in F1. Similarly, the directed edge from vertex F2 to vertex L0 in Figure 2 indicates that, **if** for $k \in F$ and $l \in L$, link $(k, l) \in \mathcal{E}$, then there is a link from the copy of k in F2 to the copy of l in L0. Other solid edges in Figure 2 represent other communication links in $\mathcal{N}$ similarly.
- *Dotted edges:* Dotted edges are defined similar to the solid edges, with the difference being that the dotted edges emulate a broadcast operation. Specifically, in certain cases, if link $(i, j) \in \mathcal{E}$, then one copy of node i in $\mathcal{N}$ may have links to **two** copies of node j in $\mathcal{N}$, with

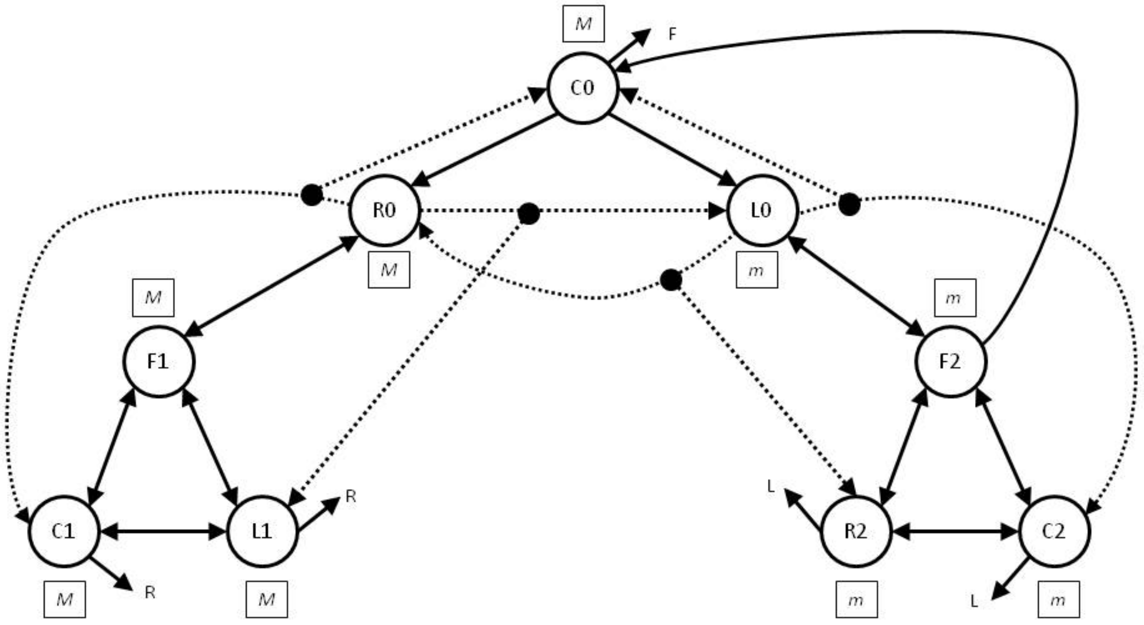

Figure 2: Network $\mathcal{N}$

both copies of node j receiving identical messages from the same copy of node i. This should be viewed as a "broadcast" operation that is being emulated *unbeknownst* to the nodes in $\mathcal{N}$. There are four such "broadcast edges" in the figure, shown as dotted edges. The broadcast edge from L0 to R0 and R2 indicates that **if** for $l \in L$ and $r \in R$, link $(l, r) \in \mathcal{E}$, then messages from the copy of node l in L0 are broadcast to the copies of node r in R0 and R2 both. Similarly, the broadcast edge from R0 to C0 and C1 indicates that **if** for $r \in R$ and $c \in C$, link $(r, c) \in \mathcal{E}$, then messages from the copy of node r in R0 are broadcast to the copies of node c in C0 and C1 both. There is also a broadcast edge from L0 to C0 and C2, and another broadcast edge from R0 to L0 and L1.

- *"Hanging" edges:* Five of the edges in Figure 2 do not terminate at any vertex. One such edge originates at each of the vertices C1, L1, R2, C2, and C0, and each such edge is labeled as R, L or F, as explained next. A *hanging* edge signifies that the corresponding transmissions are discarded silently without the knowledge of the sender. In particular, the *hanging* edge originating at L1 with label R indicates the following: if for $l \in L$ and $r \in R$, $(r, l) \in \mathcal{E}$, then transmissions by the copy of node l in L1 to node r are silently discarded *without the knowledge* of the copy of node l in L1. Similarly, the *hanging* edge originating at C0 with label F indicates the following: if for $c \in C$ and $k \in F$, $(c, k) \in \mathcal{E}$, then transmissions by the copy of node c in C0 to node k are silently discarded *without the knowledge* of the copy of node c in C0.

  It is possible to avoid using such "hanging" edges by introducing additional vertices in $\mathcal{N}$. We choose the above approach to make the representation more compact.

Whenever $(i, j) \in \mathcal{E}$, in network $\mathcal{N}$, each copy of node i has an incoming edge from **one copy of** node j, as discussed above. The broadcast and hanging edges defined above are consistent with our *communication model* in Section 1. As noted there, each node, when sending a message, simply puts the message in the send buffer. Thus, it is possible for us to emulate hanging edges by discarding messages from the buffer, or broadcast edges by replicating the messages into two send buffers. (Nodes do not read messages in send buffers.)

Now, let us assign input of m or M, where $m \neq M$, to each of the nodes in $\mathcal{N}$. The inputs are shown next to the vertices in small rectangles Figure 2. For instance, M next to vertex C1 means that each node represented by C1 has input M (recall that C1 represents one copy of each node in C). Similarly, m next to vertex L0 means that each node represented by L0 has input m.

Consider three sub-networks of $\mathcal{N}$. In each case, we will identify a set of $\geq n - f$ nodes in $\mathcal{N}$ as being fault-free. The behavior of the faulty nodes is modeled by the rest of $\mathcal{N}$.

- **Sub-network I** consists of nodes in R0, C1, L1 and F1. Let the incoming neighbors of nodes in R0 that are $\underline{\text{n}}$ot in R0 or F1 be faulty, with the rest of the nodes being fault-free. The behavior of the faulty nodes (i.e., incoming neighbors of R0 that are not in R0 or F1) is modeled by the behavior of the senders for the incoming links at the nodes in R0. Recall from (2) that $|N_R| \leq f$. Since ALGO is correct in $G(\mathcal{V}, \mathcal{E})$, it must be correct in sub-network I. Therefore, the nodes in R0 must agree on M, because all the fault-free nodes in sub-network I have input M.

- **Sub-network II** consists of nodes in L0, C2, R2 and F2. Let the incoming neighbors of nodes in L0 that are $\underline{\text{n}}$ot in L0 or F2 be faulty, with the rest of the nodes being fault-free. The behavior of the faulty nodes (i.e., incoming neighbors of L0 that are not in L0 or F2) is modeled by the behavior of the senders for the incoming links at the nodes in L0. Recall from

(1) that $|N_L| \le f$. Since ALGO is correct in $G(\mathcal{V}, \mathcal{E})$, it must be correct in sub-network II. Therefore, the nodes in L0 must agree on m, because all the fault-free nodes in sub-network II have input m.

- **Sub-network III** consists of nodes in C0, L0 and R0. In this case, the fault-free nodes are the nodes in C0, L0, and R0, with the behavior of faulty nodes being modeled by the nodes in F1 and F2. Note that $|C0 \cup L0 \cup R0| \ge n - f$. Therefore, since ALGO is correct in $G(\mathcal{V}, \mathcal{E})$, it must be correct in sub-network III. Therefore, the nodes in L0 and R0 must agree on an identical value. However, this requirement contradicts with sub-networks I and II, where nodes in R0 agree on M, and nodes in L0 agree on m, respectively.

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

B Proof of Theorem 2

We first prove two lemmas below, and then prove Theorem 2 using the two lemmas.

Lemma 6 *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 $|F| \le f$, if $B \not\to A$, then $A \overset{\mathcal{V}-F}{\rightsquigarrow} B$.*

Proof: Suppose that A, B, F is a partition of $\mathcal{V}$, where A is non-empty, $|F| \le f$, and $B \not\to A$. If $B = \emptyset$, then by Definition 2, the lemma is trivially true. In the rest of this proof, assume that $B \neq \emptyset$.

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,[2] 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 [5] implies that there exists a set $F_1 \subseteq (A \cup B) - \{b\}$ with $|F_1| \le f$, such that, in graph G_{+v}, there is no (v, b)-path excluding $F \cup F_1$. In other words, all (v, b)-paths excluding F contain at least one node in F_1.

Let us define the following sets L, R, C. Some of the sets defined in this proof are illustrated in Figure 3.

- $L = A$.

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

[2] `Footnote:` *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 an (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 $R \cup C = B$.

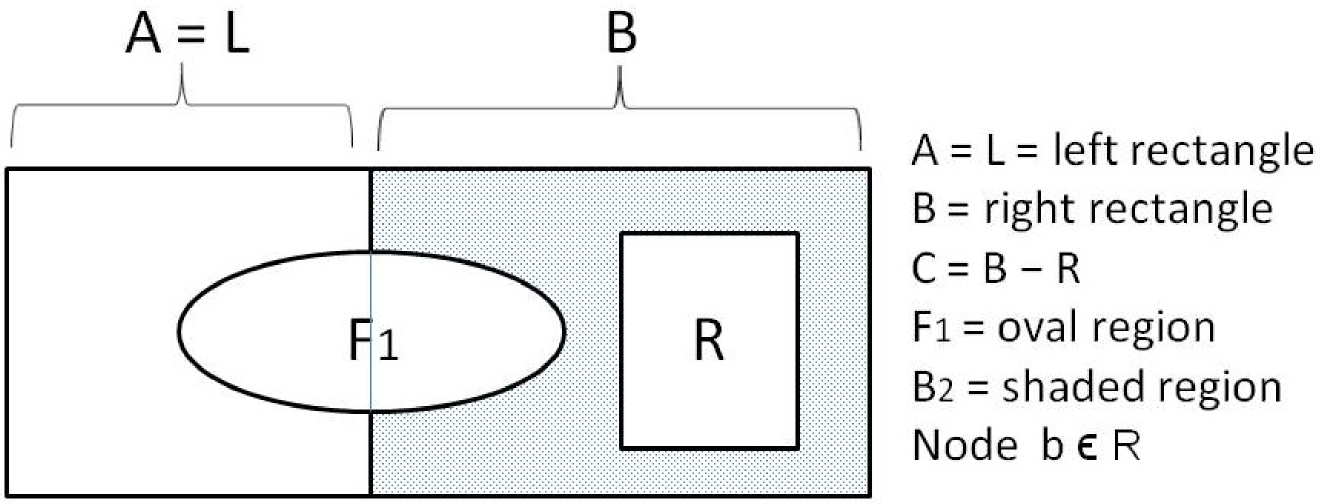

Figure 3: Illustration for the proof of Lemma 6

Observe that L, R, C are disjoint sets, because A and B are disjoint, and $L \cup R \cup C = A \cup B$. Since set $F_1 \subseteq A \cup B$, $L = A$, and $R \cap F_1 = \emptyset$, 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 $B_2 \cap F_1 = \emptyset$.

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 from definition of R 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 $B_2 \cap R \subseteq C \cap R = \emptyset$.

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

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

Recall that $B \not\Rightarrow A$. By definitions of L, R, C above, we have $A = L$ and $B = C \cup R$. Thus,

$$C \cup R \not\Rightarrow L \tag{4}$$

(3) and (4) contradict the condition in Theorem 1. Thus, we have proved that $A \overset{\mathcal{V}-F}{\rightsquigarrow} B$. $\square$

Lemma 7 *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*
- *$B' \not\Rightarrow A'$.*

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

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 $B \overset{\mathcal{V}-F}{\not\rightsquigarrow} A$, 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}.[3] Also by construction, $(w,a) \notin \mathcal{E}$. Then, by Menger's theorem [5], 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:

[3]See footnote 2.

- Sets L and R are disjoint, and set $L \subseteq A - F_1 \subseteq A$. Also, $A \cup B = L \cup R \cup C$.

  Justification: $F_1 \cap L = F_1 \cap R = \emptyset$. 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 $(B - F_1) \cap L = \emptyset$, 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 and 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 in $F \cup F_1$. Recall that $F_1 \subseteq C$ and $|F_1| \leq f$. Thus,

$$R \cup C \not\rightarrow L \tag{5}$$

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 $A' \cup B' = L \cup (R \cup C) = \mathcal{V} - F = A \cup B$.

- 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 $L \subseteq A$. Thus, $A' \subseteq A$.

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

  Justification: Recall that L, R, C are disjoint, and $L \cup R \cup C = A \cup B$. 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.

- $B' \not\rightarrow A'$

  Justification: Follows directly from (5), and the definition of A' and B'.

This concludes the proof. □

We now prove that the condition in Theorem 1 implies the condition in Theorem 2, as claimed in Section 3.2.

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 7 in Appendix B, there exist non-empty sets A', B' that form a partition of $A \cup B$ such that $A' \subseteq A$, $B \subseteq B'$, and $B' \not\rightsquigarrow A'$. Lemma 6 in Appendix B then implies that $A' \overset{\mathcal{V}-F}{\rightsquigarrow} B'$.

  Because $A' \overset{\mathcal{V}-F}{\rightsquigarrow} B'$, 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 $F \cap A = \emptyset$, 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. Therefore, $A \overset{\mathcal{V}-F}{\rightsquigarrow} B$.

□

C Proof of Claim (i) in Corollary 1 in Section 3.2

Proof: Claim (ii) in the corollary is proved in the main body of the paper already. Now, we present the proof of Claim (i).

Since $n \geq 3f+1$ is a necessary condition for Byzantine consensus in undirected graphs [4], it follows that $n \geq 3f+1$ is also necessary for directed graphs. As presented below, this necessary condition can also be derived from Theorem 1.

For $f = 0$, the corollary is trivially true. Now consider $f > 0$. The proof is by contradiction. Suppose that $n \leq 3f$. As stated in Section 1, 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 $C = \emptyset$. Since L, R are both non-empty, and contain at most f nodes each, we have $L \cup C \not\rightsquigarrow R$ and $R \cup C \not\rightsquigarrow L$, violating the condition in Theorem 1. □

D Source Component

We introduce some definitions and results that are useful in the other appendices.

Definition 5 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 6 Source component: *Let H be a directed graph, and let H^d be its decomposition as per Definition 5. 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 7 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, $\mathcal{E}_{F,F_1} = \mathcal{E} - \{(i,j) \mid i \in F \ \text{ or } \ j \in F\} - \{(i,j) \mid i \in F_1\}$.*

Corollary 2 *Suppose that graph $G(\mathcal{V}, \mathcal{E})$ satisfies the condition stated 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,

$$(\mathcal{V} - S - F) \not\rightarrow S$$

Lemma 6 in Appendix B then implies that

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

□

Lemma 8 *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}. (G_{-F} is defined in Definition 3 in Section 4.)*
- *The source component of G_{F,F_1} does not contain any nodes in F_1.*

Proof: By Definition 5, 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, a 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. □

E Sufficiency for $f = 0$

The proof below uses the terminologies and results presented in Appendix D. We now prove that, when $f = 0$, the necessary condition in Theorem 2 is sufficient to achieve consensus.

Proof:

When $f = 0$, suppose that the graph G satisfies the necessary condition in Theorem 2. Consider the source component S in reduced graph $G_{\emptyset,\emptyset} = G$, i.e., in the reduced graph where $F = F_1 = \emptyset$, as per Definition 7 in Appendix D. Note that by definition, S is non-empty. Pick a node i in the source component. By Lemma 8 in Appendix D, S is strongly connected in G, and thus i has a directed path to each of the nodes in S. By Corollary 2 in Appendix D, because $F = \emptyset$, $S \overset{\mathcal{V}}{\rightsquigarrow} \mathcal{V} - S$, i.e., for each node $j \in \mathcal{V} - S$, an (S, j)-path exists. Since S is strongly connected, an (i, j)-path also exists. Then consensus can be achieved simply by node i routing its input to all the other nodes, and requiring all the nodes to adopt node i's input as the output (or decision) for the consensus. It should be easy to see that termination, validity and agreement properties are all satisfied.

□

F Proof of the Claims in Section 4.4

The proof below uses the terminologies and results presented in Appendix D. We first prove a simple lemma.

Lemma 9 *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 prove the two claims in in Section 4.4.

Proof of Claim (i) in Section 4.4:

Consider the two cases in the INNER loop. We now prove that set S as required in Case 1 and Case 2 of the INNER loop always 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 7 in Appendix B, there exist non-empty sets A', B' that form a partition of $A \cup B = \mathcal{V} - F$ such that $A' \subseteq A$ and

  $$B' \not\rightarrow A'$$

  Let F_1 be the set of incoming neighbors of A' in B'. Since $B' \not\rightarrow A'$, $|F_1| \leq f$. 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'. (The definition of source component is in Appendix D.) Let S denote the set of nodes in this source component. Since S is the source component, by Corollary 2 in Appendix D,

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

  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 9 above). By Lemma 8 in Appendix D, S is strongly connected in G_{-F}. (The definition of G_{-F} is in Section 4.) Thus, set S as required in Case 1 exists.

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

  Recall that we consider $f > 0$ in Section 4.4.

  By Corollary 1 in Section 3.2, 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 $|F_1| = f > 0$, 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 2 in Appendix D,

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

  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, and F_1 is non-empty, $\mathcal{V} - S - F$ is non-empty; also, since $S \overset{\mathcal{V}-F}{\rightsquigarrow} \mathcal{V} - S - F$, set S must be non-empty (by Lemma 9 above). By Lemma 8 in Appendix D, S is strongly connected in G_{-F}. Thus, set S as required in Case 2 exists.

□

Proof of Claim (ii) in Section 4.4:

Consider nodes in set F. As shown in Corollary 1 in Section 3.2, 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 $\mathcal{V} - F$. Thus, the desired set N_k exists, satisfying the requirement in step (j) of Algorithm BC.

□

G Proof of Lemma 5

The proof below uses the terminologies and results presented in Appendix D. Now, we present the proof of Lemma 5.

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

Since the OUTER loop of Algorithm BC 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 INNER loop iteration is a "deciding" iteration if one of the following condition is true.

C1 : The A, B partition of $\mathcal{V} - F$ considered in the 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 A (including $S \cap A$) 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 at the end of step (g).

C2 : The A, B partition of $\mathcal{V} - F$ considered in the iteration is non-conformant; however, the values at the nodes are such that, at the end of step (b) of Case 1, or at the end of step (g) of Case 2 (depending on which case applies), every node in the corresponding set S has the same value t. (The definition of source component is in Appendix D.) 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 INNER loop iteration under consideration is a non-deciding iteration. Observe that since the paths used in procedures `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, and 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, at the end of the INNER loop iteration, the value v at each node stays unchanged based on the following two observations:

- nodes in S in Case 1, and 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, and for each node $i \in \mathcal{V} - F - (A \cap S)$ in Case 2.

Thus, no node in $\mathcal{V} - F$ will change their v value (where $F = F^*$).

Note that by assumption, there is no fault-free node in $F = F^*$, and hence, we do not need to consider STEP 2 of the INNER loop. Therefore, Claim 1 is proved. □

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

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

Claim 2 *At least one INNER loop iteration for $F = F^*$ is a deciding iteration.*

Proof: The input at each process is in $\{0, 1\}$. Therefore, by repeated application of Lemma 2 in Section 4.5, it is always true that $v_i \in \{0, 1\}$ for each fault-free node i. Thus, when the OUTER iteration for $F = F^*$ begins, 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 of $\mathcal{V} - F$, the INNER loop will eventually consider either the above conformant partition, or sometime prior to considering the above conformant partition, it will consider a non-conformant partition with properties in (C2) above. □

Thus, Phase 2 will be eventually performed when $F = F^*$. Now, let us consider each phase separately:

- Phase 1: Recall that all the nodes in $\mathcal{V} - F = \mathcal{V} - F^*$ are fault-free. By Claim 1, the v_i at each fault-free node $i \in \mathcal{V} - F$ stays unchanged.
- Phase 2: Now, consider the first deciding iteration of the INNER loop.

  Recall from Algorithm BC that a suitable set S is identified in each INNER loop iteration. We will show that in the deciding 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 above, we found a set S and α such that for all $i \in S$, $t_i = \alpha$ after step (b) in Case 1, and after step (g) in Case 2.

  Then, consider the remaining steps in the deciding 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). (Each node $p \in S$ does not modify its v_p, which is already equal to α.)

  – 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. (Each node $p \in A \cap S$ does not modify its v_p, 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. The goal now is to show that the agreement property is not violated by actions taken in any future INNER loop iterations.
- 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 2, all future INNER loop iterations 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 OUTER loop iterations for other choices of set F. However, due to Lemma 2, the value v_i at each $i \in \mathcal{V} - F^*$ (i.e., all fault-free nodes) continues being equal to α.

Thus, Algorithm BC satisfies the *agreement* property, as stated in Section 1. $\square$

H Consensus in 2-Clique Network

We first prove the following lemma for any graph $G(\mathcal{V}, \mathcal{E})$ that satisfies the necessary condition.

Lemma 10 *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 [5], 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 $P \cap A \neq \emptyset$; 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. $\square$

Now, we use Lemma 10 to prove the following Lemma.

Lemma 11 *Suppose that $G(\mathcal{V}, \mathcal{E})$ is a 2-clique network. Then graph 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 4 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

$$A_1 \cup B_1 \overset{\mathcal{V}-F}{\rightsquigarrow} A_2 \cup B_2. \tag{6}$$

$|K_1 - F| \geq 2f + 1$ 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

$$A_1 \overset{\mathcal{V}-F}{\rightsquigarrow} B_1 \tag{7}$$

(6) and (7), along with Lemma 10 above 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 $B = B_1 \cup B_2$, $A \overset{\mathcal{V}-F}{\rightsquigarrow} B$. $\square$