Exact Byzantine Consensus on Undirected Graphs under *Local Broadcast* Model *

Muhammad Samir Khan
mskhan6@illinois.edu
University of Illinois at Urbana-Champaign

Syed Shalan Naqvi
naqvi5@illinois.edu
University of Illinois at Urbana-Champaign

Nitin H. Vaidya
nitin.vaidya@georgetown.edu
Georgetown University

ABSTRACT

This paper considers the Byzantine consensus problem for nodes with binary inputs. The nodes are interconnected by a network represented as an undirected graph, and the system is assumed to be synchronous. Under the classical point-to-point communication model, it is well-known that the following two conditions are both necessary and sufficient to achieve Byzantine consensus among n nodes in the presence of up to f Byzantine faulty nodes: $n \geq 3f + 1$ and vertex connectivity at least $2f + 1$. In the classical point-to-point communication model, it is possible for a faulty node to *equivocate*, i.e., transmit conflicting information to different neighbors. Such equivocation is possible because messages sent by a node to one of its neighbors are *not overheard* by other neighbors.

This paper considers the *local broadcast* model. In contrast to the point-to-point communication model, in the local broadcast model, messages sent by a node are received identically by *all of its neighbors*. Thus, under the local broadcast model, attempts by a node to send conflicting information can be detected by its neighbors. Under this model, we show that the following two conditions are both necessary and sufficient for Byzantine consensus: vertex connectivity at least $\lfloor 3f/2 \rfloor + 1$ and minimum node degree at least $2f$. Observe that the local broadcast model results in a lower requirement for connectivity and the number of nodes n, as compared to the point-to-point communication model.

We extend the above results to a *hybrid model* that allows some of the Byzantine faulty nodes to equivocate. The hybrid model bridges the gap between the point-to-point and local broadcast models, and helps to precisely characterize the trade-off between equivocation and network requirements.

CCS CONCEPTS

• **Theory of computation → Distributed algorithms**.

KEYWORDS

complexity and impossibility results for distributed computing, fault-tolerance, reliability, self-organization, self-stabilization

ACM Reference Format:
Muhammad Samir Khan, Syed Shalan Naqvi, and Nitin H. Vaidya. 2019. Exact Byzantine Consensus on Undirected Graphs under *Local Broadcast* Model . In *2019 ACM Symposium on Principles of Distributed Computing (PODC '19), July 29-August 2, 2019, Toronto, ON, Canada.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3293611.3331619

1 INTRODUCTION

This paper considers Byzantine consensus for nodes with binary inputs. The nodes are interconnected by a communication network represented as an undirected graph, and the system is assumed to be synchronous. Under the classical point-to-point communication model, it is well-known [7] that the following two conditions are both necessary and sufficient to achieve Byzantine consensus among n nodes in the presence of up to f Byzantine faulty nodes: $n \geq 3f + 1$ and vertex connectivity at least $2f + 1$. In the classical point-to-point communication model, it is possible for a faulty node to *equivocate* [4], i.e., transmit conflicting information to different neighbors. Such equivocation is possible because messages sent by a node to one of its neighbors are *not overheard* by other neighbors.

In contrast, in the *local broadcast* model [3, 15] considered in this paper, a message sent by any node is received identically by *all of its neighbors* in the communication network. Thus, under the local broadcast model, attempts by a node to equivocate (i.e., send conflicting information to its neighbors) can be detected by its neighbors.

This paper obtains tight necessary and sufficient conditions on the underlying communication network to be able to achieve Byzantine consensus under the *local broadcast* model. As summarized in Section 2, although there has been significant work [1, 3, 5, 6, 9–12, 15, 16, 19, 23, 25–27, 29, 33] that uses either the local broadcast model or other models that restrict equivocation, tight necessary and sufficient conditions for Byzantine consensus under the local broadcast model have not been obtained previously. In particular, this paper makes the following contributions, some of which have been documented elsewhere [13, 14, 21, 22] as well:

1) **Necessary and sufficient condition:** In Sections 4 and 5, we establish that, under the *local broadcast* model, to achieve Byzantine consensus, it is necessary and sufficient for the communication graph to have vertex connectivity at least $\lfloor 3f/2 \rfloor + 1$ and minimum degree at least $2f$. Observe that the local broadcast model results in a lower connectivity requirement as compared to the $(2f + 1)$-connectivity required under the point-to-point communication model.

*This research is supported in part by the National Science Foundation awards 1409416 and 1733872, and Toyota InfoTechnology Center. Any opinions, findings, and conclusions or recommendations expressed here are those of the authors and do not necessarily reflect the views of the funding agencies or the U.S. government.

PODC '19, July 29-August 2, 2019, Toronto, ON, Canada

ACM ISBN 978-1-4503-6217-7/19/07…$15.00
https://doi.org/10.1145/3293611.3331619

2) **Efficient algorithm:** We constructively prove the sufficiency of the tight condition stated above by presenting an algorithm that achieves Byzantine consensus. This algorithm, however, is not efficient. For the case when vertex connectivity is at least $2f$, we have a more efficient algorithm. Due to lack of space, the algorithm is presented in the full version of the paper [13]. In Section 5.3, we present a tool used in the algorithm which exploits the $2f$-connectivity. Note that for $f = 1, 2$, the tight condition presented above implies vertex connectivity of 2 and 4, respectively (i.e., connectivity equal to $2f$ when $f = 1, 2$).
3) **Hybrid model:** In Section 6, we extend the above necessary and sufficient condition to a *hybrid model* wherein at most $t \leq f$ faulty nodes may have the ability to equivocate to their neighbors (i.e., the ability to send messages to each neighbor without being overheard by the other neighbors), while the remaining faulty nodes are restricted to local broadcast (i.e., their messages will be received identically by all the neighbors). The point-to-point communication model and the local broadcast model are both obtained as special cases of the hybrid model when $t = f$ and $t = 0$, respectively. Thus, the hybrid model provides a bridge between those two models and helps to precisely characterize the trade-off between equivocation and network requirements.

2 RELATED WORK

There is a large body of work on Byzantine fault-tolerant algorithms [2, 7, 17, 20, 24]. Here we focus primarily on related work that imposes constraints on a faulty node's ability to equivocate.

Rabin and Ben-Or [26] considered a global broadcast model and showed that $n \geq 2f + 1$ is both sufficient and necessary for consensus in a synchronous system. The local broadcast model considered in our work reduces to the global broadcast model when the network is a complete graph. The necessary and sufficient conditions obtained in this paper are (not surprisingly) equivalent to $n \geq 2f + 1$ when the network is a complete graph. Clement et. al. [5] also considered non-equivocation in a *complete* graph under asynchronous communication. Our work obtains results for arbitrary *incomplete* graphs with synchronous communication.

The goal of a Byzantine broadcast algorithm is to allow a single source node to deliver its message reliably to all the other nodes. Prior work has explored such algorithms under the local broadcast model [3, 15, 16]. However, the results for Byzantine *broadcast* do not provide insights into the network requirements for Byzantine *consensus* problem considered in this paper.

There has been significant work on other similar models, sometimes called "partial broadcast". To motivate these models, consider a network consisting of several Ethernet channels, with a subset of nodes S_i being connected to channel i. Then, transmission by any node in S_i on channel i will be received by all the nodes in channel i. Each node may be connected to several different such channels. An Ethernet channel to which h nodes are connected may be viewed as an h-hyperedge in a communication network represented by a hypergraph. Fitzi and Maurer [9] considered a network in which every subset of three nodes form a hyperedge, in addition to every subset of two nodes also forming a hyperedge (i.e., a complete (2,3)-uniform hypergraph). Ravikant et. al. [27] also considered a (2,3)-uniform hypergraph, but with only a subset of 3-hyperedges being present. Jaffe et. al. [12] gave asymptotically tight bounds for the number of 3-hyperedges required in such graphs. Amitanand et. al. [1] considered a complete network wherein, for each faulty node k, the remaining nodes are partitioned such that transmission by the faulty node k to any node is received identically by all the nodes in its partition. One key difference from our work is that we consider incomplete networks, whereas [1] assumes that each node can communicate directly with all the other nodes. Amitanand et. al. [1] considered an adversary structure that characterizes the set of nodes that may be simultaneously faulty, instead of using a threshold on the number of faults. Other work [6, 10, 11, 33] has explored the trade-off between reliability and privacy on partial broadcast networks.

Some prior work has explored a *restricted* class of iterative algorithms that achieve *approximate* Byzantine consensus under the local broadcast model [18, 35]. In particular, this class of algorithms is iterative in nature, with a state variable at each node being updated in each iteration as a linear interpolation of the states of selected neighbors. Due to the restriction on the algorithm behavior, the network requirements exceed the necessary and sufficient conditions shown in this paper. Additionally, these restricted algorithm structures yield only approximate consensus in finite time. Li et. al. [19] extended this line of work to a network consisting of 3-hyperedges and 2-hyperedges. Vaidya et. al. [28, 31, 32] have investigated the iterative algorithm structure in the point-to-point communication model.

3 SYSTEM MODEL AND NOTATION

We consider a synchronous system. The communication network interconnecting n nodes is represented by an undirected graph $G = (V, E)$. Every node in the system knows graph G. Each node u is represented by vertex $u \in V$. We will use the terms *node* and *vertex* interchangeably. Two nodes u and v are *neighbors* if and only if $uv \in E$ is an edge of G. For a set $S \subseteq V$, node u is said to be a neighbor of set S if $u \notin S$ and there exists $v \in S$ such that $uv \in E$.

Each edge uv represents a FIFO link between the two nodes u and v. When a message m sent by node u is received by node v, node v knows that m was sent by node u. In Sections 4 and 5, we assume the *local broadcast* model wherein a message sent by a node u is received identically and correctly by each node v such that $uv \in E$ (i.e., by each neighbor of u). A *hybrid* model is considered later in Section 6.

A *Byzantine* faulty node may exhibit arbitrary behavior. We consider the *Byzantine consensus problem* assuming that each node has a binary input. The goal is for each node to output a binary value, satisfying the following conditions, in the presence of at most $f < n$ Byzantine faulty nodes.

1) **Agreement:** All non-faulty nodes must output the same value.
2) **Validity:** The output of each non-faulty node must be the input of some non-faulty node.
3) **Termination:** All non-faulty nodes must decide on their output in finite time.

Paths in graph G: A path is a sequence of nodes such that any two adjacent nodes in the sequence are neighbors in the graph.

- For $u, v \in V$, a uv-path P_{uv} is a path between nodes u and v. u and v are *endpoints* of path P_{uv}. Any node in path P_{uv} that

is not an endpoint is said to be an *internal* node of P_{uv}. All uv-paths have u and v as endpoints, by definition. Two uv-paths are *node-disjoint* if they do not have any internal nodes in common.

- For $U \subset V$ and a node $v \notin U$, a Uv-path is a uv-path for some node $u \in U$. All Uv-paths have v as one endpoint. Two Uv-paths are *node-disjoint* if they do not have any nodes in common except endpoint v. In particular, although two node-disjoint Uv-paths have endpoint v in common, the other endpoints are distinct for the two paths.

Fault-free paths: A path is said to *exclude* a set of nodes $X \subseteq V$ if no internal node of the path belongs to X; however, its endpoints may potentially belong to X. A path is said to be *fault-free* if none of its internal nodes are faulty. In other words, a path is fault-free if it excludes the set of faulty nodes. Note that a fault-free path may have a faulty node as an endpoint.

Degree and Connectivity: The *degree* of a node u is the number of u's neighbors (i.e., the number of edges incident to u). Minimum degree of G is the minimum over the degree of all the vertices in G. A graph G is k-connected if $n > k$ and removal of less than k nodes does not disconnect G. By Menger's Theorem [34] a graph G is k-connected if and only if for any two nodes $u, v \in V$ there exist k node disjoint uv-paths. Another standard result [34] for k-connected graphs is that if G is k-connected, then for any node v and a set of at least k nodes U there exist k node-disjoint Uv-paths.

4 NECESSARY CONDITIONS UNDER LOCAL BROADCAST

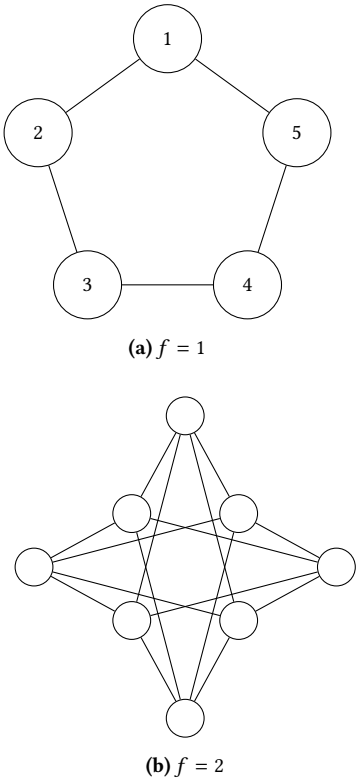
Theorem 4.1 below states the necessary conditions for Byzantine consensus under the local broadcast model. Section 5 presents a Byzantine consensus algorithm and constructively proves that the necessary conditions are also sufficient.

Theorem 4.1. *If there exists a Byzantine consensus algorithm under the local broadcast model on graph G tolerating at most f Byzantine faulty nodes, then both the following conditions must be true: (i) G has minimum degree at least $2f$, and (ii) G is $(\lfloor 3f/2 \rfloor + 1)$-connected.*

Appendix A presents a proof of Theorem 4.1. Necessity of conditions (i) and (ii) in the theorem is proved separately in Lemmas A.1 and A.2, respectively. We use a state machine based approach [2, 7, 8] to prove Lemmas A.1 and A.2.

It should be easy to see that a complete graph consisting of $2f + 1$ nodes satisfies the necessary conditions in Theorem 4.1 for any f. Figure 1 presents other examples of graphs that satisfy the necessary conditions. Figure 1(a) shows a cycle consisting of 5 nodes. For this graph, the minimum degree is 2, and the graph is 2-connected. Thus, the cycle satisfies the conditions in Theorem 4.1 for $f = 1$. The graph in Figure 1(b) satisfies the necessary conditions when $f = 2$.

Section 5 below proves that the above necessary conditions are also sufficient. Before proceeding to Section 5, let us consider the 5-node cycle in Figure 1(a) to build some intuition on why these conditions may be sufficient. Since $f = 1$, we do not have $2f + 1$ node-disjoint paths between every pair of nodes in the cycle in Figure 1(a). Despite lower connectivity, we can show a useful property. In particular, suppose that node 1 attempts to send a message M to node 4, by transmitting it along two node-disjoint paths 1-2-3-4 and 1-5-4, respectively. That is, node 1 will send the message to nodes 2 and 5 (its neighbors) in a single transmission – local broadcast model allows node 1 to send its message to all its neighbors simultaneously. The neighbors 2 and 5 will then forward the message and subsequently, node 3 will forward the message received from node 2. Due to the local broadcast model, all neighbors of each node will receive its transmissions. Consider two cases:



(a) $f = 1$

(b) $f = 2$

Figure 1: Graphs satisfying conditions in Theorem 4.1

- Case (i): The internal nodes (namely, 2, 3 and 5) on the two paths behave correctly: In this case, node 4 receives identical copies of the message along the two disjoint paths. Because $f = 1$, node 4 can be certain that it has correctly received the message that was transmitted by node 1 (when $f = 1$, internal nodes on at most one disjoint path may be faulty).
- Case (ii): Node 3 is faulty, and tampers the message received from node 2 before forwarding it to node 4: In this case, node 4 will not receive two identical copies of the message along the two disjoint paths. Therefore, node 4 cannot determine the message sent by node 1. However, in this case, node 2 is non-faulty. Node 2 correctly forwards message M received from node 1 to node 3, and then observes that node 3 is forwarding a tampered message to node 4, not the correct message M. Node 2 can observe the faulty behavior of node 3 due to the local broadcast property – when node 3 sends the tampered message to node 4, node 2 receives it too, because node 2 is a neighbor of node 3.
  Node 2 can now notify node 1 that node 3 is faulty. Of course, on receiving this notification, node 1 cannot be certain whether

node 3 is indeed faulty, or node 2 is faulty and it is incorrectly accusing node 3 of misbehavior. However, node 1 can be certain that one of nodes 2 and 3 must be faulty. Because $f = 1$, node 1 can then infer that the path 4-5-1 is fault-free. Thus, any messages received by node 1 from node 4 along the path 4-5-1 could not be tampered by the internal node on this path, namely, node 5. This allows node 1 to now receive messages transmitted by node 4 reliably.

The example above can be generalized to derive a similar capability for reliable communication in at least one direction between each pair of nodes. Although the algorithm presented next does not seem to explicitly utilize this property, it implicitly relies on such a behavior.

5 BYZANTINE CONSENSUS ALGORITHM UNDER LOCAL BROADCAST

Theorem 5.1 below states that the necessary conditions in Theorem 4.1 are also sufficient.

Theorem 5.1. *Under the local broadcast model, Byzantine consensus tolerating at most f Byzantine faulty nodes is achievable on graph G if both the following conditions are true: (i) G has minimum degree at least $2f$, and (ii) G is $(\lfloor 3f/2 \rfloor + 1)$-connected.*

We prove correctness of Theorem 5.1 constructively by providing a Byzantine consensus algorithm (in Section 5.1) and showing its correctness (in Section 5.2).

5.1 Proposed Algorithm

Assume that graph G satisfies the properties stated in Theorem 5.1. That is, G has minimum degree at least $2f$ and is $(\lfloor 3f/2 \rfloor + 1)$-connected. Pseudo-code for the proposed algorithm is presented below. To understand the description presented next, it will help the reader to read the corresponding steps in Algorithm 1 below.

Initialization: Each node $v \in V$ maintains a local state variable named γ_v, which is initialized to equal node v's binary input.

Phases of the algorithm: Recall that we assume a synchronous system. The execution of the algorithm is divided into many *phases*, each phase corresponding to a distinct subset of nodes F, such that F contains at most f nodes. Each iteration of the `For` loop in Algorithm 1 corresponds to one phase of the algorithm. The set F chosen in each phase is a *candidate* for the actual set of faults; however, set F in only one of the phases will exactly equal the actual set of faulty nodes in the given execution. This is similar to the Byzantine consensus algorithm for directed graphs under the point-to-point communication model by Tseng and Vaidya [30]. However, the rest of the algorithm proceeds differently since we consider the local broadcast model.

Step (a): At the beginning of each phase, each node v attempts to communicate its current value of γ_v via "flooding", as described soon. Any message transmitted during flooding has the form (b, Π), where $b \in \{0, 1\}$ and Π is a path. Flooding proceeds in synchronous rounds, with each node possibly forwarding messages received in the previous round, following the rules presented below. Flooding will end after n rounds, as should be clear from the following description.

To initiate flooding of its γ_v value, node v transmits message $(\gamma_v, \perp)$ to its neighbors, where $\perp$ represents an empty path. Each node in the network similarly initiates flooding of its own γ value in step (a). If v is faulty and does not initiate flooding, then non-faulty neighbors of v replace the missing message with the default message of $(1, \perp)$. Therefore, we can assume that a value is indeed flooded by each node, even if it is faulty. When node v receives from a neighbor u a message (b, Π), where $b \in \{0, 1\}$ and Π is a path, it takes the following steps sequentially. In the following, Π - u denotes a path obtained by concatenating identifier u to path Π.

(i) If path Π - u does not exist in graph G, then node v discards the message (b, Π). Recall that each node knows graph G, and the message (b, Π) was received by node v from node u.

(ii) Else if, in the current phase, node v has previously received from u another message containing path Π (i.e., a message of the form (b', Π)), then node v discards the message (b, Π).

(iii) Else if path Π already includes node v's identifier, node v discards the message (b, Π).

(iv) Else node v is said to have received value b along path Π - u and node v forwards message $(b, \Pi$ - $u)$ to its neighbors. Recall that v received message (b, Π) from neighbor u.

Rules (i) and (ii) above are designed to prevent a faulty node from sending spurious messages. Recall that, under the local broadcast model, all neighbors of any node receive all its transmissions. Thus, rule (ii) above essentially prevents a faulty node from successfully delivering mismatching messages to its neighbors (i.e., this prevents equivocation). Rule (iii) ensures that flooding will terminate after n rounds.

Rule (ii) above crucially also ensures the following useful property due to the local broadcast model: even if node u is Byzantine faulty, but paths P_{uv} and P_{uw} are fault-free, then nodes v and w will receive identical value in the message from u forwarded along paths P_{uv} and P_{uw}, respectively. Recall that a path is fault-free if none of the internal nodes on the path are faulty.

Step (b): Recall that a path is said to exclude set F if none of its *internal* nodes are in F. For each $u \in V$, node v chooses an arbitrary uv-path P_{uv} that excludes F. It can be shown (Lemma 5.4) that such a path always exists under the conditions in Theorem 5.1. For the purpose of step (b), node v is deemed to have received its own γ_v along path P_{vv} (containing only node v). Node v computes sets Z_v and $N_v = V - Z_v$, as shown in the pseudo-code.

Step (c): This step specifies the rules for updating value γ_v. γ_v is not necessarily updated in each phase.

Output: After all the phases (i.e., all iterations of the `For` loop) are completed, the value of γ_v is chosen as the output of node v.

The proof of correctness of Algorithm 1 is presented in Section 5.2. As we discussed earlier when describing the flooding mechanism, the faulty nodes are effectively limited to delivering a unique value on all fault-free paths (i.e., paths that do not have faulty nodes as internal nodes). If this unique value corresponding to a faulty node is 0, we will say that the faulty node flooded value 0; else we will say that the node flooded value 1.

Let Z be the set of nodes that flooded 0 in step (a) and let $N = V - Z$ be the set of the remaining nodes that flooded 1. Recall from

the description of flooding above, that even a faulty node effectively floods one value, and one value only, in step (a) of each phase. Note that either Z or N may possibly be empty, but not both. In step (b), each non-faulty node v obtains its own estimates Z_v and N_v of Z and V, respectively. If the set F in the current phase does not contain all the faulty nodes, then these estimates can be incorrect. However, at least in one phase, F will contain all the faulty nodes and, in that phase, as shown later, it is guaranteed that $Z = Z_v$ and $N = N_v$. This observation is important for the correctness of the algorithm, as seen later.

As shown later, step (c) is designed to ensure that the fault-free nodes will reach consensus in a phase in which F contains all the faulty nodes. In each phase, step (c) also ensures the invariant that γ_v, at any non-faulty node v, at the end of the phase is equal to γ_u, for some non-faulty node u, at the start of that phase.

Algorithm 1: Proposed algorithm for Byzantine consensus under the local broadcast model: Steps performed by node v are shown here.

Each node v has a binary input value in $\{0, 1\}$ and maintains a binary state $\gamma_v \in \{0, 1\}$.

Initialization: $\gamma_v :=$ input value of node v

For *each* $F \subseteq V$ *such that* $|F| \leq f$ **do**

Step (a): Flood value γ_v. (The steps taken to achieve flooding are described in the text preceding Algorithm 1 pseudo-code.)

Step (b): For each node $u \in V$, identify a single uv-path P_{uv} that excludes F. Let,

$Z_v := \{u \in V \mid$ v received value 0 from u along P_{uv} in step (a) $\}$,

$N_v := V - Z_v$.

Step (c): Define sets A_v and B_v as follows.

Case 1: If $|Z_v \cap F| \leq \lfloor f/2 \rfloor$ and $|N_v| > f$, then $A_v := N_v$ and $B_v := Z_v$.

Case 2: If $|Z_v \cap F| \leq \lfloor f/2 \rfloor$ and $|N_v| \leq f$, then $A_v := Z_v$ and $B_v := N_v$.

Case 3: If $|Z_v \cap F| > \lfloor f/2 \rfloor$ and $|Z_v| > f$, then $A_v := Z_v$ and $B_v := N_v$.

Case 4: If $|Z_v \cap F| > \lfloor f/2 \rfloor$ and $|Z_v| \leq f$, then $A_v := N_v$ and $B_v := Z_v$.

If $v \in B_v$ and v receives value $\delta \in \{0, 1\}$ along any $f + 1$ node-disjoint $A_v v$-paths that exclude F in step (a), then $\gamma_v := \delta$.

end

Output γ_v.

5.2 Proof of Correctness of Algorithm 1

In this section, we assume that graph G satisfies the conditions stated in Theorem 5.1, even if this is not always stated explicitly below. Appendix B presents the proofs of the lemmas in this section.

The proof of correctness relies on two key lemmas, Lemma 5.2 and 5.3. We will present additional results and discuss the intuition behind the proofs of these lemmas subsequently. Formal proofs are presented in Appendix B. Recall that the algorithm execution is divided into *phases*, each phase corresponding to a different choice of F, where $|F| \leq f$. For convenience of presentation, we will refer to γ_v as the "state of node v". Recall that state γ_v of node v may possibly be modified in step (c) in each phase.

Lemma 5.2. *For a non-faulty node v, its state γ_v at the end of any given phase equals the state of some non-faulty node at the start of that phase.*

Lemma 5.3. *Consider a phase of Algorithm 1 wherein all the faulty nodes are contained in set F corresponding to that phase. At the end of this phase, every pair of non-faulty nodes $u, v \in V$ have identical state, i.e., $\gamma_u = \gamma_v$.*

As shown next, these two lemmas imply correctness of Algorithm 1, thus proving Theorem 5.1.

Proof of Theorem 5.1: Algorithm 1 terminates in finite time because the number of phases is finite, and flooding in each phase completes in finite time. Thus, the algorithm satisfies the *termination* condition.

Since there are at most f faulty nodes in any given execution, there exists at least one phase in which set F will contain all the faulty nodes. Then, from Lemma 5.3, we have that all non-faulty nodes have the same state at the end of this phase. Lemma 5.2 implies that the state of the non-faulty nodes will remain unchanged after any subsequent phases. Therefore, all non-faulty nodes will have the same state at the end of the algorithm and their output will be identical. This proves that the algorithm satisfies the *agreement* condition.

At the start of phase 1, the state of each non-faulty node equals its own input. Now, applying Lemma 5.2 inductively implies that the state of a non-faulty node always equals the *input* of some non-faulty node. This, in turn, implies that the algorithm satisfies the *validity* condition. Thus, we have proved correctness of Algorithm 1 under the conditions stated in Theorem 5.1. □

The proofs of Lemmas 5.2 and 5.3, stated above, rely on two other lemmas, presented next. The reader may skip the rest of this section without a loss of continuity. Lemma 5.4 below implies that path P_{uv} used in step (b) of the algorithm indeed exists.

Lemma 5.4. *For any choice of set F in the algorithm, and any two nodes $u, v \in V$, there exists a uv-path that excludes F.*

Proof: Recall that a path is said to exclude F if none of the *internal* nodes in the path belong to F. Since G is $(\lfloor 3f/2 \rfloor + 1)$-connected, by Menger's Theorem, there are at least $\lfloor 3f/2 \rfloor + 1$ node-disjoint paths between any two nodes u, v. For any $f \geq 0$, we have $\lfloor 3f/2 \rfloor + 1 \geq f + 1$. Thus, there are at least $f + 1$ node-disjoint uv-paths, of which at least one path must exclude F, since $|F| \leq f$. □

The next lemma states that the choice of sets A_v and B_v in step (c) ensures that the node-disjoint paths used in that step indeed exist.

Lemma 5.5. *For any non-faulty node v, and any given phase with the corresponding set F, in step (c), if $v \in B_v$, then there exist $f + 1$ node-disjoint $A_v v$-paths that exclude F.*

The formal proof of the above lemma is presented in Appendix B. Observe that there are four distinct cases in step (c) for determining

sets A_v and B_v. The proof of the above claim in cases 1 and 3 in step (c) follows from $(\lfloor 3f/2 \rfloor + 1)$-connectivity of graph G, while the proof in cases 2 and 4 follows from the fact that the minimum degree of G is at least $2f$.

In step (c), observe that if node v modifies its state γ_v, then it must have received identical value along $f + 1$ node-disjoint $A_v v$ paths. So, at least one of these path must not only be fault-free, but also have non-faulty endpoints. The proof of Lemma 5.2 in Appendix B relies on this observation.

Now consider Lemma 5.3. The correctness of this lemma relies on the local broadcast property and rule (ii) used in flooding. Suppose that set F in a certain phase contains all the faulty nodes. Since the paths used in step (b) of the algorithm exclude set F, these paths are fault-free (i.e., none of their internal nodes are faulty). Then, the earlier discussion of flooding implies that any two non-faulty nodes u, v will obtain $Z_u = Z_v$ and $N_u = N_v$ in step (b) of this phase. By a similar argument, all the paths used in step (c) of this phase are also fault-free, and any two non-faulty nodes will end step (c) of this phase with an identical state. A complete proof of Lemma 5.3 is presented in Appendix B.

5.3 An Efficient Algorithm

The number of phases in Algorithm 1 is exponential in f, since there exists one phase corresponding to each choice of set F such that $|F| \leq f$. When the communication graph G is $2f$-connected, we have developed an efficient algorithm that requires $O(n)$ time. Although in general $2f$-connectivity is a stronger requirement on graph G as compared to the requirement in Theorem 5.1, observe that when $f = 1, 2$ these two requirements are equivalent.

Theorem 5.6. *Under the local broadcast model, Byzantine consensus tolerating at most f Byzantine faulty nodes is achievable on graph G in $O(n)$ synchronous rounds if G is $2f$-connected.*

We prove Theorem 5.6 constructively by giving an efficient algorithm. Due to lack of space, the efficient algorithm is presented in the full version of the paper [13]. Using the example in Figure 1(a), we now illustrate a tool used in that algorithm, which exploits the $2f$-connectivity. The graph in Figure 1(a) is 2-connected, i.e., $2f$-connected for $f = 1$. Similar to our algorithm in the previous section, suppose that node 1 floods value b, which is propagated to node 4 along path 1-2-3-4. When node 2 forwards the value to node 3, all of node 2's neighbors hear the forwarded message. Similarly, when node 3 forwards the message to node 4, all of node 3's neighbors hear the forwarded message.

Suppose now that we ask each node to additionally "report" on its neighbors by flooding any messages heard/received in the above propagation from node 1 to node 5. Then the message forwarded by node 4, as overheard by node 3, will be flooded by node 3. Since the graph G is $2f$-connected, node 4 has at least $2f$ neighbors, each of which will take similar steps. Now consider two cases:

- Node 3 is faulty: Since node 3 is faulty, there are at most $f - 1$ other faulty nodes. In this case, $2f$-connectivity implies that there are at least $f + 1$ node-disjoint fault-free paths from neighbors of node 3 to node 1. Thus, node 1 can receive $f + 1$ identical and correct reports of all messages transmitted by node 3. Therefore, node 1 can correctly determine the message forwarded by node 3. In general, each node can correctly learn messages transmitted by any faulty node, when the connectivity is $2f$.
- Node 3 is non-faulty: In this case, it is possible that node 1 does not receive identical reports about node 3's messages on $f + 1$ node-disjoint paths. This inability to receive $f + 1$ identical reports, however, allows node 1 to infer that node 3 must be non-faulty.

In summary, as illustrated above, each node can observe all messages sent by any faulty node. Also, each node can either observe all messages sent by another non-faulty node, or learn that it is non-faulty. In some instances, these observations help non-faulty nodes identify the faulty nodes accurately. Our algorithm in [13] relies on this property to improve the time complexity.

6 HYBRID MODEL

In this section we consider a hybrid model. The hybrid model is designed to help explore the gap between the network requirements for Byzantine consensus under the point-to-point communication model and the local broadcast model. Under the hybrid model, up to $f < n$ nodes may be Byzantine faulty. The faulty nodes are of two types:

- Equivocating faulty nodes: At most $t \leq f$ of the faulty nodes may equivocate, that is, they are not restricted to perform local broadcast. If u is an equivocating faulty node and has neighbors v and w, then node u may send message M_v to node v without node w receiving M_v, and similarly send message M_w to node w without node v receiving M_w. Thus, equivocating faulty nodes can behave similar to the faulty nodes under the point-to-point communication model.
- Non-equivocating faulty nodes: Any faulty node that is not an equivocating faulty node is said to be a non-equivocating faulty node. A non-equivocating faulty node conforms to the local broadcast model. Thus, if node u is a non-equivocating faulty node, and has neighbors v and w, then any message M transmitted by node u will be received identically by v and w both.

Observe that when $t = 0$, the hybrid model reduces to the local broadcast model, since all the faulty nodes are restricted to perform local broadcast. On the other hand, when $t = f$, the hybrid model reduces to the classical point-to-point communication model because all the faulty nodes can equivocate. The following theorem extends results from Sections 4 and 5 to the hybrid model.

Theorem 6.1. *Under the hybrid model, Byzantine consensus tolerating at most f Byzantine faulty nodes, of which at most t are equivocating faulty nodes, is achievable on graph G if and only if all the following conditions are true:*

(i) G is $(\lfloor 3(f - t)/2 \rfloor + 2t + 1)$-connected,

(ii) if $t = 0$, then G has minimum degree at least $2f$, and

(iii) if $t > 0$, then every set of nodes S, such that $0 < |S| \leq t$, has at least $2f + 1$ neighbors.

Observe that when $t = 0$, condition (ii) lower bounds the number of neighbor of each vertex. On the other hand, when $t > 0$, condition (iii) lower bounds the number of neighbors of each subset of nodes of size at most t. Recall that neighbors of S are nodes outside of S that have an edge to some node in S. Theorem 6.1 is proved in the full version of the paper [13]. Consider three cases:

- When $t = 0$, as noted earlier, the hybrid model reduces to the local broadcast model. In this case, condition (iii) imposes no restrictions, and conditions (i) and (ii) reduce to the graph requirements in Theorems 4.1 and 5.1.
- When $t = f$, the hybrid model reduces to the point-to-point communication model. In this case, condition (ii) imposes no restrictions, condition (i) requires G to be $(2f + 1)$-connected, and condition (iii) implies $n \geq 3f + 1$.
- When $0 < t < f$, the above theorem provides insights into the trade-off between equivocation and network requirements.

The necessity and sufficiency of the conditions in Theorem 6.1 is proved using similar techniques as the proofs of Theorems 4.1 and 5.1, respectively. The proof of Theorem 6.1 appears in [13]. To prove sufficiency, Algorithm 1 is modified to obtain an algorithm for the hybrid model – the modified algorithm is presented in [13].

7 SUMMARY

In this work, we investigated Byzantine Consensus under the local broadcast model. We showed that $(\lfloor 3f/2 \rfloor + 1)$-connectivity and minimum degree at least $2f$ are together necessary and sufficient conditions to achieve Byzantine consensus in the presence of at most f Byzantine faults under the local broadcast model. The sufficiency proof is constructive. However, the algorithm presented requires exponential synchronous rounds. For a stronger network condition of $2f$-connectivity, an efficient algorithm achieves consensus in linear number of rounds. We leave finding an efficient algorithm for the tight condition for future work.

We also considered a hybrid model where some faulty nodes may equivocate but other faulty nodes are restricted to local broadcast. We presented necessary and sufficient conditions for this model, which provide insights into the trade-off between equivocation and network requirements.

REFERENCES

[1] S. Amitanand, I. Sanketh, K. Srinathant, V. Vinod, and C. Pandu Rangan. 2003. Distributed Consensus in the Presence of Sectional Faults. In *Proceedings of the Twenty-second Annual Symposium on Principles of Distributed Computing (PODC '03)*. ACM, New York, NY, USA, 202–210. https://doi.org/10.1145/872035.872065

[2] Hagit Attiya and Jennifer Welch. 2004. *Distributed Computing: Fundamentals, Simulations and Advanced Topics.* John Wiley & Sons, Inc., USA.

[3] Vartika Bhandari and Nitin H. Vaidya. 2005. On Reliable Broadcast in a Radio Network. In *Proceedings of the Twenty-fourth Annual ACM Symposium on Principles of Distributed Computing (PODC '05)*. ACM, New York, NY, USA, 138–147. https://doi.org/10.1145/1073814.1073841

[4] Byung-Gon Chun, Petros Maniatis, Scott Shenker, and John Kubiatowicz. 2007. Attested Append-only Memory: Making Adversaries Stick to Their Word. *SIGOPS Oper. Syst. Rev.* 41, 6 (Oct. 2007), 189–204. https://doi.org/10.1145/1323293.1294280

[5] Allen Clement, Flavio Junqueira, Aniket Kate, and Rodrigo Rodrigues. 2012. On the (Limited) Power of Non-equivocation. In *Proceedings of the 2012 ACM Symposium on Principles of Distributed Computing (PODC '12)*. ACM, New York, NY, USA, 301–308. https://doi.org/10.1145/2332432.2332490

[6] Jeffrey Considine, Matthias Fitzi, Matthew Franklin, Leonid A. Levin, Ueli Maurer, and David Metcalf. 2005. Byzantine Agreement Given Partial Broadcast. *Journal of Cryptology* 18, 3 (01 Jul 2005), 191–217. https://doi.org/10.1007/s00145-005-0308-x

[7] Danny Dolev. 1982. The Byzantine generals strike again. *Journal of Algorithms* 3, 1 (1982), 14 – 30. https://doi.org/10.1016/0196-6774(82)90004-9

[8] Michael J. Fischer, Nancy A. Lynch, and Michael Merritt. 1986. Easy impossibility proofs for distributed consensus problems. *Distributed Computing* 1, 1 (01 Mar 1986), 26–39. https://doi.org/10.1007/BF01843568

[9] Mattias Fitzi and Ueli Maurer. 2000. From Partial Consistency to Global Broadcast. In *Proceedings of the Thirty-second Annual ACM Symposium on Theory of Computing (STOC '00)*. ACM, New York, NY, USA, 494–503. https://doi.org/10.1145/335305.335363

[10] Matthew Franklin and Rebecca N. Wright. 2000. Secure Communication in Minimal Connectivity Models. *Journal of Cryptology* 13, 1 (01 Jan 2000), 9–30. https://doi.org/10.1007/s001459910002

[11] M. Franklin and M. Yung. 2004. Secure Hypergraphs: Privacy from Partial Broadcast. *SIAM Journal on Discrete Mathematics* 18, 3 (2004), 437–450. https://doi.org/10.1137/S0895480198335215 arXiv:https://doi.org/10.1137/S0895480198335215

[12] Alexander Jaffe, Thomas Moscibroda, and Siddhartha Sen. 2012. On the Price of Equivocation in Byzantine Agreement. In *Proceedings of the 2012 ACM Symposium on Principles of Distributed Computing (PODC '12)*. ACM, New York, NY, USA, 309–318. https://doi.org/10.1145/2332432.2332491

[13] Muhammad Samir Khan, Syed Shalan Naqvi, and Nitin H. Vaidya. 2019. Exact Byzantine Consensus on Undirected Graphs under Local Broadcast Model. *CoRR* abs/1903.11677 (2019). arXiv:1903.11677 http://arxiv.org/abs/1903.11677

[14] Muhammad Samir Khan and Nitin H. Vaidya. 2019. Byzantine Consensus under Local Broadcast Model: Tight Sufficient Condition. *CoRR* abs/1901.03804 (2019). arXiv:1901.03804 http://arxiv.org/abs/1901.03804

[15] Chiu-Yuen Koo. 2004. Broadcast in Radio Networks Tolerating Byzantine Adversarial Behavior. In *Proceedings of the Twenty-third Annual ACM Symposium on Principles of Distributed Computing (PODC '04)*. ACM, New York, NY, USA, 275–282. https://doi.org/10.1145/1011767.1011807

[16] Chiu-Yuen Koo, Vartika Bhandari, Jonathan Katz, and Nitin H. Vaidya. 2006. Reliable Broadcast in Radio Networks: The Bounded Collision Case. In *Proceedings of the Twenty-fifth Annual ACM Symposium on Principles of Distributed Computing (PODC '06)*. ACM, New York, NY, USA, 258–264. https://doi.org/10.1145/1146381.1146420

[17] Leslie Lamport, Robert Shostak, and Marshall Pease. 1982. The Byzantine Generals Problem. *ACM Trans. Program. Lang. Syst.* 4, 3 (July 1982), 382–401. https://doi.org/10.1145/357172.357176

[18] H. J. LeBlanc, H. Zhang, X. Koutsoukos, and S. Sundaram. 2013. Resilient Asymptotic Consensus in Robust Networks. *IEEE Journal on Selected Areas in Communications* 31, 4 (April 2013), 766–781. https://doi.org/10.1109/JSAC.2013.130413

[19] C. Li, M. Hurfin, Y. Wang, and L. Yu. 2016. Towards a Restrained Use of Non-Equivocation for Achieving Iterative Approximate Byzantine Consensus. In *2016 IEEE International Parallel and Distributed Processing Symposium (IPDPS)*. 710–719. https://doi.org/10.1109/IPDPS.2016.62

[20] Nancy A. Lynch. 1996. *Distributed Algorithms.* Morgan Kaufmann Publishers Inc., San Francisco, CA, USA.

[21] Syed Shalan Naqvi. 2018. *Exact Byzantine consensus under local-broadcast channels (Advisor: Nitin Vaidya).* Master's thesis. University of Illinois at Urbana-Champaign.

[22] Syed Shalan Naqvi, Muhammad Samir Khan, and Nitin H. Vaidya. 2018. Exact Byzantine Consensus Under Local-Broadcast Model. *CoRR* abs/1811.08535 (2018). arXiv:1811.08535 http://arxiv.org/abs/1811.08535

[23] Aris Pagourtzis, Giorgos Panagiotakos, and Dimitris Sakavalas. 2017. Reliable broadcast with respect to topology knowledge. *Distributed Computing* 30, 2 (01 Apr 2017), 87–102. https://doi.org/10.1007/s00446-016-0279-6

[24] M. Pease, R. Shostak, and L. Lamport. 1980. Reaching Agreement in the Presence of Faults. *J. ACM* 27, 2 (April 1980), 228–234. https://doi.org/10.1145/322186.322188

[25] Andrzej Pelc and David Peleg. 2005. Broadcasting with Locally Bounded Byzantine Faults. *Inf. Process. Lett.* 93, 3 (Feb. 2005), 109–115. https://doi.org/10.1016/j.ipl.2004.10.007

[26] T. Rabin and M. Ben-Or. 1989. Verifiable Secret Sharing and Multiparty Protocols with Honest Majority. In *Proceedings of the Twenty-first Annual ACM Symposium on Theory of Computing (STOC '89)*. ACM, New York, NY, USA, 73–85. https://doi.org/10.1145/73007.73014

[27] D. V. S. Ravikant, V. Muthuramakrishnan, V. Srikanth, K. Srinathan, and C. Pandu Rangan. 2004. On Byzantine Agreement over (2,3)-Uniform Hypergraphs. In *Distributed Computing*, Rachid Guerraoui (Ed.). Springer Berlin Heidelberg, Berlin, Heidelberg, 450–464.

[28] Lewis Tseng and Nitin Vaidya. 2013. Iterative Approximate Byzantine Consensus under a Generalized Fault Model. In *Distributed Computing and Networking*, Davide Frey, Michel Raynal, Saswati Sarkar, Rudrapatna K. Shyamasundar, and Prasun Sinha (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 72–86.

[29] Lewis Tseng, Nitin Vaidya, and Vartika Bhandari. 2015. Broadcast using certified propagation algorithm in presence of Byzantine faults. *Inform. Process. Lett.* 115, 4 (2015), 512 – 514. https://doi.org/10.1016/j.ipl.2014.11.010

[30] Lewis Tseng and Nitin H. Vaidya. 2012. Exact Byzantine Consensus in Directed Graphs. *CoRR* abs/1208.5075 (2012). arXiv:1208.5075 http://arxiv.org/abs/1208.5075

[31] Nitin H. Vaidya. 2014. Iterative Byzantine Vector Consensus in Incomplete Graphs. In *Distributed Computing and Networking*, Mainak Chatterjee, Jian-nong Cao, Kishore Kothapalli, and Sergio Rajsbaum (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 14–28.

[32] Nitin H. Vaidya, Lewis Tseng, and Guanfeng Liang. 2012. Iterative Approximate Byzantine Consensus in Arbitrary Directed Graphs. In *Proceedings of the 2012*

ACM Symposium on Principles of Distributed Computing (PODC '12). ACM, New York, NY, USA, 365–374. https://doi.org/10.1145/2332432.2332505
[33] Yongge Wang and Yvo Desmedt. 2001. Secure Communication in Multicast Channels: The Answer to Franklin and Wright's Question. *Journal of Cryptology* 14, 2 (01 Mar 2001), 121–135. https://doi.org/10.1007/s00145-001-0002-y
[34] Douglas Brent West et al. 2001. *Introduction to graph theory*. Vol. 2. Prentice hall.
[35] H. Zhang and S. Sundaram. 2012. Robustness of information diffusion algorithms to locally bounded adversaries. In *2012 American Control Conference (ACC)*. 5855–5861. https://doi.org/10.1109/ACC.2012.6315661

A PROOFS OF SECTION 4

In the appendices, with a slight abuse of terminology, we allow a *partition* of a set to have empty parts. That is, $(Z_1, \ldots, Z_k)$ is a partition of a set Y if $\bigcup_{i=1}^{k} Z_i = Y$ and $Z_i \cap Z_j = \emptyset$ for all $i \neq j$, but some Z_i's can be possibly empty.

A Byzantine Consensus algorithm $\mathcal{A}$ outlines a procedure $\mathcal{A}_u$ for each node $u \in V$ that describes state transitions of u. In each synchronous round, each node optionally sends messages to its neighbors, receives messages from the neighbors, and then updates its state. The new state of u depends entirely on u's previous state and the messages received by u from its neighbors. The state of u determines the messages sent by u.

Lemma A.1. *If there exists a Byzantine consensus algorithm under the local broadcast model on a graph G tolerating at most f Byzantine faulty nodes, then G has minimum degree at least $2f$.*

Proof: When $f = 0$, the lemma does not impose any restrictions on G. So we assume that $f > 0$. It is easy to show that, when $n > 1$, each node must have at least one neighbor to be able to achieve consensus. So in the rest of the proof we assume that the degree of each node in G is at least 1. Suppose for the sake of contradiction that there exists a node z in G of degree less than $2f$ and there exists an algorithm $\mathcal{A}$ that solves Byzantine consensus under the local broadcast model on G. Then there exists a partition (F^1, F^2) of the neighborhood of z such that $|F^1| < f$ and $|F^2| \leq f$. Let $W = V - (F^1 \cup F^2 \cup \{z\})$ be the set of remaining nodes. Note that some of these sets can be possibly empty. However, since $n > f > 0$ and z has degree at least 1, we select these sets so that F^2 is necessarily non-empty. Recall that $\mathcal{A}$ outlines a procedure $\mathcal{A}_u$ for each node u that describes u's state transitions in each round.

We first create a network $\mathcal{G}$ to model behavior of nodes in G in three different executions E_1, E_2, and E_3, which we will describe later. Figure 2 depicts $\mathcal{G}$. The network $\mathcal{G}$ has some directed edges, the behavior of which will be explained later. We denote a directed edge from u to v as $\overrightarrow{uv}$. $\mathcal{G}$ consists of two copies of each node in W and a single copy of the remaining nodes. We denote the two sets of copies of W as W_0 and W_1. For each node $u \in W$, we denote by u_0 and u_1 the two copies of u in W_0 and W_1 respectively. For each edge $uv \in E(G)$, we create edges in $\mathcal{G}$ as follows:

1) If $u, v \in W$, then there are two edges u_0v_0 and u_1v_1 in $\mathcal{G}$. These edges are not shown in Figure 2.
2) If $u, v \in V - W$, then there is a single edge uv in $\mathcal{G}$. Some of these edges are also not shown in Figure 2.
3) If $u \in F^1$ and $v \in W$, then there are two edges uv_0 and $\overrightarrow{uv_1}$ in $\mathcal{G}$. In Figure 2, these edges are illustrated by a single undirected edge between sets F^1 and W_0, and a single directed edge from F^1 to W_1.
4) If $u \in F^2$ and $v \in W$, then there are two edges $\overrightarrow{uv_0}$ and uv_1 in $\mathcal{G}$. In Figure 2, these edges are illustrated by a single directed edge from F^2 to W_0, and a single undirected edge between sets F^2 and W_1.

Note that there are no edges between W_0 and W_1. This is emphasized in Figure 2 by drawing a cross on a dotted line between W_0 and W_1.

All message transmissions in $\mathcal{G}$ are via local broadcast, as follows. When a node u in $\mathcal{G}$ transmits a message, the following nodes receive this message identically: each node with whom u has an undirected edge and each node to whom there is an edge directed away from u. Note that a directed edge $e = \overrightarrow{uv}$ behaves differently for u and v. All messages sent by u are received by v. No message sent by v is received by u. Observe that with this behavior of directed edges, the structure of $\mathcal{G}$ ensures the following property. For each edge uv in the original graph G, each copy of u receives messages from exactly one copy of v in $\mathcal{G}$. This allows us to create an algorithm for $\mathcal{G}$ corresponding to $\mathcal{A}$ as follows. For each node

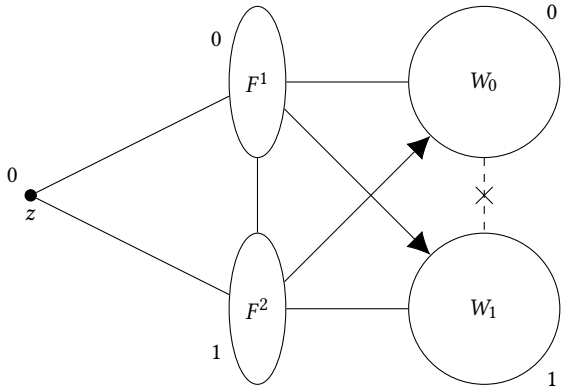


Figure 2: Network $\mathcal{G}$ to model executions E_1, E_2, and E_3. The edges in $\mathcal{G}$ are described in the text. Edges within the sets are not shown while edges between sets/nodes are depicted as single edges. The crossed dotted line between W_0 and W_1 emphasizes that there are no edges between W_0 and W_1. The numbers adjacent to the sets/nodes are the corresponding inputs in execution $\mathcal{E}$. Table 1 illustrates which nodes in $\mathcal{E}$ model the corresponding nodes in E_1, E_2, and E_3.

		(Sets of) nodes in network $\mathcal{G}$			
		z	F^1	F^2	W
Executions on G	E_1	z	F^1	F^2	W_0
	E_2	z	F^1	F^2	W_1
	E_3	z	F^1	F^2	W_1

Table 1: This table illustrates which nodes in execution $\mathcal{E}$ on network $\mathcal{G}$ (Figure 2) model the corresponding nodes in the three executions E_1, E_2, and E_3 on graph G for proof of Lemma A.1. The entries in red indicate faulty nodes in E_1, E_2, and E_3.

$u \in G$, if $\mathcal{G}$ has one copy of u, then u runs $\mathcal{A}_u$. Otherwise there are two copies u_0 and u_1 of u. Both u_0 and u_1 run $\mathcal{A}_u$.

Consider an execution $\mathcal{E}$ of the above algorithm on $\mathcal{G}$ as follows. Each node in $W_0 \cup F^1 \cup \{z\}$ has input 0 and the remaining nodes have input 1. Observe that it is not guaranteed that nodes in $\mathcal{G}$ will decide on the same value or that the algorithm will terminate. We will show that the algorithm does indeed terminate but nodes do not reach agreement in $\mathcal{G}$, which will be useful in deriving the desired contradiction. We use $\mathcal{E}$ to describe three executions E_1, E_2, and E_3 of $\mathcal{A}$ on the original graph G as follows (see also Table 1).

E_1: F^2 is the set of faulty nodes. In each round, a faulty node broadcasts the same messages as the corresponding node in $\mathcal{G}$ in execution $\mathcal{E}$ in the same round. All non-faulty nodes have input 0. Note that the behavior of non-faulty nodes in $F^1 \cup \{z\}$ and W is modelled by the corresponding (copies of) nodes in $F^1 \cup \{z\}$ and W_0, respectively, in $\mathcal{E}$. Since $\mathcal{A}$ solves Byzantine consensus on G, nodes in $W \cup F^1 \cup \{z\}$ decide on output 0 (by validity) in finite time.

E_2: F^1 is the set of faulty nodes. In each round, a faulty node broadcasts the same messages as the corresponding node in $\mathcal{G}$ in execution $\mathcal{E}$ in the same round. z has input 0 and all the remaining non-faulty nodes have input 1. Note that the behavior of non-faulty nodes in $F^2 \cup \{z\}$ and W is modelled by the corresponding (copies of) nodes in $F^2 \cup \{z\}$ and W_1, respectively, in $\mathcal{E}$. The output of the non-faulty nodes will be described later.

E_3: $F^1 \cup \{z\}$ is the set of faulty nodes. In each round, a faulty node broadcasts the same messages as the corresponding node in $\mathcal{G}$ in execution $\mathcal{E}$ in the same round. All non-faulty nodes have input 1. Note that the behavior of non-faulty nodes in F^2 and W is modelled by the corresponding (copies of) nodes in F^2 and W_1, respectively, in $\mathcal{E}$. Since $\mathcal{A}$ solves Byzantine consensus on G, nodes in $W \cup F^2$ decide on output 1 (by validity) in finite time.

Due to the output of nodes in $W \cup F^1 \cup \{z\}$ in E_1, the nodes in $W_0 \cup F^1 \cup \{z\}$ output 0 in $\mathcal{E}$. Similarly, due to the output of nodes in $W \cup F^2$ in E_3, the nodes in $W_1 \cup F^2$ output 1 in $\mathcal{E}$. It follows that in E_2, nodes in W and F^2, as modeled by W_1 and F^2 in $\mathcal{E}$, output 1 while z outputs 0. Recall that, by construction, F^2 is non-empty. This violates agreement, a contradiction. □

Lemma A.2. *If there exists a Byzantine consensus algorithm under the local broadcast model on a graph G tolerating at most f Byzantine faulty nodes, then G is $(\lfloor 3f/2 \rfloor + 1)$-connected.*

Proof: Suppose for the sake of contradiction that G is not $(\lfloor 3f/2 \rfloor + 1)$-connected and there exists an algorithm $\mathcal{A}$ that solves Byzantine consensus under the local broadcast model on G. Then there exists a vertex cut C of G of size at most $\lfloor 3f/2 \rfloor$ with a partition (A, B, C) of V such that A and B (both non-empty) are disconnected in $G - C$ (so there is no edge between a node in A and a node in B). Since $|C| \leq \lfloor 3f/2 \rfloor$, there exists a partition (C^1, C^2, C^3) of C such that $|C^1|, |C^2| \leq \lfloor f/2 \rfloor$ and $|C^3| \leq \lceil f/2 \rceil$. Recall that $\mathcal{A}$ outlines a procedure $\mathcal{A}_u$ for each node u that describes u's state transitions in each round.

Similar to the proof of Lemma A.1, we first create a network $\mathcal{G}$ to model behavior of nodes in G in three different executions E_1, E_2, and E_3, which we will describe later. $\mathcal{G}$ consists of two copies of each node in A and B, and a single copy of the remaining nodes. Figure 3 depicts $\mathcal{G}$. The edges in $\mathcal{G}$ can be deduced from Figure 3 along the lines of the proof of Lemma A.1. However, edges within C are not shown in Figure 3. They are copied exactly from G. As in proof of Lemma A.1, the structure of $\mathcal{G}$ ensures that, for each edge uv in the original graph G, each copy of u receives messages from exactly one copy of v in $\mathcal{G}$. This allows us to create an algorithm for $\mathcal{G}$ corresponding to $\mathcal{A}$ as follows. For each node $u \in G$, if $\mathcal{G}$ has one copy of u then u runs $\mathcal{A}_u$. Otherwise there are two copies u_0 and u_1 of u. Both u_0 and u_1 run $\mathcal{A}_u$.

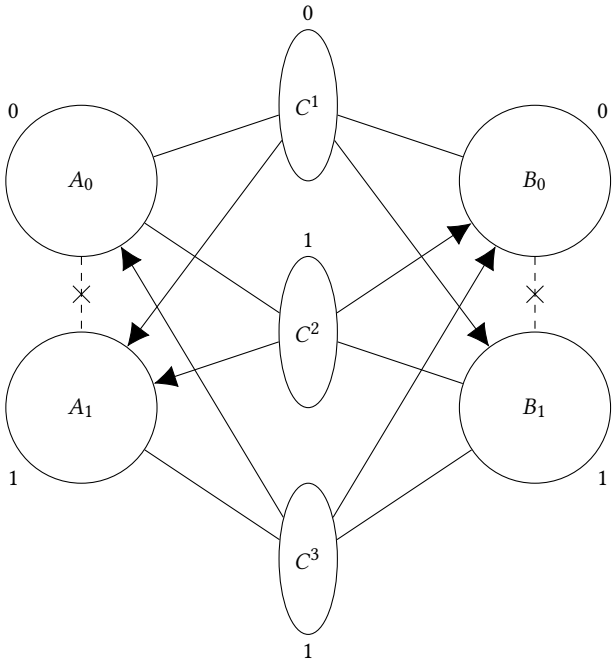


Figure 3: Network $\mathcal{G}$ to model executions E_1, E_2, and E_3. Edges within the sets are not shown while edges between sets/nodes are depicted as single edges. The crossed dotted lines emphasize that there are no edges between the corresponding sets. The numbers adjacent to the sets/nodes are the corresponding inputs in execution $\mathcal{E}$. To reduce clutter, edges within C are not shown.

Consider an execution $\mathcal{E}$ of the above algorithm on $\mathcal{G}$ as follows. Each node in A_0, B_0, and C^1 has input 0 and the remaining nodes have input 1. As in proof of Lemma A.1, we will show that the algorithm does indeed terminate but nodes do not reach agreement in $\mathcal{G}$, which will be useful in deriving the desired contradiction. We use $\mathcal{E}$ to describe three executions E_1, E_2, and E_3 of $\mathcal{A}$ on the original graph G as follows.

E_1: $C^2 \cup C^3$ is the set of faulty nodes. In each round, a faulty node in $C^2 \cup C^3$ broadcasts the same messages as the corresponding node in $\mathcal{G}$ in execution $\mathcal{E}$ in the same round. All non-faulty nodes have input 0. Note that the behavior of non-faulty nodes in A, B, and C^1 is modelled by the corresponding (copies of) nodes in A_0, B_0, and C^1, respectively, in $\mathcal{E}$. Since $\mathcal{A}$ solves

Byzantine consensus on G, nodes in A, B, and C^1 decide on output 0 (by validity) in finite time.

E_2: $C^1 \cup C^3$ is the set of faulty nodes. In each round, a faulty node in $C^1 \cup C^3$ broadcasts the same messages as the corresponding node in $\mathcal{G}$ in execution $\mathcal{E}$ in the same round. A has input 0 and all the remaining non-faulty nodes have input 1. Note that the behavior of non-faulty nodes in A, B, and C^2 is modelled by the corresponding (copies of) nodes in A_0, B_1, and C^2, respectively, in $\mathcal{E}$. The output of the non-faulty nodes will be described later.

E_3: $C^1 \cup C^2$ is the set of faulty nodes. In each round, a faulty node in $C^1 \cup C^2$ broadcasts the same messages as the corresponding node in $\mathcal{G}$ in execution $\mathcal{E}$ in the same round. All non-faulty nodes have input 1. Note that the behavior of non-faulty nodes in A, B, and C^3 is modelled by the corresponding (copies of) nodes in A_1, B_1, and C^3, respectively, in $\mathcal{E}$. Since $\mathcal{A}$ solves Byzantine consensus on G, nodes in A, B, and C^3 decide on output 1 (by validity) in finite time.

Due to the output of nodes in A, B, and C^1 in E_1, the nodes in A_0, B_0, and C^1 output 0 in $\mathcal{E}$. Similarly, due to the output of nodes in A, B, and C^3 in E_3, the nodes in A_1, B_1, and C^3 output 1 in $\mathcal{E}$. It follows that in E_2, nodes in A, as modeled by A_0 in $\mathcal{E}$, output 0 while nodes in B, as modeled by B_1, output 1. Recall that, by construction, both A and B are non-empty. This violates agreement, which is a contradiction. □

Proof of Theorem 4.1: Directly from Lemmas A.1 and A.2. □

B PROOFS OF SECTION 5

In this section, we assume that the graph G satisfies the conditions in Theorem 5. The following observation follows from the rules used for flooding.

Observation B.1. *For any phase of Algorithm 1, in step (a) for any two nodes $u, v \in V$ (possibly faulty), if v receives value b along a fault-free uv-path then u broadcast the value b to its neighbors during flooding.*

Recall that, if a faulty node does not initiate flooding, then for the purpose of the above observation, its behavior is equivalent to it flooding the value 1. We now give proofs of Lemmas 5.5, 5.2, and 5.3.

Proof of Lemma 5.5: Fix a phase of the algorithm and the corresponding set F. Consider an arbitrary non-faulty node v such that $v \in B_v$ in step (c). There are 4 cases to consider, corresponding to the 4 cases in step (c).

Case 1: $|Z_v \cap F| \leq \lfloor f/2 \rfloor$ and $|N_v| > f$. Then $A_v := N_v$ and $B_v := Z_v$. Therefore there exist at least $f+1$ nodes in A_v. Node v selects $f+1$ nodes A'_v from A_v by choosing all nodes from $A_v \cap F$ and the rest arbitrarily from $A_v - F$. Define $B'_v := B_v \cap (F - v)$. Now, $|B'_v| \leq |B_v \cap F| = |Z_v \cap F| \leq \lfloor f/2 \rfloor$ by assumption of case 1. Since G is $(\lfloor 3f/2 \rfloor + 1)$-connected, $G - B'_v$ is at least $(f+1)$-connected and there exist $f+1$ node-disjoint $A'_v v$-paths in $G - B'_v$. Furthermore, since all the nodes in $A_v \cap F$ are endpoints in these paths and $F = (A_v \cap F) \cup (B_v \cap F)$, we have that these paths exclude[1] F.

[1]Recall that a path that excludes X does not have nodes from X as internal nodes; however, nodes from X may be the endpoints of the path.

Case 2: $|Z_v \cap F| \leq \lfloor f/2 \rfloor$ and $|N_v| \leq f$. Then $A_v := Z_v$ and $B_v := N_v$. Note that when $f = 0$, this case is not possible since $v \in B_v$ and so $B_v = N_v$ must be non-empty. Since the degree of v is at least $2f$ and there are at most f nodes in B_v (including v), we have that v has at least $f+1$ neighbors in A_v. There are therefore $f+1$ node-disjoint $A_v v$-paths that have no internal nodes and hence exclude F.

Case 3: $|Z_v \cap F| > \lfloor f/2 \rfloor$ and $|Z_v| > f$. Then $A_v := Z_v$ and $B_v := N_v$. We have that

$$|N_v \cap F| = |F| - |Z_v \cap F| \leq f - \lfloor f/2 \rfloor - 1 \leq \lfloor f/2 \rfloor .$$

So this case is the same as Case 1 with the roles of Z_v and N_v swapped.

Case 4: $|Z_v \cap F| > \lfloor f/2 \rfloor$ and $|Z_v| \leq f$. Then $A_v := N_v$ and $B_v := Z_v$. From the analysis in Case 3, we have that $|N_v \cap F| \leq \lfloor f/2 \rfloor$. So this case is the same as Case 2 with the roles of Z_v and N_v swapped.

In all four cases we have that there do exist $f+1$ node-disjoint $A_v v$-paths that exclude F. □

Proof of Lemma 5.2: Fix a phase of the algorithm and the corresponding set F. For any node u, we denote the state at the beginning of the phase by γ_u^{start} and the state at the end of the phase by γ_u^{end}. Consider an arbitrary non-faulty node v, and the sets A_v and B_v in step (c). If $\gamma_v^{\text{start}} = \gamma_v^{\text{end}}$, then the claim is trivially true. So suppose $v \in B_v$ and v receives identical value along $f+1$ node-disjoint $A_v v$-paths that exclude F in step (a). Since the number of faulty nodes is at most f, thus at least one of these paths is both fault-free and has a non-faulty endpoint (other than v), say u. By Observation B.1, it follows that whatever value is received by v along this path in step (a) is the value flooded by u. Therefore, $\gamma_v^{\text{end}} = \gamma_u^{\text{start}}$, where u is a non-faulty node, as required. □

Proof of Lemma 5.3: Fix a phase of the algorithm and the corresponding set F such that all faulty nodes are contained in F. Let Z be the set of nodes that flooded 0 in step (a) of the phase and let N be the set of nodes that flooded 1 in step (a). We first show that for any non-faulty node v, $Z_v = Z$ and $N_v = N$. Consider an arbitrary node $w \in Z$ (resp. $w \in N$) that flooded 0 (resp. 1) in step (a) of this phase. Observe that P_{wv} identified in step (b) of the phase excludes F and is fault-free. Therefore, by Observation B.1, v receives 0 (resp. 1) along P_{wv} and correctly puts w in the set Z_v (resp. N_v), as required.

It follows that for any two non-faulty nodes u and v, we have that $Z_u = Z_v = Z$ and $N_u = N_v = N$. Thus $A_u = A_v$ and $B_u = B_v$. Let $A := A_u$ and $B := B_u$. First note that A is always non-empty as follows. If B is empty, then $A = V$ is non-empty. If B is non-empty, then let $w \in B$ be a node in B. By Lemma 5.5, there exist $f+1$ node-disjoint Aw-paths, which implies that $|A| \geq f+1$. Now all nodes in A flooded identical value in step (a), say α. If $u \in A$, then u's state is α at the beginning of the phase and stays unchanged in step (c). Therefore, at the end of the phase $\gamma_u = \alpha$. If $u \in B$, then observe that the $f+1$ node-disjoint Au-paths identified by u in step (c) are all fault-free. By Observation B.1, it follows that u receives α identically along these $f+1$ paths and so, at the end of the phase, $\gamma_u = \alpha$. Similarly for v, we have that $\gamma_v = \alpha$, as required. □