Byzantine Consensus with Local Multicast Channels

Muhammad Samir Khan ✉
Department of Computer Science, University of Illinois at Urbana-Champaign, IL, USA

Nitin H. Vaidya ✉
Department of Computer Science, Georgetown University, Washington, DC, USA

Abstract

Byzantine consensus is a classical problem in distributed computing. Each node in a synchronous system starts with a binary input. The goal is to reach agreement in the presence of Byzantine faulty nodes. We consider the setting where communication between nodes is modelled via an *undirected* communication graph. In the classical *point-to-point* communication model all messages sent on an edge are private between the two endpoints of the edge. This allows a faulty node to *equivocate*, i.e., lie differently to its different neighbors. Different models have been proposed in the literature that weaken equivocation. In the *local broadcast* model, every message transmitted by a node is received identically and correctly by all of its neighbors. In the *hypergraph* model, every message transmitted by a node on a hyperedge is received identically and correctly by all nodes on the hyperedge. Tight network conditions are known for each of the three cases.

We introduce a more general model that encompasses all three of these models. In the *local multicast* model, each node u has one or more local multicast channels. Each channel consists of multiple neighbors of u in the communication graph. When node u sends a message on a channel, it is received identically by all of its neighbors on the channel. For this model, we identify tight network conditions for consensus. We observe how the local multicast model reduces to each of the three models above under specific conditions. In each of the three cases, we relate our network condition to the corresponding known tight conditions. The local multicast model also encompasses other practical network models of interest that have not been explored previously, as elaborated in the paper.

2012 ACM Subject Classification Theory of computation → Distributed algorithms

Keywords and phrases Byzantine fault, distributed algorithm, consensus, broadcast, multicast

Digital Object Identifier 10.4230/LIPIcs.DISC.2021.26

Related Version *Full Version*: `https://arxiv.org/abs/2109.01205` [10]

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

1 Introduction

Byzantine consensus is a classical problem in distributed computing introduced by Lamport et al [12, 14]. There are n nodes in a synchronous system. Each node starts with a binary input. At most f of these nodes can be Byzantine faulty, i.e., exhibit arbitrary behavior. The goal of a consensus protocol is for the non-faulty nodes to reach agreement on a single output value in finite time. To exclude trivial protocols, we require that the output must be an input of some non-faulty node.

In this paper, we study consensus under the *local multicast* model. We formalize this model in Section 2. Intuitively, nodes are connected via an undirected graph G. A local multicast channel is defined by a sender and a set of receivers. Each node u may potentially serve as the sender on multiple local multicast channels. When node u sends a message on

35th International Symposium on Distributed Computing (DISC 2021).
Editor: Seth Gilbert; Article No. 26; pp. 26:1–26:16
Leibniz International Proceedings in Informatics
LIPICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

one of its local multicast channels, it is received identically and correctly by all the receivers in the channel. This model generalizes the following models that have been considered before in the literature.

1. *Point-to-point communication model:* In the classical *point-to-point* communication model, each edge uv in the communication graph represents a private link between the nodes u and v. This model is well-studied [1, 4, 12, 13, 14]. It is well-known that $n \geq 3f + 1$ and node connectivity at least $2f + 1$ are both necessary and sufficient in this model.
2. *Local broadcast model:* Recently, we [8] studied consensus under the *local broadcast* model [2, 11], where a message sent by any node is received identically by all of its neighboring nodes in the communication graph. We obtained that minimum node degree at least $2f$ and node connectivity at least $\lfloor 3f/2 \rfloor + 1$ are both necessary and sufficient for Byzantine consensus under the local broadcast model [8].
3. *Hypergraph model:* A *hypergraph* is a generalization of graphs consisting of nodes and hyperedges. Unlike an edge in a graph, a hyperedge can connect any number of nodes. For a communication network modelled as a hypergraph, a message sent by a node u on a hyperedge e (that contains u) is received identically by all nodes in the hyperedge e. Communication networks modelled as hypergraphs have been studied in the literature [6, 7, 15]. Ravikant et al [15] gave tight conditions for Byzantine consensus on $(2, 3)$-hypergraphs.[1] As discussed in Section 4, this result extends to general undirected hypergraphs as well.

The classical point-to-point communication model allows a faulty node to *equivocate* [3], i.e., send conflicting messages to its neighbors without this inconsistency being observed by the neighbors. For example, a faulty node z may tell its neighbor u that it has input 0, but tell another neighbor v that it has input 1. Since messages on each edge are private between the two endpoints, so node u does not overhear the message sent to node v and vice versa. The local broadcast model and the hypergraph model restrict a faulty node's ability to equivocate by detecting such attempts. In the local broadcast model, a faulty node's attempt to equivocate is detected by its neighboring nodes in the communication graph. In the hypergraph model, a faulty node's attempt to equivocate on a hyperedge is detected by the nodes in that hyperedge. In our local multicast model, a faulty node's attempt to equivocate on a single multicast channel is detected by the receivers in that channel.

In this work, we introduce the *local multicast* model, that unifies the models identified above, and make the following main contributions:

1. **Necessary and sufficient condition for local multicast model:** In Section 3, we present a network condition and show that it is both necessary and sufficient for Byzantine consensus under the local multicast model. The identified condition is inspired by the network conditions for directed graphs [9, 17], where node connectivity does not adequately capture the network requirements for consensus. We present a simple algorithm, inspired by [8, 9, 17].
2. **Reductions to the existing models:** The two extremes of the local multicast model are 1) each channel consists of exactly one receiver, and 2) each node has exactly one multicast channel. These correspond to the point-to-point communication model and the local broadcast model, respectively. In Section 4, we discuss how the network condition for the local multicast model reduces to the network requirements for the point-to-point

[1] i.e., each hyperedge consists of either 2 or 3 nodes.

model and the local broadcast model at the two extremes. On the other hand, if the multicast channels are induced from the hyperedges in a hypergraph, then this corresponds to the hypergraph model. In this case, the network condition reduces to the network requirements of the undirected hypergraph model given by Ravikant et al [15]. Moreover, our algorithm for the local multicast model works for all the three models identified here as well.

3. **Extensions to other models:** The local multicast model also captures some other models of practical interest (see Section 5). For instance, consider the scenario where nodes are connected via a wireless network. This can be modelled as local multicast over a graph G_1. Separately, the nodes are also connected via a bluetooth network, modelled using local multicast over a graph G_2 (with the same node set as G_1). Then the union of these networks $G_1 \cup G_2$ can be captured using the local multicast model as well. As another example, consider the scenario where nodes are connected via point-to-point channels. Additionally, nodes are also connected via a wireless network with local broadcast guarantees. As before, this can also be captured using the local multicast model. Our algorithm works for these cases as well.

In our recent work [10], we have generalized the results in this paper to the *directed* local multicast model. The directed local multicast model corresponds to directed hypergraphs where each directed hyperedge models a multicast channel with a single sender and a non-empty set of receivers. The tight condition obtained in [10] for the directed case is a natural extension of the tight condition obtained here for the undirected case. The results and proofs in [10] are more general and encompass the results in this paper.

2 System Model and Problem Formulation

We consider a synchronous system of n nodes. Nodes communicate using *local multicast channels*. Each node u has a set of multicast channels ζ_u. Each multicast channel $\chi_u \in \zeta_u$ is defined by the sender u and a non-empty set of receivers. For example, $\{v, w\} \in \zeta_u$ is a multicast channel of sender u with two receivers v and w. By convention used here, u is not included in the set of receivers. However, trivially, each node receives its own message transmissions as well. The communication between nodes is bidirectional so that if a node $v \in \chi_u$ for some channel $\chi_u \in \zeta_u$, then there exists a channel $\chi_v \in \zeta_v$ such that $u \in \chi_v$. A message m sent by a node u on a multicast channel χ_u is received identically and correctly by all nodes in χ_u. Moreover, each recepient $v \in \chi_u$ knows that m was sent by u on channel χ_u. We assume that each multicast channel is a FIFO communication channel.

The communication graph $G = (V(G), E(G))$ is an undirected graph where $V(G)$ is the set of n nodes and $uv \in E(G)$ is an edge of G if and only if there are channels χ_u and χ_v at nodes u and v, respectively, such that $u \in \chi_v$ and $v \in \chi_u$. Nodes u and v are *neighbors* in G. Observe that each multicast channel χ_u consists of a non-empty subset of the neighbors of u, such that each neighbor of u is in at least one channel in ζ_u.

- *Neighborhood:* For a set $S \subseteq V(G)$, a node $v \in V(G) - S$ is a neighbor of S if it is a neighbor of some node $u \in S$. More generally, for two disjoint sets $A, B \subseteq V(G)$, $\Gamma_G(A, B)$ defined below is the set of neighbors of B in A.

$$\Gamma_G(A,B) := \big\{u \in A \mid \exists v \in B : uv \in E(G)\big\}.$$

- *Adjacent:* For two disjoint sets $A, B \subseteq V(G)$, we use $A \rightarrow_G B$ (read as A is "adjacent" to B in G) to denote that either

(i) $B = \emptyset$, or

(ii) nodes in B have at least $f+1$ neighbors in A in the graph G, i.e.,

$$\left|\Gamma_G(A, B)\right| \geq f + 1.$$

A *Byzantine faulty* node may exhibit arbitrary behavior. In *Byzantine consensus problem* each node starts with a binary input and must output a binary value satisfying the following constraints, in the presence of up to f Byzantine faulty nodes.

1. **Agreement:** All non-faulty nodes must output the same value.
2. **Validity:** If a non-faulty node outputs $b \in \{0, 1\}$, then at least one non-faulty node must have input b.
3. **Termination:** All non-faulty nodes must decide in finite time.

It is easy to show that $f < n$ is necessary for Byzantine consensus. So we assume $f < n$ throughout the paper.

Node split

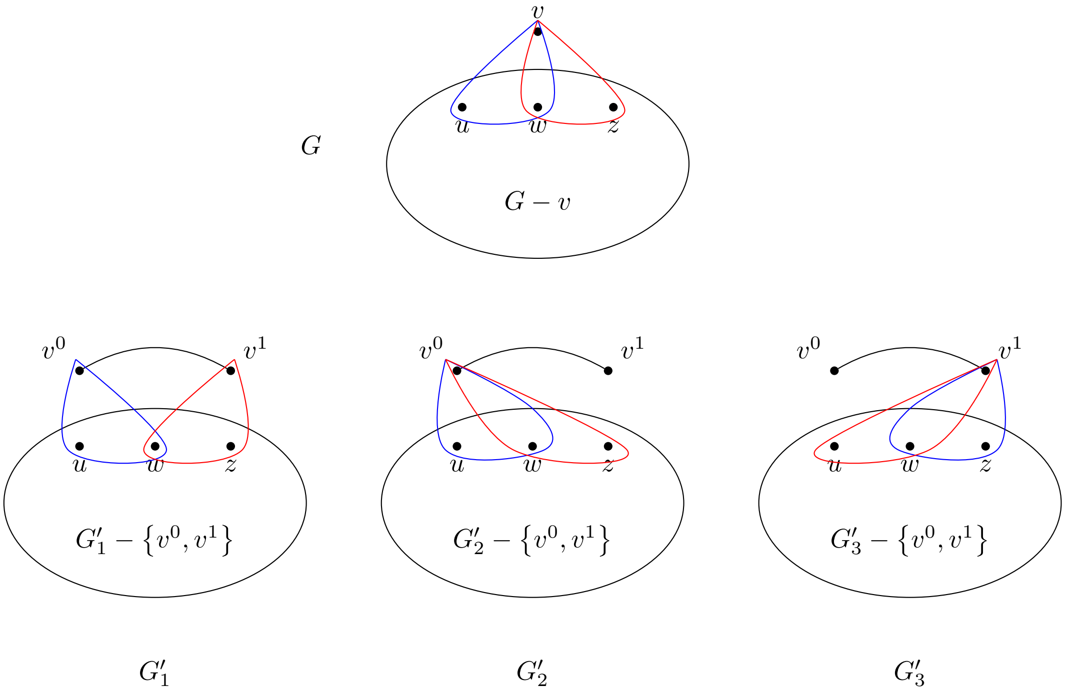
We now introduce the notion of a *node split* that is used to specify the necessary and sufficient condition under the local multicast model. As seen later, we will use the notion of node split to simulate possible equivocation by a faulty node. Intuitively, by splitting a node v, we are creating two copies of v and dividing up the channels amongst the two copies. Figure 1 shows two examples of node split. Formally, splitting a node v in G creates a new graph G' as follows.

- The node v is replaced by two nodes v^0 and v^1.
- We add an edge v^0v^1 to $E(G')$.
- We add a multicast channel $\left\{v^1\right\}$ to v^0 and a multicast channel $\left\{v^0\right\}$ to v^1.
- For every multicast channel χ_v of node v in G, choose exactly one of v^0 and v^1 as node v'. Create a multicast channel $\chi'_{v'}$ of v' with $\chi'_{v'} = \left\{u \mid u \in \chi_v\right\}$, i.e., each neighbor of v in χ_v is assigned to $\chi'_{v'}$.
- The above step adds edges to $E(G')$, of the form uv' such that $v' \in \left\{v^0, v^1\right\}$, but v' is not assigned to any multicast channel at node u. We specify these assignments as follows. Consider an edge uv', for $v' \in \left\{v^0, v^1\right\}$, in G'. For each multicast channel χ_u of node u in G, such that $v \in \chi_u$, add v' to the corresponding multicast channel χ'_u in G'. Now each neighbor w of u in G' is part of at least one multicast channel at node u.

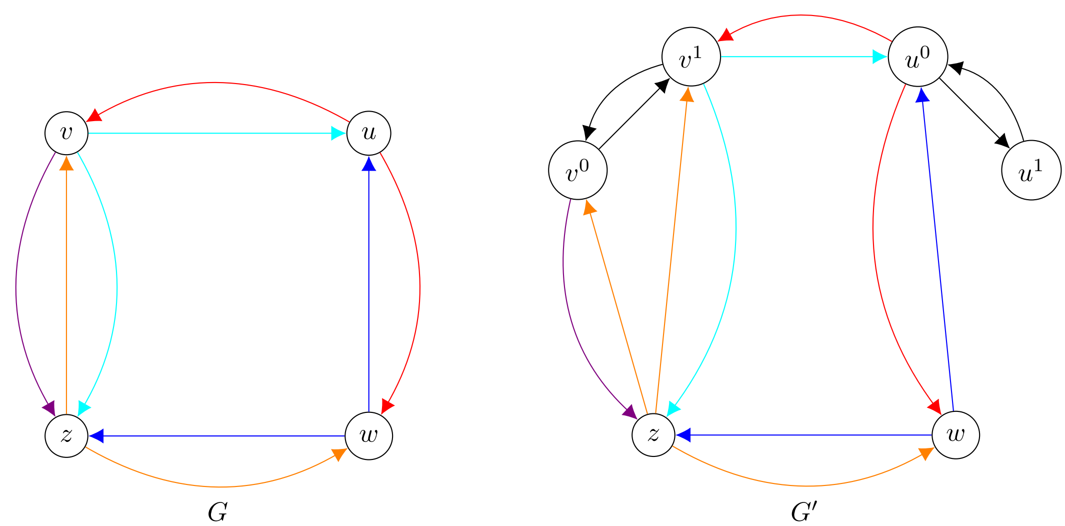
Observe that for every node $u \in V(G')$, each of its multicast channels in G' corresponds to a single multicast channel in G, except for the two channels $\left\{v^1\right\}$ and $\left\{v^0\right\}$ at nodes v^0 and v^1, respectively (where node v was split). Similarly, for every node $u \in V(G)$, each of its multicast channels in G corresponds to a single multicast channel in G'.

To split two nodes u and v in G, we first split u to obtain G'. We then split v to obtain G'' from G'. The order of splits does not matter. This process naturally extends to splitting multiple nodes as well. For a set $F \subseteq V(G)$, let $\Lambda_F(G)$ be the set of all graphs that can be obtained from G by splitting some subset of nodes in the set F. For a graph $G' \in \Lambda_F(G)$, we use F' to denote the set of nodes in G' that correspond to nodes in F in G, i.e.,

$$F' := \left(V(G') \cap F\right) \cup \left(V(G') - V(G)\right).$$



(a) Splitting a single node v. Only the channels in ζ_v are drawn here. There are two channels in ζ_v: $\{u, w\}$ and $\{w, z\}$, drawn with blue and red colors, respectively. There are three possible graphs in $\Lambda_{\{v\}}(G)$, other than G, corresponding to the assignment of channels when v is split into v^0 and v^1. These are depicted as G'_1, G'_2, and G'_3.



(b) Splitting two nodes u, v in a 4-node graph G. Directed edges of the same color, pointing out from the same sender node, represent a single channel. G' is obtained by splitting nodes u and v into u^0, u^1 and v^0, v^1, respectively. The cyan channel is assigned to v^1, the violet channel is assigned to v^0, and the red channel is assigned to u^0.

Figure 1 Examples of the node split operation.

Note that there are two choices in the node split operation above which give rise to all the graphs in $\Lambda_F(G)$:

1. choice of which nodes in F to split, and
2. assignment of multicast channels for each split node.

As needed, we will occasionally clarify these choices to specify how a graph $G' \in \Lambda_F(G)$ was constructed by splitting some nodes in F.

3 Main Result

The main result of this paper is a tight characterization of network requirements for Byzantine consensus under the local multicast model. Consider a graph $G' \in \Lambda_F(G)$ obtained from G by splitting some nodes in a set F. Recall that we use F' to denote the set of nodes in G' that correspond to nodes in F in G.

▶ **Theorem 1.** *Under the local multicast model, Byzantine consensus tolerating at most f faulty nodes is achievable on graph G if and only if for every $F \subseteq V(G)$ of size at most f, every $G' \in \Lambda_F(G)$ satisfies the following: for every partition*[2] *(L, C, R) of $V(G')$, either*

1. *$L \cup C \rightarrow_{G'} R - F'$, or*
2. *$R \cup C \rightarrow_{G'} L - F'$.*

While we allow a partition to have empty parts, the interesting partitions are those where both L and R are non-empty, but C can be possibly empty. In Section 4, we show that when the local multicast model corresponds to the point-to-point, local broadcast, or hypergraph model, the above condition reduces to the corresponding known tight network conditions in each of the three cases.

We prove the necessity of Theorem 1 in Section 6. In Section 7, we give an algorithm to constructively show the sufficiency. The above condition is similar to the network condition for directed graphs in the point-to-point communication model [16, 17] and in the local broadcast model [9]. Note that [9, 16, 17] deal with consensus on arbitrary *directed* graphs, where connectivity constraints do not adequately capture the tight network requirements. In this paper, we are interested in undirected graphs. However, since the local multicast model is quite general and captures various models with different connectivity requirements, it is plausible that no concise network connectivity property will be able to properly characterize the tight condition.

For convenience, we give a name to the condition in Theorem 1.

▶ **Definition 2.** *A graph G satisfies* condition LCR with parameter F *if for every $G' \in \Lambda_F(G)$ and every partition (L, C, R) of $V(G')$, we have that either*

1. *$L \cup C \rightarrow_{G'} R - F'$, or*
2. *$R \cup C \rightarrow_{G'} L - F'$.*

We say that G satisfies condition LCR, *if G satisfies condition LCR with parameter F for every set $F \subseteq V(G)$ of cardinality at most f.*

[2] with a slight abuse of terminology, we allow a partition of a set to have empty parts.

4 Reductions to Other Models

In this section, we discuss how condition LCR relates to the tight conditions for the classical point-to-point communication model, the local broadcast model, and the hypergraph model.

Point-to-piont Channels

The classical point-to-point communication model corresponds to the case where each multicast channel in the graph consists of a single receiver node, so that the communication on an edge uv is private between the two nodes u and v. Under the point-to-point communication model, it is well known that $n \geq 3f+1$ [5, 12, 14] and node connectivity at least $2f+1$ [4, 5] are both necessary and sufficient for consensus on arbitrary undirected graphs.

When G has only point-to-point channels, i.e., each multicast channel consists of a single receiver node, then G satisfies condition LCR if and only if $n \geq 3f+1$ and G has node connectivity $\geq 2f+1$. We prove this formally in [10]. Therefore, the two models are equivalent when only point-to-point channels are present.

Local Broadcast

The local broadcast model corresponds to the other extreme where each node in the graph has exactly one multicast channel, so that the messages transmitted by a node u are received identically and correctly by all neighbors of u. Under the local broadcast model, we [8] showed that node degree at least $2f$ and connectivity at least $\lfloor 3f/2 \rfloor + 1$ are both necessary and sufficient for consensus on arbitrary undirected graphs.

When G has only local broadcast channels, i.e., each node has a single multicast channel, then G satisfies condition LCR if and only if G has minimum node degree $\geq 2f$ and node connectivity $\geq \lfloor 3f/2 \rfloor + 1$. We prove this formally in [10]. Therefore, the two models are equivalent when only local broadcast channels are present.

Hypergraphs

The last model we consider in this section is the hypergraph model. In a hypergraph $H = (V(H), E(H))$, each hyperedge $e \in E(H)$ is a subset of nodes $e \subseteq V(H)$. A hyperedge $e \in E(H)$ is called an $|e|$-hyperedge. Each hyperedge is effectively a multicast channel, i.e., a message sent by a node u on an edge $e \supseteq \{u\}$ is received identically and correctly by all nodes $v \in e$. However, any node on a hyperedge can act as a sender for this channel. In our local multicast model with communication graph G, this corresponds to the case where, for every pair of nodes u, v and multicast channel χ_u of u such that $v \in \chi_u$, there exists a channel χ_v of v such that $\chi_v = \big(\chi_u \cup \{u\}\big) - v$.

Ravikant et al. [15] obtained tight conditions for the hypergraph model. We observe that while the conditions were presented as a tight characterization for $(2,3)$-hypergraphs[3] in [15], they also hold for general hypergraphs. In our local multicast model, when the communication graph G and its local multicast channels correspond to an undirected hypergraph, then condition LCR reduces to the tight conditions for hypergraphs given in [15]. The formal proof is given in [10].

[3] H is a $(2,3)$-hypergraph if each hyperedge is either a 2-hyperedge or a 3-hyperedge.

5 Application to New Models

As mentioned in Section 1, the local multicast model also encompasses some other network models of practical interest that, to the best of our knowledge, have not been considered before in the literature. Suppose the n nodes are connected via a local multicast network with graph G_1. For example, network connectivity in G_1 can be via point-to-point links or via wireless channels modelled as local broadcast. Additionally, the n nodes are connected via another local multicast network with graph G_2. For example, G_2 may correspond to a wireless network with different frequencies and/or technologies. The complete system, where nodes can communicate on channels in G_1 as well as on channels in G_2, can also be characterized by the local multicast model. We omit details for brevity, but this corresponds to the natural union of G_1 and G_2, with each node now having access to its multicast channels in G_1 as well as its multicast channels in G_2.

6 Necessity of Condition in Theorem 1

Intuitively, consider a set $F \subseteq V(G)$ of size at most f, such that the graph G violates condition LCR with parameter F. With F as a candidate faulty set, the splitting of nodes in F captures possible equivocation by nodes in F: a faulty node can behave as if it has input 0 on some of its multicast channels and behave as if it has input 1 on the other multicast channels. Let $G' \in \Lambda_F(G)$ be a graph obtained by splitting nodes in F. We use F' to denote the nodes in G' that correspond to nodes in F in G. Suppose (L, C, R) is a partition of G'. Now consider the execution where non-faulty nodes in L have input 0. Since $R \cup C \not\rightarrow_{G'} L - F'$, nodes in $L - F'$ can not distinguish between F and its neighbors in $R \cup C$, i.e., $\Gamma_{G'}(R \cup C, L - F')$ as the set of faulty nodes. So non-faulty nodes in L are stuck with outputting 0 in this case. Similarly if non-faulty nodes in R have input 1, then they have no choice but to output 1, creating the desired contradiction.

A formal necessity proof is given in [10] for the *directed* local multicast model, which generalizes the *undirected* local multicast model considered in this paper. It follows the standard state machine based approach [1, 4, 5], similar to [9, 17]. Suppose there exists a set $F \subseteq V(G)$, of size at most f, such that G does not satisfy condition LCR with parameter F, but there exists an algorithm $\mathcal{A}$ that solves consensus on G. Algorithm $\mathcal{A}$ outlines a procedure $\mathcal{A}_u$ for each node u that describes u's state transitions, as well as messages transmitted on each channel of u in each round. Now there exists a graph $G' \in \Lambda_F(G)$ and a partition of $V(G')$ that does not satisfy the requirements of condition LCR. To create the required contradiction, we work with an algorithm for G' instead of $\mathcal{A}$. To see why this works, observe that an algorithm $\mathcal{A}$ on graph G can be adapted to create an algorithm $\mathcal{A}'$ for a graph $G' \in \Lambda_F(G)$ as follows. Each round i in the algorithm $\mathcal{A}$ is now split into two sub-rounds $i(a)$ and $i(b)$ in $\mathcal{A}'$. We consider each of these rounds separately and specify the corresponding steps for each node in G' for the algorithm $\mathcal{A}'$.

- *Round $i(a)$:* Each node $v \in V(G') \cap V(G)$ that was not split runs $\mathcal{A}_v$ as specified for round i. For a node $v \in V(G') - V(G)$ that was split into $v^0, v^1 \in V(G')$, both v^0 and v^1 run $\mathcal{A}_v$ for round i with the following modification. Consider a multicast channel $\chi_v \in \zeta_v$ of node v in G. Let χ'_{v^0} (resp. χ'_{v^1}) be the corresonding multicast channel in G' at node v^0 (resp. v^1). If the algorithm $\mathcal{A}_v$ wants to transmit a message on χ_v, then v^0 (resp. v^1) sends the message on χ'_{v^0} (resp. χ'_{v^1}), while v^1 (resp. v^0) ignores this message transmission. Observe that, for any node $u \in \chi_v$, u receives messages on the channel from exactly one of v^0 and v^1.

- *Round $i(b)$:* This round is reserved for the split nodes. Consider a node $v \in F - V(G')$ that was split into $v^0, v^1 \in V(G')$. Node v^0 forwards all messages it received in round $i(a)$ to v^1 and v^1 forwards all messages it received in round $i(a)$ to v^0. This allows both v^0 and v^1 to run $\mathcal{A}_v$ in the next round.

Now, $\mathcal{A}'$ might not solve consensus on G', or may not even terminate. However, as long as care is taken with regards to which nodes are allowed to be faulty in G' and the input of the split nodes, the guarantees for $\mathcal{A}$ will imply that $\mathcal{A}'$ does indeed terminate and solve consensus on G'. In particular, we want that

1. the faulty nodes in G' correspond to at most f nodes in G,
2. for each node $v \in F - V(G')$ that was split into $v^0, v^1 \in V(G')$, either
   a. both v^0 and v^1 have the same input, or
   b. at least one of v^0 and v^1 is faulty.

So for necessity, it is enough to show that no algorithm exists for a hypergraph $G' \in \Lambda_F(G)$, under the two conditions identified above. We formalize this property and use it in the formal necessity proof in [10].

7 Algorithm for the Local Multicast Model

To prove the sufficiency portion of Theorem 1, we work with a different network condition, which we will be equivalent to condition LCR. We first introduce some notation that is used in the algorithm. For a set of nodes $U \subseteq V(G)$, we use $G[U]$ to denote the subgraph induced by the nodes in U. The multicast channels in $G[U]$ are obtained from the multicast channels in G by removing nodes in $V(G) - U$ from each channel, with some channels possibly being deleted entirely. We use $G - U$ to denote the subgraph $G[V(G) - U]$.

A *path* is a sequence of distinct nodes such that if u precedes v in the sequence, then v is a neighbor of u in G (i.e., uv is an edge). For a path P and node z, we use $z \bullet P$ to denote the path obtained by prefixing the node z to P.

- *uv-paths:* For two nodes $u, v \in V(G)$, a uv-path P_{uv} is a path from u to v. u is called the *source* and v the *terminal* of P_{uv}. Any other node in P_{uv} is called an *internal* node of P_{uv}. Two uv-paths are *node-disjoint* if they do not share a common internal node.
- *Uv-paths:* For a set $U \subset V(G)$ 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 terminal. Two Uv-paths are *node-disjoint* if they do not have any nodes in common except the terminal node v. In particular, two node-disjoint Uv-paths have different source nodes. By definition, the number of disjoint Uv-paths is upper bounded by the size of the set U. Note the difference in definition between node-disjoint uv-paths and node-disjoint Uv-paths when $U = \{u\}$ is a singleton set. The former requires only internal nodes to be different, while the latter needs to have different source nodes as well. For the former, there can be more than one such node-disjoint path, while for the latter, there is at most one.
- *Propagate:* For two disjoint node sets $A, B \subseteq V(G)$, we use $A \rightsquigarrow_G B$ (read as A "propagates" to B in G) to denote that either
  (i) $B = \emptyset$, or
  (ii) for every $v \in B$, there exist at least $f+1$ node-disjoint Av-paths in the graph $G[A \cup B]$.

We now give a different network condition which is equivalent to condition LCR, but will be useful for specifying an algorithm for the local multicast model and proving its correctness. Recall that we use F' to denote the set of nodes in G' corresponding to nodes in F in G.

▶ **Definition 3.** *A graph G satisfies* condition AB with parameter F *if for every $G' \in \Lambda_F(G)$ and every partition (A, B) of $V(G')$, we have that either*

1. $A \rightsquigarrow_{G'} B - F'$, *or*
2. $B \rightsquigarrow_{G'} A - F'$.

We say that G satisfies condition AB, *if G satisfies condition AB with parameter F for every set $F \subseteq V$ of cardinality at most f.*

▶ **Theorem 4.** *A graph G satisfies condition LCR if and only if G satisfies condition AB.*

We skip the proof of Theorem 4, which is (almost) identical to proof of Theorem 5.2 in [9]. We show the sufficiency of condition AB (and hence condition LCR) constructively. For the rest of this section, we assume that G satisfies condition AB. The proposed algorithm is given in Algorithm 1. It draws inspiration from algorithms in [8, 9, 17]. Each node v maintains a binary state variable γ_v, which we call v's γ value. Each node v initializes γ_v to be its input value.

The nodes use "flooding" to communicate with the rest of the nodes. We refer the reader to [8, 9] for details about the flooding primitive. Briefly, when a node u wants to flood a binary value $b \in \{0, 1\}$, it transmits b to all of its neighbors, who forward it to their neighbors, and so forth. If a node u receives a message on channel χ, then u appends the channel id of χ when fowarding the message to its neighbors. By adding some simple sanity checks, one can assume that even a faulty node v does indeed transmit some value when it is v's turn to forward a message. In at most n synchronous rounds, the value b will be "flooded" in G. However, faulty nodes may tamper messages when forwarding, so some nodes may receive a value $\bar{b} \neq b$ along paths that contain faulty nodes.

The algorithm proceeds in phases. Every iteration of the `for` loop (starting at line 3) is a phase numbered $1, \ldots, 2^f$. Let F^* denote the actual set of faulty nodes. Each iteration of the for loop, i.e. phase > 0, considers a candidate faulty set F. In this iteration, nodes attempt to reach consensus, by updating their γ state variables, assuming the candidate set F is indeed faulty. Let Z and N be the set of nodes in $G - F$ that have their state variable set to 0 and 1, respectively, at the beginning of the iteration. Each iteration has three steps.

- In `step (a)`, each node v floods its γ_v value.
- In `step (b)`, based on the values received during flooding, each node v creates its estimate of the sets Z and N, by ignoring all paths that pass through the candidate faulty set F, i.e., paths that have internal nodes from F. This estimate is created in a manner so that
  1. when $F \neq F^*$, this estimate may be incorrect, but
  2. when $F = F^*$, this estimate is indeed correct.
- In `step (c)`, based on the estimates created in `step (b)`, a node v may update its γ_v value. The rules for updates ensure that
  1. when $F \neq F^*$, for each non-faulty node v, its state γ_v at the end of the iteration equals the γ value of some non-faulty node at the beginning of the iteration (Lemma 5).
  2. when $F = F^*$, all non-faulty nodes have identical γ values at the end of this iteration (Lemma 6).

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

Each node has a binary input value in $\{0, 1\}$.
Each node v maintains a binary state $\gamma_v \in \{0, 1\}$, initialized to the input value of v.
For *each* $F \subseteq V$ *such that* $|F| \leq f$ **:**

  `Step (a):` Flood value γ_v.
  if $v \in F$ **then** skip steps (b) and (c)

  `Step (b):`
  Create a graph G'_v by splitting all nodes in F as follows. Set

$$F' := \left\{u^0 \mid u \in F\right\} \cup \left\{u^1 \mid u \in F\right\} \quad \text{and} \quad V(G'_v) := (V(G) - F) \cup F'.$$

  The edges and channels of G'_v are as determined by the split operation, with the following choices: **For** *every node* $z \in F$ *and a multicast channel* $\chi_z \in \zeta_z$ **:**
    if $\exists w \in \chi_z$ *such that* $w \in V(G) - F$ **then**
      identify a single wv-path P_{wv} in $G - F$ (Lemma 7).
      if *v received 0 from z along the path $z \bullet P_{wv}$ in* `step (a)`*, such that the initial message was sent by node z on channel χ_z* **then** assign χ_z to z^0.
      else assign χ_z to z^1.
    else
      assign χ_z to z^1.

  For each node $u \in V - F$, identify a single uv-path P_{uv} in $G - F$ (Lemma 7). Note that path P_{vv} trivially exists (P_{vv} contains only v). Initialize a partition (Z_v, N_v) of $V(G'_v)$ as follows,

$$Z_v := \{u^0 \mid u \in F\} \cup \left\{u \in V - F \mid v \text{ received value 0 along } P_{uv} \text{ in } \texttt{step (a)}\right\},$$
$$N_v := \{u^1 \mid u \in F\} \cup (V - F - Z_v).$$

  `Step (c):`
  if $Z_v \rightsquigarrow_{G'_v} N_v - F$ **then** set $A_v := Z_v$ and $B_v := N_v$
  else set $A_v := N_v$ and $B_v := Z_v$

  if $v \in B_v - F$ **then**
    `// by construction, the paths of interest in` G `naturally correspond to paths in` G'_v`.`
    if *in* `step (a)`*, v received a value $\delta \in \{0, 1\}$ identically along any $f + 1$ node-disjoint $A_v v$-paths in the graph $G'_v[A \cup (B - F)] = G'_v - (B_v \cap F)$* **then**
      set $\gamma_v := \delta$

Output γ_v

At the end, after all iterations of the main `for` loop, each output node v outputs its γ_v value.

The correctness of Algorithm 1 relies on the following two key lemmas, which are proven in Section A. Recall that we use F^* to denote the actual set of faulty nodes.

▶ **Lemma 5.** *For a non-faulty node $v \in V - F^*$, its state γ_v at the end of any given phase of Algorithm 1 equals the state of some non-faulty node at the start of that phase.*

▶ **Lemma 6.** *Consider a phase > 0 of Algorithm 1 wherein $F = F^*$. At the end of this phase, every pair of non-faulty nodes $u, v \in V - F^*$ have identical state, i.e., $\gamma_u = \gamma_v$.*

Lemma 5 ensures validity, i.e., that the output of each non-faulty node is an input of some non-fautly node. It also ensures that agreement among non-faulty nodes, once acheived, is not lost. Lemma 6 ensures that agreement is reached in at least one phase of the algorithm. These two lemmas imply correctness of Algorithm 1 as shown in Section A.

8 Conclusion

In this paper, we introduced the local multicast model which, to the best our knowledge, has not been studied before in the literature. The local multicast model encompasses the point-to-point, local broadcast, and hypergraph communication models, as well as some new models which have not been considered before. We identified a tight network condition for Byzantine consensus under the local multicast model, along the lines of [9, 17], and proved its necessity and sufficiency. When the local multicast model represents one of point-to-point, local broadcast, or hypergraph communication models, we showed how the identified network condition reduces to the known tight requirements for the corresponding case.

A natural extension to complete the local multicast model is to consider a *directed* communication graph, which corresponds to *directed* hypergraphs, and generalizes the directed cases of point-to-point and local broadcast models. In our recent work [10], we have extended the results in this paper to the directed setting. The natural extension of condition LCR to the directed case is the tight network condition for directed local multicast. We refer the reader to [10] for more details.

References

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

2 Vartika Bhandari and Nitin H. Vaidya. On reliable broadcast in a radio network. In *Proceedings of the Twenty-fourth Annual ACM Symposium on Principles of Distributed Computing*, PODC '05, pages 138–147, New York, NY, USA, 2005. ACM. doi:10.1145/1073814.1073841.

3 Byung-Gon Chun, Petros Maniatis, Scott Shenker, and John Kubiatowicz. Attested append-only memory: Making adversaries stick to their word. *SIGOPS Oper. Syst. Rev.*, 41(6):189–204, 2007. doi:10.1145/1323293.1294280.

4 Danny Dolev. The byzantine generals strike again. *Journal of Algorithms*, 3(1):14–30, 1982. doi:10.1016/0196-6774(82)90004-9.

5 Michael J. Fischer, Nancy A. Lynch, and Michael Merritt. Easy impossibility proofs for distributed consensus problems. *Distributed Computing*, 1(1):26–39, March 1986. doi:10.1007/BF01843568.

6 Mattias Fitzi and Ueli Maurer. From partial consistency to global broadcast. In *Proceedings of the Thirty-second Annual ACM Symposium on Theory of Computing*, STOC '00, pages 494–503, New York, NY, USA, 2000. ACM. doi:10.1145/335305.335363.

7 Alexander Jaffe, Thomas Moscibroda, and Siddhartha Sen. On the price of equivocation in byzantine agreement. In *Proceedings of the 2012 ACM Symposium on Principles of Distributed Computing*, PODC '12, pages 309–318, New York, NY, USA, 2012. ACM. `doi:10.1145/2332432.2332491`.

8 Muhammad Samir Khan, Syed Shalan Naqvi, and Nitin H. Vaidya. Exact Byzantine Consensus on Undirected Graphs under Local Broadcast Model. In *Proceedings of the 2019 ACM Symposium on Principles of Distributed Computing*, PODC '19, page 327–336, New York, NY, USA, 2019. Association for Computing Machinery. `doi:10.1145/3293611.3331619`.

9 Muhammad Samir Khan, Lewis Tseng, and Nitin H. Vaidya. Exact Byzantine Consensus on Arbitrary Directed Graphs Under Local Broadcast Model. In *23rd International Conference on Principles of Distributed Systems (OPODIS 2019)*, volume 153 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 30:1–30:16, Dagstuhl, Germany, 2020. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik. `doi:10.4230/LIPIcs.OPODIS.2019.30`.

10 Muhammad Samir Khan and Nitin H. Vaidya. Byzantine consensus under directed hypergraphs. *arXiv*, 2021. `arXiv:2109.01205`.

11 Chiu-Yuen Koo. Broadcast in radio networks tolerating byzantine adversarial behavior. In *Proceedings of the Twenty-third Annual ACM Symposium on Principles of Distributed Computing*, PODC '04, pages 275–282, New York, NY, USA, 2004. ACM. `doi:10.1145/1011767.1011807`.

12 Leslie Lamport, Robert Shostak, and Marshall Pease. The byzantine generals problem. *ACM Trans. Program. Lang. Syst.*, 4(3):382–401, 1982. `doi:10.1145/357172.357176`.

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

14 M. Pease, R. Shostak, and L. Lamport. Reaching agreement in the presence of faults. *J. ACM*, 27(2):228–234, 1980. `doi:10.1145/322186.322188`.

15 D. V. S. Ravikant, V. Muthuramakrishnan, V. Srikanth, K. Srinathan, and C. Pandu Rangan. On byzantine agreement over (2,3)-uniform hypergraphs. In *Distributed Computing*, pages 450–464, Berlin, Heidelberg, 2004. Springer Berlin Heidelberg.

16 Lewis Tseng and Nitin Vaidya. Exact byzantine consensus in directed graphs. *arXiv preprint arXiv:1208.5075*, 2014. `arXiv:1208.5075`.

17 Lewis Tseng and Nitin H. Vaidya. Fault-tolerant consensus in directed graphs. In *Proceedings of the 2015 ACM Symposium on Principles of Distributed Computing*, PODC '15, page 451–460, New York, NY, USA, 2015. Association for Computing Machinery. `doi:10.1145/2767386.2767399`.

A Proof of Correctness of Algorithm 1

In this section, we show correctness of Algorithm 1 when the communication graph G sastisfies condition AB. For the rest of this section, we assume that G satisfies condition AB. Throughout this section, we use F^* to denote the actual set of faulty nodes. We first prove Lemma 5.

Proof of Lemma 5. Fix a phase > 0. Note that a node updates its state only in `step (c)`. Suppose a node v updates its state γ_v to α. Then, as per the update rules in `step (c)`, v must have received the value α identically along $f+1$ node-disjoint $A_v v$-paths in `step (a)`. Since there are at most f faulty nodes, so at least one of these paths, say P, must have neither any faulty internal node nor a faulty source node. Since α was received along P, which has only non-faulty internal nodes, so the source node of P, say u, flooded α in `step (a)` of this phase. Since u is non-faulty, so γ_u had value α at the start of this phase. Therefore, the state of node v at the end of this phase equals the state of a non-faulty node u at the start of this phase. ◀

Before proving Lemma 9, we need some intermediate results. We first show that in every iteration of the main `for` loop, the paths in `step (b)` do exist.

▶ **Lemma 7.** *In any phase > 0 of the algorithm with a candidate faulty set F, for any two nodes $u, v \in V(G) - F$, there exists a uv-path in $G - F$.*

Proof. Suppose for the sake of contradiction that there exist two nodes $u, v \in V(G) - F$ such that there is no uv-path in $G - F$. Let A be the set of nodes that are reachable by node u in $G - F$, and let $B = V - A$. Note that

(i) $|F| \leq f$,

(ii) $u \in A = A - F$ so that $A - F \neq \emptyset$, and

(iii) $v \in B - F$ so that $B - F \neq \emptyset$.

Now, there are no edges between A and $B - F$. Since $|F| \leq f$, so there are at most f node-disjoint Av-paths and at most f node-disjoint Bu-paths in graph G. Therefore, we have

1. $A \not\rightarrow_G B - F$, and
2. $B \not\rightarrow_G A - F$.

Since $G \in \Lambda_F(G)$, so condition AB is violated, a contradiction. ◀

When a non-faulty node wants to flood a value $b \in \{0, 1\}$, it sends a single value b on all of its multicast channels. But a faulty node might send different messages on different channels. Note however, that even a faulty node must send the exact same value on a single multicast channel.

▶ **Lemma 8.** *Consider a phase > 0 of Algorithm 1 wherein $F = F^*$. For any two non-faulty nodes $u, v \in V(G) - F^*$, we have $G'_u = G'_v$ in `step (b)` of this phase. Furthermore, if in `step (a)` of this phase, faulty node $z \in F^*$ transmitted 0 (resp. 1) on one of its channels $\chi_z \in \zeta_z$, such that $\chi_z - F^*$ is non-empty, then in `step (b)` of this phase χ_z is assigned to z^0 (resp. z^1) in $G'_u = G'_v$.*

Proof. Consider the phase where $F = F^*$ and any two non-faulty nodes $u, v \in V(G) - F^*$. Observe that the node set of the two graphs G'_u and G'_v are the same. For the edges and channels, by construction, it is sufficient to show that for any $z \in F^*$, the assignment of multicast channels to z^0 and z^1 in the split operation is the same in G'_u as in G'_v. Consider an arbitrary node $z \in F^*$ and a multicast channel $\chi_z \in \zeta_z$ at node z. There are two cases to consider:

- **Case 1:** There exists a node $w \in \chi_z$ such that $w \in V(G) - F^*$.
  Let w be any arbitrary such node. By Lemma 7, there exists a wu-path (resp. wv-path) in $G - F^*$. Let P_{wu} (resp. P_{wv}) be any arbitrary wu-path (resp. wv-path) identified by u (resp. v) in line 9. Note that P_{wu} (resp. P_{wv}) does not contain *any* faulty nodes. Therefore, a message transmitted by z on χ_z, is received by u (resp. u) along $z \bullet P_{wu}$ (resp. $z \bullet P_{wv}$) untampered. Therefore, in `step (a)`, if z transmitted 0 on channel χ_z, then u (resp. v) received value 0 from z along $z \bullet P_{wu}$ (resp. $z \bullet P_{wv}$). So, in line 11, node u (resp. node v) assigns χ_z to z^0 in G'_u (resp. G'_v). Similarly, if z transmitted 1 on channel χ_z in `step (a)`, then both u and v assign χ_z to z^1 in G'_u and G'_v, respectively.
- **Case 2:** There does not exist any node $w \in \chi_z$ such that $w \in V(G) - F^*$.
  In this case, in line 13, both u and v assign χ_z to z^1 in G'_u and G'_v, respectively.

In both cases, we have that the multicast channel χ_z was assigned identically by both u and v. As shown in Case 1, if z transmitted 0 (resp. 1) on χ_z and $\chi_z - F^*$ is non-empty, then χ_z was assigned to z^0 (resp. z^1) by both u and v, as required. ◀

▶ **Lemma 9.** *Consider a phase > 0 of Algorithm 1 wherein $F = F^*$. Let*

$$Z := \left\{u^0 \mid u \in F\right\} \cup \left\{w \in V(G) - F^* \mid w \text{ flooded value } 0 \text{ in } \texttt{step (a)} \text{ of this phase}\right\}$$
$$N := \left\{u^1 \mid u \in F\right\} \cup \left\{w \in V(G) - F^* \mid w \text{ flooded value } 1 \text{ in } \texttt{step (a)} \text{ of this phase}\right\}.$$

For any two non-faulty nodes $u, v \in V(G) - F^*$, we have $Z_u = Z_v$ and $N_u = N_v$ in `step (b)` *of this phase.*

Proof. Consider the phase where $F = F^*$ and any two non-faulty nodes $u, v \in V(G) - F^*$. We show that $Z \subseteq Z_v$ and $N \subseteq N_v$ (resp. $Z \subseteq Z_u$ and $N \subseteq N_u$). Since $Z \cup N = Z_u \cup N_u = Z_v \cup N_v$, it follows that $Z = Z_u = Z_v$ and $N = N_u = N_v$. For a node $w \in F^*$, the two split nodes w^0 and w^1 are assigned identically by both u and v. So consider an arbitrary node $w \in V(G) - F^* = (Z \cup N) - \left\{u^0, u^1 \mid u \in F^*\right\}$. Recall that we are considering the phase > 0 of the algorithm where $F = F^*$ is the actual set of faulty nodes. There are two cases to consider:

- **Case 1:** $w \in Z - \left\{u^0 \mid u \in F^*\right\}$, i.e., $w \notin F^*$ flooded 0 in `step (a)` of this phase.
  Let P_{wv} be the wv-path identified by v in `step (b)`. Note that P_{wv} is contained entirely in $G - F^*$ so that P_{wv} does not have any faulty nodes. It follows that, in `step (a)`, since w flooded value 0 so v received value 0 along P_{wv}. Therefore, in `step (b)`, v puts w in the set Z_v.
- **Case 2:** $w \in N - \left\{u^1 \mid u \in F^*\right\}$, i.e., $w \notin F^*$ flooded 1 in `step (a)` of this phase.
  Let P_{wv} be the wv-path identified by v in `step (b)`. Note that P_{wv} is contained entirely in $G - F^*$ so that P_{wv} does not have any faulty nodes. It follows that, in `step (a)`, since w flooded value 1 so v received value 1 along P_{wv}. Therefore, in `step (b)`, v puts w in the set N_v.

So we have that $Z \subseteq Z_v$ and $N \subseteq N_v$, as required. A symmetric argument gives $Z \subseteq Z_u$ and $N \subseteq N_u$. As argued before, this implies that $Z = Z_u = Z_v$ and $N = N_u = N_v$. ◀

We are now ready to prove Lemma 6.

Proof of Lemma 6. Consider the phase where $F = F^*$. Suppose $u, v \in V(G) - F^*$ are any two non-faulty nodes. By Lemma 9, we have $Z = Z_u = Z_v$ and $N = N_u = N_v$, where Z and N are as in the statement of Lemma 9. By Lemma 8, we have $G'_u = G'_v$, Let $G' = G'_u = G'_v$. We use F' to denote the set of nodes in G' corresponding to nodes in F^* in G.

We now show that all non-faulty nodes in $V(G) - F^*$ have identical state at the end of this phase. Consider `step (c)` of this phase. If either $Z - F^*$ or $N - F^*$ is empty, then all non-faulty nodes have identical state at the start of the phase and they do not update their state in `step (c)`. So suppose that both $Z - F^*$ and $N - F^*$ are non-empty. Observe that, at the start of `step (c)`, all nodes in $Z - F^*$ have identical state of 0, while all nodes in $N - F^*$ have identical state of 1. We show that in `step (c)` either all nodes in $Z - F^*$ update their state to 1, or all nodes in $N - F^*$ update their state to 0.

Note that $G' \in \Lambda_{F^*}(G)$. By condition AB, either $Z \rightsquigarrow_{G'} N - F'$ or $N \rightsquigarrow_{G'} Z - F'$. We consider each case as follows.

- **Case 1:** $Z \rightsquigarrow_{G'} N - F'$.
  Consider an arbitrary node $v \in (Z \cup N) - F'$. In `step (c)`, v sets $A_v = Z$ and $B_v = N$. If $v \in A_v - F' = Z - F'$, then v has state 0 at the start of this phase and does not update it in `step (c)`. So suppose that $v \in B_v - F' = N - F'$. Now, if in `step (a)` v received the value 0 identically along some $f+1$ node-disjoint Zv-paths in $G' - (N \cap F')$, then v sets $\gamma_v = 0$ in `step (c)`. We show that such $f+1$ node-disjoint Zv-paths do indeed exist. Since $Z \rightsquigarrow_{G'} N - F'$, so there exist $f+1$ node-disjoint Zv-paths in $G' - (N \cap F')$. Without loss of generality, only the source nodes on these paths are from Z. For each such path, observe that only the source node, say $z \in Z$, can be faulty. If the source node z is faulty, then by Lemma 8, and construction of G' and Z, z sent the value 0 on the first channel on this path in `step (a)`. If z is non-faulty, then by construction of Z, z flooded value 0 in `step (a)`. Now all other nodes on the path are non-faulty, so v received value 0 along this path in `step (a)`. Therefore, v received value 0 identically along the $f+1$ node-disjoint Zv-paths in `step (a)`, as required.
- **Case 2:** $Z \not\rightsquigarrow_{G'} N - F'$ so that $N \rightsquigarrow_{G'} Z - F'$ by condition AB.
  Consider an arbitrary node $v \in (Z \cup N) - F'$. In `step (c)`, v sets $A_v = N$ and $B_v = Z$. If $v \in A_v - F' = N - F'$, then v has state 1 at the start of this phase and does not update it in `step (c)`. So suppose that $v \in B_v - F' = Z - F'$. As in Case 1, since $N \rightsquigarrow_{G'} Z - F'$, so there exist $f+1$ node-disjoint Nv-paths in $G' - (Z \cap F')$ such that v received the value 1 identically along these paths in `step (a)`. Therefore, v sets $\gamma_v = 1$ in `step (c)`, as required.

In both of the cases, all non-faulty nodes have identical state at the end of this phase, as required. ◀

Using Lemmas 5 and 6, we can now prove the sufficiency of condition AB. Recall that by Theorem 4, condition AB is equivalent to condition LCR. Thus this shows the reverse direction of Theorem 1.

Proof of Theorem 1 ($\Leftarrow$ direction). Algorithm 1 satisfies the *termination* condition because it terminates in finite time.

In one of the iterations of the main `for` loop, we have $F = F^*$, i.e., F is the actual set of faulty nodes. By Lemma 6, all non-faulty nodes have the same state at the end of this phase. By Lemma 5, these states remain unchanged in any subsequent phases. Therefore, all nodes output an identical state. So the algorithm satisfies the *agreement* condition.

At the start of phase 1, the state of each non-faulty node equals its own input. By inductively applying Lemma 5, we have that the state of a non-faulty node always equals the *input* of some non-fautly node, including in the last phase of the algorithm. So the output of each non-faulty node is an input of some non-faulty node, satisfying the *validity* condition. ◀