

© 2018 Syed Shalan Naqvi

EXACT BYZANTINE CONSENSUS UNDER LOCAL-BROADCAST CHANNELS

BY

SYED SHALAN NAQVI

THESIS

Submitted in partial fulfillment of the requirements
for the degree of Master of Science in Computer Science
in the Graduate College of the
University of Illinois at Urbana-Champaign, 2018

Urbana, Illinois

Adviser:

Professor Nitin Vaidya

Abstract

We consider the problem of achieving exact consensus with Byzantine faults under a local-broadcast communication channel. We prove necessary and sufficient conditions on the underlying communication graph to achieve consensus. We show that under this model consensus is possible on undirected graphs that have $2f + 1$ nodes and are $2f$ -connected. In contrast, it is well known that with point-to-point links, achieving consensus requires at least $3f + 1$ nodes and $2f + 1$ connectivity. We show a tight result for the case of a single fault, by proving that consensus is impossible on any undirected graph that has at most 1 connectivity, and providing an algorithm for 2-connected graphs with at least 3 nodes. We give another algorithm for achieving consensus with at most f faulty nodes, on arbitrary undirected graphs with $2f$ connectivity and $2f + 1$ nodes. Additionally, we prove that consensus is impossible on any graph with connectivity less than $f + 1$. We also show some necessity results for directed graphs. Finally, we present an example network that suggests that connectivity less than $2f$ may be sufficient in general.

To my parents, for their love and support.

Acknowledgments

I want to thank my advisor, Professor Nitin Vaidya, for his guidance and mentorship. I would also like to thank Muhammad Samir for the valuable discussions and feedback.

Table of Contents

Chapter 1	Introduction	1
1.1	Related Work	1
1.2	System Model	2
Chapter 2	Some Necessity Results	3
Chapter 3	A Tight Bound for $f=1$	6
3.1	Algorithm (for $f=1$)	6
3.2	Correctness	9
Chapter 4	A sufficiency result for arbitrary f	18
4.1	Algorithm	18
4.2	Correctness	22
Chapter 5	Other Results	34
Chapter 6	Conclusion	35
References	36

Chapter 1: Introduction

We consider the problem of achieving exact Byzantine consensus in a synchronous system under a local-broadcast channel. By local-broadcast we mean that whenever a node sends a message, it is received by all of its outgoing neighbors. This model is motivated by wireless networks. Up to f of the nodes may deviate from the algorithm and behave in an arbitrary manner. We prove necessary and sufficient conditions on the underlying communication graph to achieve exact consensus in this model. Under the point-to-point model, it is known that $2f + 1$ connectivity and $3f + 1$ nodes are both necessary and sufficient to achieve consensus in undirected graphs with up to f Byzantine faulty nodes [1]. Under the assumption of non-equivocation, which means that faulty nodes are not allowed to send conflicting messages to different nodes, consensus can be solved with point-to-point links on a complete graph with $2f + 1$ nodes [2]. Because of a local-broadcast channel, our model allows us to not only achieve non-equivocation, but furthermore, it also allows neighbors to detect when a faulty node tampers a message before forwarding it. We show that $2f$ connectivity and $2f + 1$ nodes are sufficient for achieving consensus in undirected graphs under this model, if the nodes have binary-valued inputs. An algorithm for the special case of a single faulty node (i.e $f = 1$) is given in **Chapter 3**. **Chapter 4** extends this to arbitrary values of f . Both algorithms assume undirected graphs and FIFO links. **Chapter 2** describes some necessary conditions on the communication graphs. These results do not make the same assumptions.

1.1 RELATED WORK

The problem of consensus has been studied under many different assumptions and fault-models. Fischer et al. proved that under f Byzantine faults with point-to-point links, consensus is impossible on any undirected graph that has less than $3f + 1$ nodes or is at most $2f$ -connected [1]. The problem of consensus on directed graphs was studied by Tseng et al. [3], who showed necessary and sufficiency results for achieving crash-tolerant and Byzantine consensus on directed graphs. Some of the lemmas and notation in **Chapter 3** are motivated by their results. Byzantine consensus using partial authentication is studied in [4]. [5] discussed the problem of reliable global broadcast in radio networks. The model is similar to ours, in that a message sent by a node is received by all neighbors, but we discuss the problem of Byzantine consensus. Solving approximate consensus under malicious faults using iterative algorithms is discussed in [6]. Malicious faulty nodes are defined as

nodes that can update their state arbitrarily. This fault-model is also similar to our model, because malicious faulty nodes cannot send different values to different neighbors, a property we achieve by using a local-broadcast channel. The property of not sending conflicting messages to different neighbors is called non-equivocation in the literature. Consensus with non-equivocation is also studied in [7], but in the context of $(2, 3)$ uniform hypergraphs, i.e hypergraphs having edges of size 2 and 3 only. [8] shows bounds on the number of 3-hyperedges required to achieve consensus. [9] studies the problem of iterative approximate Byzantine consensus using 3-hyperedges in which a unique sender is identified. [10] studies asynchronous multiparty computation under non-equivocation. [2] discusses asynchronous Byzantine consensus with non-equivocation and transferable authentication.

1.2 SYSTEM MODEL

Formally, we model the communication network using a graph $G = (V, E)$, such that there is an edge from node u to node v , if node v can receive messages from node u . Each node has a binary input value. Each node u maintains a variable d_u , that indicates the “decision” of u . When a node sets d_u to a non-null value b , it is said to “decide” the value b . We assume that communication is synchronous and that any time a node u sends a message, it is received by all outgoing neighbors of u . We will also assume that nodes receive their own messages. Furthermore, when a node u receives a message from a neighbor node, it can identify the neighbor that sent the message. The links are assumed to be FIFO. Up to f of the nodes in the graph are allowed to be Byzantine faulty. A faulty node is assumed to have complete knowledge of the states of all the nodes, the algorithm, and the network topology. An algorithm is said to achieve consensus on graph G , if the following conditions hold

- **Termination** : The algorithm terminates after a bounded number of steps.
- **Agreement** : When the algorithm terminates, all non-faulty nodes decide the same value d .
- **Validity** : If the input value of all the non-faulty nodes is b , then when the algorithm terminates, all non-faulty nodes decide b .

Solving consensus when the graph has only one node is trivial. Hence, we will assume that $|V| \geq 2$.

Chapter 2: Some Necessity Results

In this chapter, we discuss necessary conditions on the communication graphs of networks that allow for exact consensus under f faulty nodes. The arguments are similar in flavor to necessity arguments for Byzantine faults with point-to-point links, but they are complicated by the fact that we have a local-broadcast channel. So that when faulty nodes deviate from the algorithm, non-faulty nodes may realize this and update their state accordingly. These nodes may also tell other nodes about the faulty nodes. So arguments that work for the point-to-point model do not necessarily apply here. The strategy is to have faulty nodes behave as a non-faulty node would under a different execution, so that without sufficient connectivity constraints, it may be impossible for some subset A of the nodes to distinguish between faulty and non-faulty nodes. Nodes in A then have to decide a value with incomplete information about the rest of the nodes. We exploit this fact to establish constraints on the communication graph.

Let $G = (V, E)$ be a directed graph and let A, B be two disjoint subsets of V . We will use the notation A^- to denote the incoming neighbors of set A , outside of A . That is, $A^- = \{(b, a) : a \in A, b \in V - A\}$. Similarly we will use the notation $B \xrightarrow{x} A$ to denote that there are at least x incoming neighbors of A in B . That is, $|A^- \cap B| \geq x$. When this is not the case, we will write $B \not\xrightarrow{x} A$. This notation is adopted from [3]. Then we have the following result.

Lemma 2.1. *In a synchronous system with up to f faulty nodes, in any partition L, C, R of the nodes, where L and R are non-empty, either $L \cup C \xrightarrow{f+1} R$ or $R \cup C \xrightarrow{f+1} L$*

Proof. The proof is by contradiction. Suppose there exists a partition L, C, R of the communication graph such that neither $L \cup C \xrightarrow{f+1} R$ nor $R \cup C \xrightarrow{f+1} L$, but there is an algorithm A that achieves consensus on this graph. Consider the following executions of algorithm A

E₁ : Suppose all the nodes in L have input 1 and all nodes in C and R have input 0. Suppose all nodes are non-faulty. Since we have assumed that consensus is achievable and this is a deterministic system, there is a fixed value $v \in \{0, 1\}$ that all the nodes decide on.

E₂ : Suppose all non-faulty nodes have input 0 and the nodes in $R^- \cap (L \cup C)$ are faulty. Note that this is possible since $L \cup C \not\xrightarrow{f+1} R$. These faulty neighbors of R send the

same messages to R as in E_1 . The input value of nodes in R is also the same as in E_1 , so E_2 and E_1 are indistinguishable to nodes in R . Hence, nodes in R must decide v in E_2 as well.

E₃ : Suppose all non-faulty nodes have input 1 and the nodes in $L^- \cap (R \cup C)$ are faulty. Again, this is possible since $R \cup C \stackrel{f+1}{\not\Rightarrow} L$. These faulty neighbors of L send the same messages to L as in E_1 . The input value of nodes in L is also the same as in the E_1 , so E_3 and E_1 are indistinguishable to nodes in L . Hence, nodes in L must decide v in E_3 as well.

Note that the validity constraint in E_2 implies that v should be 0, and the validity constraint in E_3 implies that v should be 1. Since v can not be both these values, we conclude that consensus is impossible in E_1 . □

Corollary 2.2. *If there are f faulty nodes, then consensus is impossible for any undirected graph that is not at least $(f + 1)$ -connected.*

Proof. Consider a graph G that is not $(f + 1)$ -connected. Then, by definition, there exists a set $C \subseteq V$ of size at most f whose removal disconnects the graph. Let L and R be two components of the disconnected graph. Then, since $|C| \leq f$, $L \cup C \stackrel{f+1}{\not\Rightarrow} R$ and $R \cup C \stackrel{f+1}{\not\Rightarrow} L$. Hence, by **Lemma 2.1**, consensus is impossible on graph G . □

Next we will prove that this condition is not sufficient to achieve consensus in the presence of f faulty nodes. This is because even with local-broadcast, each node must have at least $2f$ incoming neighbors to achieve consensus. The overall idea for this is as follows. Suppose there is a node u with at most $2f - 1$ neighbors $v_1 \dots v_{2f-1}$. Consider an execution in which u has input value 0 and every other non-faulty node has input value 1. Suppose $v_1 \dots v_{2f-1}$ are faulty and behave as if everyone has input value 0. u should not be able to distinguish this execution from an execution where every non-faulty node had input value 0 and $v_f \dots v_{2f-1}$ were faulty. Hence, u must decide on 0, and in order to achieve agreement, every non-faulty node must decide on 0. But now consider an execution with exactly the same messages, except that u was also a faulty node (in addition to $v_1 \dots v_{2f-1}$). The decision in this execution would also be 0, which violates validity. A formal proof is given below.

Lemma 2.3. *In any synchronous system that achieves consensus while tolerating up to f faulty nodes, each node must have $2f$ incoming neighbors.*

Proof. The proof is by contradiction. Suppose there is an algorithm A that solves consensus on a directed graph $G = (V, E)$ that has a node u with less than $2f$ incoming edges. Let

the incoming neighbors of u be $v_1 \dots v_{f-1}, v_f \dots v_{2f-1}$. We will construct three executions, E_1, E_2, E_3 .

E₁ : In execution E_1 , $v_1 \dots v_{f-1}$ are faulty nodes. u has input 0 and all other non-faulty nodes have input 1. The behavior of the faulty nodes is as follows. The faulty nodes run a simulation E_2 of algorithm A on a graph G' . If v was a node in graph G , then we will denote the corresponding node in G' as v' . In the simulation every non-faulty node has input 0, and nodes $v'_f \dots v'_{2f-1}$ are faulty. In E_1 (the actual execution) whenever a node $v_k \in v_f \dots v_{2f-1}$ sends a message, the faulty node in the simulation, v'_k sends the same message. Note that this is possible since v'_k is a faulty node and it can send any arbitrary message. In E_2 (the simulated execution) whenever a node $v'_k \in v'_1 \dots v'_{f-1}$ sends a message, the faulty node v_k in E_1 sends the same message. Again this is possible since v_k is a faulty node in E_1 . Let the decision in E_1 be d .

E₂ : Execution E_2 is the simulated execution defined above. Note that this is a valid execution, since every non-faulty node is running algorithm A . Also recall that in E_2 , all non-faulty nodes had input 0. Node u receives the same messages from each of its neighbors in both the executions. Hence, u 's decision would be the same in both the execution. By the validity constraint in E_2 , we have $d = 0$.

E₃ : In execution E_3 , in addition to $v_1 \dots v_{f-1}$, u is also a faulty node. All non-faulty nodes have inputs 1. Node u behaves exactly the same as in E_1 . The faulty nodes, $v_1 \dots v_{f-1}$, also behave the same way as in E_1 . Hence, the messages received by any non-faulty node remain the same as in E_1 . The inputs of the non-faulty nodes are also the same as in E_1 . By the validity constraint in E_3 , the decision should be 1.

Since E_1 and E_2 are indistinguishable to node u , by the validity condition in E_2 , we have that the decision in E_1 has to be 0. However execution E_3 and execution E_1 are indistinguishable to all the other non-faulty nodes. By the validity condition in E_3 , we have that the decision in E_1 has to be 1. Both these statements can not be true. Hence, we conclude that consensus is impossible on the graph G .

□

Chapter 3: A Tight Bound for $f=1$

In this chapter, we will describe the necessary and sufficient conditions for achieving consensus on an undirected graph in the case of a single fault. From **Corollary 2.2**, we know that for $f = 1$, consensus is impossible on any graph that is not 2-connected. Here we will prove that this bound is tight. That is, consensus is achievable on any arbitrary 2-connected, undirected graph, with at least 3 nodes. The proof is constructive, and an algorithm for achieving consensus in this case is given below.

3.1 ALGORITHM (FOR $F=1$)

Intuitively, the algorithm works as follows. Initially, each node only knows its own input value. Then nodes alternate between exchanging the sets of values they know and reporting faulty behavior. The key idea here is that the local-broadcast channel allows some of the faulty behavior to be detected by certain nodes. Specifically, if the faulty node w tampers a message before forwarding it, the neighbor u who sent w that message can observe the tampering, because of the local-broadcast channel. Node u can then help other nodes learn that node w is faulty. Narrowing down the list of potentially faulty nodes helps, because a node u can trust a message it receives over a path that it knows does not contain the faulty node. If node u does not know anything about which node might be faulty, then u needs to receive the message over $f + 1 = 2$ node disjoint paths to be able to trust the message. Similarly, if u sends a message to v along path P , and the faulty node w tampers this message, then the neighbor x of w , who forwarded u 's message to w , can observe the tampering and relay this information back to u , along path P . So node u can learn that the faulty node must be on the subpath of P from w to u . Since nodes u and v have at least two disjoint paths, P and Q , between them, if u 's message to v is tampered along path P , then u learns that path Q from v to u is fault-free. Hence, we can at least establish reliable communication in one direction. This helps because now node v can tell node u its state. Details of the algorithm are given below.

Recall that $G = (V, E)$ is the underlying, undirected communication graph. We will assume that $|V| = n \geq 3$ and that G is 2-connected. Each node $u \in V$ keeps the following state

- An array val_u to store the input values of nodes in the graph. Initially, $val_u[u]$ is set to the input value of node u and all other elements of val_u are set to the null value,

\perp . We will say that node **u knows the input value of node v**, if $val_u[v] \neq \perp$.

- A set F_u of nodes that are potentially faulty from u 's viewpoint. F_u is initialized to $V - \{u\}$.
- A set I_u of node ids. If a node v is in the set I_u and node u receives any message m originated by v , then node u does not use m in updating its state. This will be clarified again later. I_u is initialized to be the empty set.
- An array $K[u]$, where $K[u][v]$ is the set of nodes whose input value may be known to node v , from u 's perspective. Initially $K[u][u] = \{u\}$, and for all $v \neq u$, u optimistically sets $K[u][v] = V$. We will use the shorthand K_u for $K[u][u]$.

Definition 3.1. We will use the phrase “node u reliably receives a message m from node v ” if any of the following conditions hold

- u receives message m directly from a neighbor node v (i.e. $(u, v) \in E$).
- u identically receives message m sent by v through $f + 1 = 2$ node-disjoint paths.
- u receives a message from v through a path disjoint with F_u .

Flooding

When a node u wants to flood a message m , it sends the pair $(m, [])$ to all of its neighbors, and we say that u *originates* the message m . m is the actual content of the message and the second element represents the path the message has travelled on so far ($[]$ denotes the empty path). If nodes u and v are neighbors, then when u sends a message (m, P) , we will assume v receives the message $(m, P + [u])$. Since a node can identify which of its neighbors sent the message, this is achievable by having node v append u on to the path P . Whenever a node v receives $(m, path)$ from a node u for the first time¹, then v possibly updates its own state (val_v , F_v and $K[v]$) and then *forwards* $(m, path)$ to the rest of its neighbors². Any received message (m, P) , where the length of the path P is greater than the number of nodes is discarded. These rules ensure that the flooding process terminates after a bounded number of steps. Otherwise, the faulty node could keep sending bogus messages and cause the flooding process to never terminate.

¹If two or more messages are received in a given round with the same path, then only the first message received is forwarded.

²When the distinction between *originating* a message and *forwarding* is not important, we will use the term, *sends*, to refer to both acts.

Note that although the *path* part of the message is an array, we will also treat it as a set whenever it is convenient.

The algorithm proceeds in phases. In phase 0, nodes flood their input values. If the faulty node tampers some of the messages, then not all the nodes will reliably receive all the values. Phase 1 has two parts. In phase 1.1, nodes try to propagate the values they know to other nodes by flooding their *val* array. Again, not all of these values will be reliably received. So in phase 1.2, nodes that know which node is faulty try to tell this to other nodes. Phase 1 is repeated $n(n - 1) + 1$ times. Each iteration of Phase 1 is called a *round* of Phase 1, which consists of a round of phase 1.1 and a round of phase 1.2. Finally in the decision phase, the nodes decide on a common value, based on the information collected in the previous phases. We now describe the algorithm for an arbitrary node u .

Phase 0

- Node u floods its own input value a , and sets $val_u[u] = a$.
- If u reliably receives a value b from a node v , then u sets $val_u[v] = b$.
- If a neighbor w of u receives a message from u and forwards it incorrectly (i.e. w tampers the message before forwarding, or does not forward it at all), then u sets $F_u = \{w\}$.

We will assume that all nodes send their input values in phase 0. Since there could be a faulty node in the graph, this property is not guaranteed. However, it can be achieved by having the neighbors of the faulty node assume a default value in case they do not receive a message from a faulty node.

Phase 1 is repeated $n(n - 1) + 1$ times.

Phase 1

- **Phase 1.1**
 - Node u floods val_u .
 - If a neighbor w of u receives a message from u and forwards it incorrectly, then u sets $F_u = \{w\}$.
- **Phase 1.2**

1. If F_u is a singleton $\{w\}$, then u floods a message saying that node w is faulty. If node u receives a message (m, P) , saying that node v is faulty, then u sets $F_u = F_u \cap (P \cup \{v\})$. Recall P is the *alleged* path that the message travelled on, before it reached u .
2. If a neighbor w of u receives a message from u during flooding and forwards it incorrectly, then u sets $F_u = \{w\}$.
3. For a node $v \notin (F_u \cup I_u)$, if u reliably received val_v in phase 1.1 (with respect to the current value of F_u), then u sets $K[u][v] = \{x \mid val_v[x] \neq \perp\}$, and “merges” val_u and val_v . That is, for all x such that $val_u[x] = \perp$, u sets $val_u[x] = val_v[x]$.
4. For a node $v \notin (F_u \cup I_u)$, if u *did not* reliably received val_v in phase 1.1 (with respect to the current value of F_u), then u sets $I_u = I_u \cup \{v\}$.
5. Node u sets $I_u = I_u \cup F_u$
6. Finally, node u sets $K[u][u] = \{x \mid val_u[x] \neq \perp\}$, and for any node $v \in I_u$, u sets $K[u][v] = V$.

Decision Phase

Node u computes the “decision set”, $D_u = \cap_{v \in (V - F_u)} K[u][v]$. Consider the multi-set $S = \{val_u[i] \mid i \in D_u\}$ of values of the nodes in the decision set. Note that S cannot contain \perp , since $D_u \subseteq K[u][u]$. Node u decides the majority value (the value that appears most frequently) in S . If there is a tie, then 0 is chosen as the decision value (recall that the inputs are binary).

3.2 CORRECTNESS

The algorithm described above trivially satisfies termination, since the algorithm runs a bounded number of steps.

Proving validity and agreement needs more work. Through the lemmas below, we will prove that **(a)** the decision set for each node is non-empty, **(b)** each node computes the same decision set, and **(c)** the computed decision set contains at least $2f + 1 = 3$ elements.

When talking about a path subgraph P of G , we will use the notation $[u, v]_P$ to denote the path between node u and v (including u and v) in P . If the end points are not included, then we will denote this by parentheses, e.g. $(u, v)_P$ or $[u, v)_P$, or $(u, v)_P$. As an example if $P = a - b - c - d - e$, then

$$(a, c)_P = b \quad (3.1)$$

$$[a, c)_P = a - b \quad (3.2)$$

$$[a, c]_P = a - b - c \quad (3.3)$$

Definition 3.2. We will say that “node u knows a fault-free uv path” at a certain time, if there is a uv path that is disjoint with F_u at that time.

The decision set can be thought of as computing the minimum information that is known to all the non-faulty nodes. The idea is that if all the nodes decide based on that information, then they will agree on the same value. Note that in computing D_u , node u considers only nodes that are not in F_u . Hence, we first need to prove that if w is a faulty node, then in the decision phase, w is still a member of F_u .

Lemma 3.3. If w is the faulty node, then for any non-faulty node v , w remains in F_v until the algorithm terminates.

Proof. Consider a non-faulty node v . We will prove the claim by induction on the number of updates made to F_v . The base case (0 updates) is trivial, since at the start of the algorithm, $F_v = V - \{v\}$. Since by assumption v is non-faulty, if there is a faulty node w , then $w \in V - \{v\} = F_v$.

Now let us assume that after k updates to F_v (for some $k \in \mathbb{N}$), $w \in F_v$. For the $(k + 1)$ th update to F_v there are two cases, we will prove correctness for both of them.

- *When u is a neighbor of v and u does not correctly forward a message received from v :* Since all messages are sent by flooding, if v sends a message to u which it expects to be forwarded and u does not forward this message correctly, then u must be a faulty node. Since we are assuming there is only one faulty node, we have $u = w$. Hence, the $(k + 1)$ -th update, $F_v = \{u\}$, is correct.
- *v receives a message (m, P) that says u is faulty :* There are two cases, either the faulty node w is on path P or it is not. In the first case the faulty node is in P , and in the second case the faulty node must be u , i.e. $u = w$. This is because P does not contain any faulty nodes, so a non-faulty node x must have originated the message m . A non-faulty node x only originates such a message if u is a neighbor of x and does not correctly forward a message received from x 's. As argued previously, u must be the faulty node in this case.

Therefore, in either case, the faulty node w is in the set $P \cup \{u\}$. Furthermore, by the induction hypothesis we have, $w \in F_v$. Hence, the $(k + 1)$ -th update, $F_v = F_v \cap (P \cup \{u\})$, is correct.

Hence, after the $(k + 1)$ th update, we still have $w \in F_v$. \square

Our next two results show how the local-broadcast model helps in narrowing down the list of possibly faulty nodes and figuring out fault-free paths.

Lemma 3.4. *If node w is faulty and it tampers a message (before forwarding it) in phase 0, then at the end of the first round of phase 1, for any two non-faulty nodes u, v , at least one of the following three conditions is true.*

- u knows that v is non-faulty (i.e. $v \notin F_u$) and knows a fault-free uv path.
- v knows that u is non-faulty (i.e. $u \notin F_v$) and knows a fault-free uv path.
- u knows that v is non-faulty and v knows that u is non-faulty (i.e. $v \notin F_u$ and $u \notin F_v$).

Proof. Suppose that the faulty node w tampers a message it received from a neighbor x in phase 0. Note that since there is only one faulty node (i.e. $f = 1$), x must be non-faulty. Hence, by the end of phase 0, we will have $F_x = \{w\}$. By **Lemma 3.3**, w remains in set F_x in the first round of phase 1.2. Furthermore, since no rule for updating the set F_x makes the set larger, in the first round of phase 1.2, we still have, $F_x = \{w\}$. Therefore, in round 1 of phase 1.2, x sends a message saying that w is faulty. Now consider any two non-faulty nodes u, v . Since the graph is 2-connected, either u and v must be neighbors, or there must be two internally disjoint paths between u and v [11]. In either case, there must be one uv path, P , that does not contain the faulty node w . Again, because the graph is 2-connected, there is a path, P_x that starts at x , terminates at a node in P , and does not include w .

Consider the first node y on P_x that is also in P (it is possible that $x = y$). Let P' be the sub-path of $[x, y]_{P_x}$. We consider three cases.

- *If $y = u$:* In this case, u receives a message from x along path P' that says node w is faulty. Therefore, u sets $F_u = F_u \cap (P' \cup \{w\})$. Since $(P' \cup \{w\}) \cap (u, v]_P = \emptyset$, we have $F_u \cap (u, v]_P = \emptyset$. Hence, u learns that v is non-faulty and that path P is a fault-free uv path.
- *If $y = v$:* Similar to the above case, here v learns that u is non-faulty and that path P is a fault-free uv path.

- If $y \neq u$ and $y \neq v$: In this case, u receives a message from path $[y, u]_P \cup P'$ saying that w is faulty, and v receives a message from path $[y, v]_P \cup P'$ saying that w is faulty. Note that u and v are the end-points of P , and by construction $P \cap P' = \emptyset$. Hence, $v \notin [y, u]_P \cup P'$ and $u \notin [y, v]_P \cup P'$. Therefore both u and v learn that the other node is non-faulty.

Figure 3.1 illustrates the three different cases. □

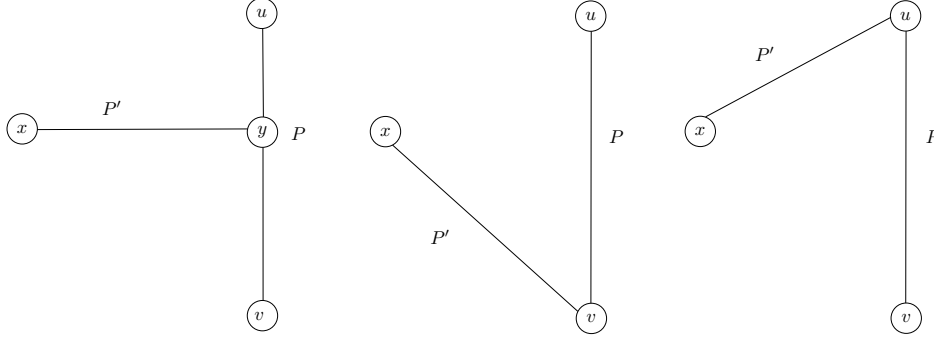


Figure 3.1: Possible paths from node x

Let $val_{u,r}$ denote the state of the array val_u at the end of round r of phase 1, and $val_{u,0}$ be the state of val_u at the end of phase 0. Analogously define $I_{v,r}, F_{v,r}, K_{u,r}$ and $K[u][v]_r$.

Lemma 3.5. *Suppose there is a faulty node w that tampers a message in phase 0. Let u, v be two non-faulty nodes and suppose there exists a smallest integer r , such that $v \in I_{u,r}$. Then the following three conditions hold.*

- $u \notin F_{v,r}$
- There is a uv path Q , such that $Q \cap F_{v,r} = \emptyset$
- For all rounds $1 \leq r' \leq n(n-1) + 1$ of phase 1, $u \notin I_{v,r'}$.

Proof. We consider two cases.

- If $v \in F_{u,r}$: Note that since there are no rules for updating F_u that make the set larger, if $v \in F_{u,r}$, then $v \in F_{u,1}$. Hence, by **Lemma 3.4**, $u \notin F_{v,1}$ and there is a uv path Q , such that $Q \cap F_{v,1} = \emptyset$. Again, since there are no rules that make the set F_v larger, therefore, $u \notin F_{v,r}$ and $Q \cap F_{v,r} = \emptyset$. Hence, for all rounds $1 \leq r' \leq n(n-1) + 1$, v reliably receives $val_{u,r'-1}$ from u , and $u \notin I_{v,r'}$.

- If $v \notin F_{u,r}$: In this case, since r is the smallest integer such that $v \in I_{u,r}$, then it must be the case that u does not reliably receive $val_{v,r-1}$ in round r of phase 1. This means, u receives conflicting messages from v on two disjoint paths P and Q in round r of phase 1.1. WLOG suppose w is on path P and tampers the message sent by v . In this case v receives a message from a subpath $(w,v)_P$ of P in round r of phase 1.2, saying that w is faulty. Hence, v sets $F_{v,r} = F_{v,r} \cap (\{w\} \cup (w,v)_P)$ and therefore $u \notin F_{v,r}$, and $F_{v,r} \cap Q = \emptyset$. So what is left to show is that $u \notin I_{v,r'}$ for any round $1 \leq r' \leq n(n-1) + 1$.

Since there are no rules for updating the set F_v that make the set larger, we have $u \notin F_{v,r'}$ and $F_{v,r'} \cap Q = \emptyset$, for any round $r' \geq r$, and hence v reliably receives $val_{u,r'-1}$ along path Q in round r' .

Furthermore, $u \notin F_{v,1}$, because otherwise by **Lemma 3.4**, $v \notin F_{u,1}$ and u knows a fault-free path from v , which means u reliably receives $val_{v,r-1}$ in round r , which contradicts our assumption. Similarly if there is a round $r' < r$ in which v does not reliably receive $val_{u,r'-1}$, then u learns a fault-free uv path in round r' . Therefore, again, it must be the case that u reliably receives $val_{v,r-1}$ in round r , which contradicts our assumption. Hence, $u \notin F_{v,1}$ and v reliably receives val_u in every round of phase 1, and hence, $1 \leq r' \leq n(n-1) + 1$, $u \notin I_{v,r'}$.

□

Next we need to show that the intersection D_u computed by every node is non-empty. For this we need to show that for every node u , there is a non-empty set S that is contained in all the sets $K[u][v]$ (where $v \notin F_u$). In fact, we will show the much stronger statement that the collection $\{K[u][v] \mid v \notin F_u\}$ is totally ordered by the subset relation.

We will use the shorthand K_u for $K[u][u]$. First we show the following lemma, which we will need later on.

Lemma 3.6. *Suppose there exists a round $1 \leq r \leq n(n-1) + 1$ of phase 1, such that for all non-faulty nodes u , $K_{u,r} = K_{u,r-1}$. Then for any round $r \leq r' \leq n(n-1) + 1$, for all non-faulty nodes u , $K_{u,r'} = K_{u,r-1}$.*

Proof. The statement is vacuously true if $r = n(n-1) + 1$. So let us assume $r < n(n-1) + 1$. The proof is by contradiction. Suppose there exists a non-faulty node x , such that $K_{x,r} = K_{x,r-1}$, but, $K_{x,r+1} \neq K_{x,r}$. Now if $K_{x,r+1} \neq K_{x,r}$, then it must be the case that x reliably receives an array $val_{y,r}$ from some node $y \notin (I_{x,r} \cup F_{u,r+1})$, such that $K_{y,r} \supset K_{x,r}$. Suppose such a node y exists. Since $y \notin F_{u,r+1}$, by **Lemma 3.3**, y is non-faulty. By assumption, we have $K_{y,r} = K_{y,r-1}$, and $K_{x,r} = K_{x,r-1}$. Hence if $K_{y,r} \supset K_{x,r}$, then $K_{y,r-1} \supset K_{x,r-1}$. Since

$K_{x,r} = K_{x,r-1}$, this means that either x did not reliably receive $val_{y,r-1}$ in round r , or that node y was in the set $(I_{x,r-1} \cup F_{u,r})$. In either case, node y must be in the set $I_{x,r}$. This contradicts our assumption that node $y \notin (I_{x,r} \cup F_{u,r+1})$. Therefore, $K_{x,r+1} = K_{x,r}$, and since x was an arbitrary non-faulty node, we have $K_{u,r+1} = K_u$ for all non-faulty nodes u . We can now repeat this argument inductively for any round $r' \geq r$. Hence for any round $r' \geq r$, for all non-faulty nodes u , $K_{u,r'} = K_{u,r-1}$. \square

Lemma 3.7 (Containment Property). *At the end of phase 1, for every pair u, v of non-faulty nodes, K_u and K_v are comparable under the subset relation. That is, we have either $(K_u \subseteq K_v)$ or $(K_v \subseteq K_u)$.*

Proof. Let u, v be any non-faulty nodes. If no messages are tampered in phase 0, then the lemma is trivially true, since in that case all nodes reliably receive all the input values, and by the end of phase 0, we have $K_u = K_v = V$. Since no rule for updating the sets K_u and K_v make them smaller, we have, $K_u = K_v = V$, at the end of phase 1.

Hence, we will assume that the faulty node w tampers some message in phase 0. Let $|val_{u,r}|$ denote the number of non-null values in $val_{u,r}$, i.e. $|val_{u,r}| = |\{v \mid val_{u,r}[v] \neq \perp\}|$. We will say that a node u “learns” a new value in round r of phase 1 if $|val_{u,r}| > |val_{u,r-1}|$. In any given round of phase 1, either there is some node that learns a new value, or no node learns any new value. We will prove the following implication :

$$\begin{aligned} \exists r \forall u \in V - \{w\} \quad & val_{u,r} = val_{u,r-1} \\ \Rightarrow \forall r - 1 \leq r' \leq n(n-1) + 1 \forall u, v \in V - \{w\} \quad & K_{u,r'} \subseteq K_{v,r'} \vee K_{v,r'} \subseteq K_{u,r'} \end{aligned} \tag{3.4}$$

That is, if no non-faulty node learns a new value in a round r , then the containment property holds in all subsequent rounds.

So let us suppose there is a round $r \geq 1$ in which no node learns any new value. Consider any pair of non-faulty nodes u, v . Then either $v \in I_{u,r}$ or $v \notin I_{u,r}$. We consider both cases.

- $v \in I_{u,r}$: In this case, by **Lemma 3.5**, $u \notin (I_{v,r-1} \cup F_{v,r})$, and v knows a uv path disjoint with $F_{v,r}$. Hence v reliably receives $val_{u,r-1}$ and merges it with $val_{v,r-1}$. Since $K_{v,r} = K_{v,r-1}$, it must be the case that $K_{u,r-1} \subseteq K_{v,r-1}$.
- $v \notin I_{u,r}$: Then it must be the case that $v \notin (F_{u,r} \cup I_{u,r-1})$, and u reliably receives $val_{v,r-1}$, because otherwise, by the update rules of phase 1.2, v would be in the set $I_{u,r}$. Since $K_{u,r} = K_{u,r-1}$, it must be the case that $K_{v,r-1} \subseteq K_{u,r-1}$.

Let x be any arbitrary non-faulty node. By **Lemma 3.6**, we have $K_{x,r'} = K_{x,r-1}$ for any round $r' \geq r$. Hence, the containment property also holds in every round $r' \geq r - 1$.

Since any node can learn at most $(n - 1)$ additional values, either all nodes learn all the values, in which case $K_u = K_v = V$, for any non-faulty nodes u, v , or there is a round r in which no node learns a new value. In either case, the containment property holds after $n(n - 1)$ rounds of phase 1.2. \square

Lemma 3.8. *Let u, v be two non-faulty nodes. In the decision phase, if $K_u \subset K_v$, then the following conditions hold.*

- $K[v][u] = K_u$
- $u \notin F_v$
- $K[u][v] = V$.

Proof. By **Lemma 3.7**, we know that at the end of phase 1 the containment property holds. If $K_u \subset K_v$, then it must be the case that the faulty node w tampers a message in phase 0 and that there is a round r of phase 1, in which no node learns a new value. Because otherwise, $K_u = K_v = V$ for all non-faulty nodes u, v . By **Lemma 3.6**, for any non-faulty node x , $K_{x,n(n-1)+1} = K_{x,r'-1}$ for any round $r' \geq r$. In particular, $K_{x,n(n-1)+1} = K_{x,n(n-1)}$. Note that the state of the variable K_u in the decision phase equals $K_{u,n(n-1)+1}$. Therefore, in the decision phase, if $K_u \subset K_v$, then $K_{u,n(n-1)} \subset K_{v,n(n-1)}$. This must mean that in round $n(n - 1) + 1$, node u either did not reliably receive $val_{v,n(n-1)}$ or that node v was in the set $(I_{u,n(n-1)} \cup F_{u,n(n-1)+1})$. In either case, by the update rules in phase 1.2, node v is in the set $I_{u,n(n-1)+1}$. Hence, by **Lemma 3.5**, $u \notin I_{v,n(n-1)+1}$ and in round $n(n - 1) + 1$ of phase 1, v knows a fault-free uv path Q . Hence, v reliably receives $K_{u,n(n-1)}$ and sets $K[v][u]_{n(n-1)+1} = K_{u,n(n-1)} = K_{u,n(n-1)+1}$. Also, since $v \in I_{u,n(n-1)+1}$, u sets $K[u][v]_{n(n-1)+1} = V$. \square

Corollary 3.9. *In the decision phase, for any non-faulty nodes u, v , either $K[u][v] = V$ or $K[u][v] = K_v$.*

Proof. By **Lemma 3.8**, this is true if in the decision phase $K_u \subset K_v$ or $K_v \subset K_u$. Hence, we only need to prove the argument when $K_u = K_v$. Note that since there are $n(n - 1) + 1$ rounds of phase 1, no node learns a new value in round $n(n - 1) + 1$ of phase 1. This is because each node can learn at most $(n - 1)$ values. So either all nodes learn all values by the end of round $n(n - 1)$ or there is a round $1 \leq r \leq n(n - 1)$, in which no node learns a

new value. In either case, by **Lemma 3.6**, no node learns a new value in round $n(n-1)+1$ of phase 1. Now either $v \in I_{u,n(n-1)+1}$, or $v \notin (F_{u,n(n-1)+1} \cup I_{u,n(n-1)})$ and u reliably receives $val_{v,n(n-1)} = val_{v,(n-1)+1}$ in round $n(n-1)+1$. In the first case, then u sets $K[u][v] = V$, and in the second case, u sets $K[u][v] = K_{v,n(n-1)} = K_{v,n(n-1)+1}$. \square

Next, we will prove that updates to the val array are correct. That is, if u, x are non-faulty nodes, and $val_u[x] = b$, then the input value of node x is b . Otherwise, if x is faulty, then for any non-faulty nodes u, v , $val_u[x] = val_v[x]$. So that all non-faulty nodes decide a value from the same multi-set of values.

Lemma 3.10. *Let u, v be two non-faulty nodes. For any $x \in V$, if $val_u[x] \neq \perp$ and $val_v[x] \neq \perp$, then $val_u[x] = val_v[x]$*

Proof. Suppose x is a non-faulty node and suppose $val_u[x] = b$ (for some $b \in \{0, 1\}$). Suppose this value is set in phase 0. Then it must be the case that either x is a neighbor of u or u receives this value from two internally disjoint paths P and Q . In the first case, since x sends this value directly to u it must be that b is the input value of x . In the second case, since the faulty node can be either in P or in Q but not both, again it must be the case that b is the input value of x . If this value is set in round 1 of phase 1, then it must be the case that u reliably receives val_v from some non-faulty node $v \notin (F_{u,0} \cup I_{u,0})$ such that $val_v[x] \neq \perp$. Since $v \notin F_{u,0}$, by **Lemma 3.3**, v must be non-faulty. This must mean that v sets $val_v[x] = b$ in phase 0. Hence, b must be the value of node x , as argued previously. If the $val_u[x]$ was instead set in some round $r > 1$ of phase 1, then we can complete the argument by using induction on the round of phase 1. So if u, v, x are any non-faulty nodes and $val_u[x] \neq \perp$ and $val_v[x] \neq \perp$ then it must be the case that $val_u[x] = val_v[x] = b$, where b is the input value of x .

Suppose that x is a faulty node. Then since there is only one faulty node, if x sends a message, there is no other faulty node to tamper this message hence every node u receives this message through 2 internally disjoint paths. Therefore, for any non-faulty nodes u, v we have $val_u[x] = val_v[x]$ at the end of phase 0. \square

Finally, we are in a position to prove that our algorithm correctly achieves consensus.

Theorem 3.11. *The algorithm described in **Section 3.1** satisfies validity and agreement.*

Proof. By **Lemma 3.7**, we know that the containment property holds at the start of the decision phase. Let W be the set containing faulty nodes, i.e. if there is a faulty node w , then $W = \{w\}$, and otherwise, $W = \emptyset$. Let u be any non-faulty node, and let K_x be the

smallest set in $\{K_v \mid v \in V - W\}$. Then there are two cases, either $K_u = K_x$ or $K_u \supset K_x$. If the former is true, then we have

$$D_u = \cap_{v \notin F_u} K[u][v] \quad (3.5)$$

$$= K[u][u] \cap_{v \notin (F_u \cup \{u\})} K[u][v] \quad (u \notin F_u) \quad (3.6)$$

$$= K_u \cap_{v \notin (F_u \cup \{u\})} K[u][v] \quad (K[u][u] = K_u) \quad (3.7)$$

$$= K_u \quad (K_u \subseteq K_v \subseteq K[u][v] \text{ by } \textbf{Corollary 3.9}) \quad (3.8)$$

$$= K_x \quad (K_u = K_x) \quad (3.9)$$

Otherwise, if $K_u \supset K_x$, then

$$D_u = \cap_{v \notin F_u} K[u][v] \quad (3.10)$$

$$= K[u][x] \cap_{v \notin (F_u \cup \{x\})} K[u][v] \quad (\text{By } \textbf{Lemma 3.8}) \quad (3.11)$$

$$= K_x \cap_{v \notin (F_u \cup \{x\})} K[u][v] \quad (\text{By } \textbf{Lemma 3.8}) \quad (3.12)$$

$$= K_x \quad (K_u \subseteq K_v \subseteq K[u][v] \text{ by } \textbf{Corollary 3.9}) \quad (3.13)$$

So all nodes compute $D = K_x$ as the decision set. By **Lemma 3.10**, every non-faulty node u computes the same the multi-set $S = \{val_u[v] \mid v \in D\}$ in the decision phase. Finally, since G is 2-connected and has at least 3 nodes, every node must have at least two neighbors. Therefore, every node knows at least three values after phase 0, its own and the values of its neighbors, i.e. $|K_x| \geq 3$, and therefore, $|S| \geq 3$. Since $f = 1$, the decision computed as the majority in S (with tie broken in favor of 0) necessarily equals the input of a non-faulty node. \square

We showed an algorithm for achieving consensus in the case of a single fault, on any arbitrary graph that is 2-connected, and has at least three nodes. As mentioned earlier, in the case of point-to-point links, consensus is impossible on any graph that is at most $2f$ -connected. The key ideas that our proof uses is that whenever a faulty node tampers a message, the neighbor of the faulty node can detect this tampering.

Chapter 4: A sufficiency result for arbitrary f

In this chapter we will discuss how to extend the sufficiency results for the case where there is more than one fault. We will show that if there are at most f faults, then consensus is achievable on any arbitrary graph that is $2f$ connected. Not all results discussed in **Chapter 3** easily extend to arbitrary values of f . For example, it turns out that the kind of arguments used in **Lemma 3.4** for the case of a single fault, do not apply if there is more than one fault. As such, we need a more complicated algorithm for the general case. Some of our modifications are just generalizations from the single fault case, but others use different ideas. A key observation that helps us here is that if faulty nodes tamper messages, then in addition to the neighbor of the faulty nodes, the originator of the message can also find out the identity of the faulty nodes. We add a new phase in the algorithm that allows this fault detection. In the modified algorithm presented in **Section 4.1**, each node u keeps two sets F_u and F_u^* , for nodes that it thinks may be faulty and for nodes that it *knows* are faulty, respectively. We will also need to redefine some of the terminology from **Chapter 3**.

Definition 4.1. *A node u reliably receives a message m from node v , if any of the following conditions are true.*

- u and v are neighbors and v sends m .
- $|F_u^*| = k$ and u receives message m from at least $f - k + 1$ internally disjoint vu paths that do not contain any node in F_u^* .
- u receives a message from v through a path disjoint to F_u .

Definition 4.2. *We will say that node u receives a trusted message m , if u receives it from $f + 1$ node disjoint paths (including the source node), or u itself sends the message m .*

4.1 ALGORITHM

Each node u keeps the following state

- An array val_u to store the input values of nodes in the graph. Initially $val_u[u]$ is set to the input value of node u and all other elements of val_u are set to the null value, \perp
- A set F_u for the nodes that u does not know to be fault-free (i.e. nodes that are potentially faulty from u 's viewpoint). F_u is initialized to $V - \{u\}$

- A set F_u^* for the nodes that u knows are faulty. This is initialized to be the empty set, \emptyset .
- An array $K[u]$, where $K[u][u]$ is a subset of the nodes whose values are known to u , and for $v \neq u$, $K[u][v]$ is the set of nodes in $K[v][v]$ from u 's perspective. We will use the shorthand K_u for $K[u][u]$.

Messages

Every message sent by a non-faulty node is an ordered triple, consisting of a tag, a payload, and a path. Each message that a node u originates is of one of the following forms.

- To send its own message, u sends $(input_u, val_u[u], [])$.
- To indicate that node u received a message (tag, m, P) , u sends $(recvd, (tag, m, P), [])$.
- To indicate that nodes in set F_u^* are faulty, u sends $(faulty, F_u^*, [])$.
- To send K_u , node u sends $(Kset_u, K_u, [])$.

The only difference between a message forwarded by u and a message originated by u is in the path part of the message.

We will assume that any received messages that are not of the above forms are discarded.

The algorithm runs in phases. Details of the different phases of the algorithm for node u are described below.

Phase 0

• Phase 0.1

- Node u floods the message $(input_u, val_u[u], [])$.
- If node u reliably receives a message $m = (input_v, b)$, then it sets $val_u[v] = b$.

We will assume that all nodes send their input values in phase 0. Since there could be faulty nodes in the graph, this property is not guaranteed. However, it can be achieved by having the non-faulty nodes assume a default value in case they do not receive a message from a node.

For a non-faulty node u , we will say that a “node u knows what node v sent in phase 0.1”, if for every message $(tag, payload, P)$ sent by v , u receives a trusted message

$(recv_d, (tag, payload, P + [v]))$. Note that if a neighbor v of u sends a message $m = (tag, payload, P)$, then in phase 0.2, u sends the message $m' = (tag, payload, P + [v])$, and hence u trivially receives the trusted message m' . Therefore, for any neighbor v of u , node u knows what v sent in phase 0.1.

• **Phase 0.2**

- If node u received message $(input_v, b, P)$ in phase 0.1, then it floods the message $(recv_d, (input_v, b, P), [])$.
- If node u did not reliably receive node v 's input value in phase 0.1, then consider a set of $2f$ internally disjoint paths $P_1 \dots P_{2f}$ from v to u . Let Q_i be the subpath $(v, u)_{P_i}$ and let $q_{i,1} \dots q_{i,l_i}$ be the subsequence of nodes in Q_i such that u knows what each $q_{i,j}$ sent in phase 0.1. Note that on each path Q_i , at least one such node exists, because Q_i terminates at a neighbor of u . Now define $w_i = q_{i,1}$, if for each $q_{i,j}$ u receives a trusted message $(recv_d, (input_v, b, [v, q_{i,j}]_{P_i}))$, and otherwise let $w_i = q_{i,j}$ such that j is the smallest integer for which u receives the trusted messages $(recv_d, (input_v, b, [v, q_{i,j-1}]_{P_i}))$ and $(recv_d, (input_v, \bar{b}, [v, q_{i,j}]_{P_i}))$. We define w_i to be $q_{i,j}$ in this case. Node u sets $F_u = F_u \cap \{w_i \mid 1 \leq i \leq 2f\}$.
- If node u reliably receives v 's input value b (u reliably receives its own value) in phase 0.1, then consider a set of $2f$ internally disjoint paths $P_1 \dots P_{2f}$ from v to any node x . Let P_i be a path on which a faulty node tampers v 's message and instead forwards $\overline{val_v[v]}$. Let $W_{v,x}$ be the set containing the first node w_i on each path P_i , such that u receives the trusted message $(recv_d, (input_v, \overline{val_u[v]}, [v, w_i]_{P_i}))$. Node u sets $F_u^* = F_u^* \cup W_{v,x}$. Furthermore, if $|F_u^*| = f$, then u sets $F_u = F_u^*$.
- At the end of the round, node u goes through all the messages it received in phase 0.1 and for all messages, $(input_v, b)$, reliably received by u , with respect to the updated sets F_u and F_u^* , u sets $val_u[v] = b$.

Phase 1

- If $|F_u^*| = f$, then u floods $(faulty, F_u^*, [])$
- If $|F_u^*| < f$, and node u reliably receives message $(faulty, F_v^*)$ from a node $v \notin F_u$, then u sets $F_u = F_u^* = F_v^*$.

- At the end of the round, node u goes through all the messages it received in phase 0.1, and for all messages, $(input_v, b)$, reliably received by u , with respect to the updated sets F_u and F_u^* , u sets $val_u[v] = b$.

After phase 1, node u sets $K_u = \{v \mid val_u[v] \neq \perp\}$.

Phase 2 (Repeated n times)

- **Phase 2.1**

- If $|F_u^*| < f$, then u floods $(Kset_u, K_u, \square)$.
- If $|F_u^*| < f$ and u reliably receives $(Kset_v, K_v)$ from node v and $K_v \supseteq F_u \cup \{u\}$ then, u sets $K[u][v] = K_v$. If u reliably receives $(Kset_v, K_v)$ from v and $K_v \not\supseteq F_u \cup \{u\}$ then u sets $K[u][v] = V$ and sets $F_u^* = F_u^* \cup \{v\}$. If u does not reliably receive any message from node v with the tag \mathbf{Kset}_v , then u sets $K[u][v] = V$.

- **Phase 2.2**

- If node u receives a message m in Phase 2.1, then u floods $(recvd, m, \square)$ in phase 2.2.
- If $|F_u^*| < f$, and node u reliably receives $(Kset_v, K_v)$ (again, recall u reliably receives its own messages) in phase 2.1, then consider a set of $2f$ internally disjoint paths $P_1 \dots P_{2f}$ from v to any node x . Let P_i be a path on which a faulty node tampers v 's message and instead forwards some set $\overline{K_v} \neq K_v$. Let $W_{v,x}$ be the set containing the first node w_i on each path P_i , such that u receives the trusted message $(recvd, (Kset_v, \overline{K_v}, [v, w_i]_{P_i}))$. Node u sets $F_u^* = F_u^* \cup W_{v,x}$.

- **Phase 2.3**

- If $|F_u^*| < f$, then u goes through all messages received in Phase 1, and if u reliably received $(faulty, F_v^*)$ (with respect to the updated sets F_u and F_u^*) from some node $v \notin F_u$, then u sets $F_u = F_u^* = F_v^*$.
- Similarly, node u goes through all messages in Phase 0.1, and for all messages $(input_v, b)$ reliably received by u with respect to the updated sets F_u and F_u^* , u sets $val_u[v] = b$.
- Finally, if $|F_u^*| < f$, then u sets $K_u = \cap_{v \in V} K[u][v]$, otherwise u sets $K_u = V$.

Decision Phase

- Node u floods $(Kset_u, K_u, [])$.
- If node u reliably receives $(Kset_v, K_v)$ where $v \in V - F_u$ then u sets $K[u][v] = K_v$. If instead u does not reliably receive any message with the tag \mathbf{Kset}_v , from a node $v \in V - F_u$, then u sets $K[u][v] = V$.
- Node u sets $D_u = \cap_{v \in V - F_u} K[u][v]$ and computes the decision as described in 3.1.

4.2 CORRECTNESS

Let $G = (V, E)$ be an arbitrary $2f$ -connected graph, where $f \geq 1$. We will prove that consensus is achievable on graph G .

We will use W to refer to set of faulty nodes in the graph G . Thus $0 \leq |W| \leq f$.

Definition 4.3. For a node u , let $N(u)$ denote the non-faulty nodes in the neighborhood of u , i.e. $N(u) = \{v \mid (u, v) \in E \wedge v \in V - W\}$

Our first two lemmas show how our modified algorithm helps nodes in detecting faulty behavior.

Lemma 4.4. If a faulty node $w \in W$ sent a message $m = (tag, payload, W)$ in phase 0.1, then in phase 0.2 all non-faulty nodes know that w sent m .

Proof. Let u be an arbitrary non-faulty node. If node w is a neighbor of u , then u receives m and sends $m' = (recvd, (tag, payload, P + [w]))$, hence, u trivially knows what w sent in phase 0.1. If node w is not a neighbor of u , then since we are assuming the graph, G , is $2f$ -connected, w has at least $2f$ internally node disjoint paths to u . Take any set of $2f$ internally disjoint paths, $P_1 \dots P_{2f}$, from w to u . Let Q_i denote the subpath $(w, u]_{P_i}$ of P_i . Then the paths $Q_i \dots Q_{2f}$ are pairwise disjoint and none of them contain w . Now since there are at most f faulty nodes, and w is one of them, it must be the case the $f + 1$ of the paths in $Q_i \dots Q_{2f}$ start at a node in $N(w)$. All nodes in $N(w)$ receive m in phase 0.1, and hence send the message $m' = (recvd, (tag, payload, P + [w]))$ in phase 0.2. Therefore, u receives m' from at least $f + 1$ pairwise disjoint paths in phase 0.2. Hence, by definition, u knows that w sent message m . \square

Next, we prove that any updates made to the sets F_u and F_u^* before phase 2 are correct. That is, if v is a non-faulty node, then v is not in the set F_u , and if a node w is in the set F_u^* , then w is a faulty node. Proving the correctness of updates made in phase 2, specifically in phase 2.2 requires more work, and is established in **Lemma 4.12**.

Lemma 4.5. *Let $w \in W$ be a faulty node. Then for any non-faulty node u , at any point in the algorithm before the start of phase 2, $W \subseteq F_u$ and $F_u^* \subseteq W$.*

Proof. Before phase 2, the sets F_u and F_u^* are updated in two phases, phase 0.2, phase 1. We will prove the correctness of updates in both phases.

- *Phase 0.2 :* Suppose there exists a non-faulty node x such that at the end of phase 0.1, $val_u[x] = \perp$. Then in phase 0.2, consider any set of $2f$ paths $P_1 \dots P_{2f}$ from x to u , and let Q_i be the subpath $(x, u)_{P_i}$. Since u and x are non-faulty, and $val_u[x] = \perp$, it must be the case that there is exactly one faulty node on f paths in $Q_1 \dots Q_{2f}$. Since the labeling is arbitrary, we will assume that the faulty nodes are on paths $Q_1 \dots Q_f$. Let q_i be the first node on the path Q_i for which u knows what it sent in phase 0.1. By **Lemma 4.4**, all nodes y for which x does not know what y sent in phase 0.1, y must be non-faulty. In particular, for all $1 \leq i \leq f$, all nodes in $[x, q_i]_{P_i}$ are non-faulty. Suppose u receives a trusted message, $(recvd, (input_x, b, [x, q_i]_{P_i}))$ in phase 0.2. Note that there is at most one faulty node on each path P_i . Therefore, if q_i is faulty, then for any other node q'_i on path Q_i for which node u knows what it sent in phase 0.2, u , must reliably receive $(recvd, (input_x, b, [x, q_i]_{P_i}))$. Hence u correctly sets $q_i = w_i$. Otherwise, if q_i is non-faulty, and q'_i is the first node on path Q_i from which u reliably receives $(recvd, (input_x, \bar{b}, [x, q'_i]_{P_i}))$, then q'_i must be a faulty node, and x correctly sets $w_i = q'_i$. So if $w \in W$ is any faulty node, then $w \in F_u \cap \{w_i \mid 1 \leq i \leq 2f\}$.
So let us suppose that x is a node such that at the end of phase 0.1, $val_u[x] \neq \perp$. Then consider any set of $2f$ paths $P_1 \dots P_{2f}$ between x and some node y . Since $val_u[x] \neq \perp$, it must be the case that u reliably receives the input value of x , and hence $val_u[x] = val_x[x]$. Let p_i be the first node on path P_i , such that u reliably receives $(recvd, (input_x, \overline{val_x[x]}, [x, p_i]_{P_i}))$ in phase 0.2. Since any node p'_i for which u does not know what p'_i sent in phase 0.1 must be non-faulty, p_i must be a faulty node. Hence u correctly sets $w_i = p_i$, and the set $F_u^* = F_u^* \cup W$ only contains faulty nodes.
- *Phase 1 :* The sets F_u and F_u^* are only updated in phase 1, if u reliably receives a message $(faulty, F_v^*)$ from a node $v \notin F_u$. If $v \notin F_u$ in phase 1, then v must be a non-faulty node. Since the last update to F_v^* was made in phase 0.2, as argued previously, F_v^* must contain only faulty nodes. Furthermore, since v is non-faulty, v must have sent this message only if $|F_v^*| = f$. Therefore, the update $F_u = F_u^* = F_v^*$ is correct.

□

Lemma 4.6. *Let u be an arbitrary non-faulty node, such that $W \subseteq F_u$ and $F_u^* \subseteq W$. Then u reliably receives any message sent by a faulty node w . Furthermore, if $|F_u^*| \geq 1$ or $|F_u| < 2f$, then u reliably receives messages from every node $x \in V$.*

Proof. We prove both claims below.

- *A faulty node w sends a message m :* Since G is $2f$ -connected, there are $2f$ internally disjoint paths between w and u . Since there are at most f faulty nodes, and w is one of them, $f + 1$ of the paths (say, $P_1 \dots P_{f+1}$) must be disjoint to W . By assumption, $F_u^* \subseteq W$. Hence, $P_1 \dots P_{f+1}$ are disjoint to F_u^* . Therefore, u receives m through $f + 1 \geq f - |F_u^*| + 1$ paths. By definition, u reliably receives m .
- *A non-faulty node x sends a message and $|F_u^*| \geq 1$ or $|F_u| < 2f$:* Let us suppose that $|F_u^*| \geq 1$ and consider any arbitrary node x . Since G is $2f$ -connected, there are $2f$ internally disjoint ux paths $P_1 \dots P_{2f}$. Since there are only f faulty nodes, f of the paths in $P_1 \dots P_{2f}$ must not contain any faulty nodes. Since labeling is arbitrary, assume the faulty nodes are on paths $P_{f+1} \dots P_{2f}$. Since, by assumption $F_u^* \subseteq W$, the paths $P_1 \dots P_f$ are disjoint to F_u^* . Hence if node x sends a message m , u receives it identically on paths $P_1 \dots P_f$. Since, $|F_u^*| \geq 1$, we have $f \geq f - |F_u^*| + 1$, and hence, u reliably receives m .

Similarly if $|F_u| < 2f$, then since the graph is $2f$ -connected, for any node $x \in V$, there must be a ux path that is disjoint to F_u . Hence in this case too, u reliably receives messages from every node.

□

Corollary 4.7. *Let u be an arbitrary non-faulty node. Then u reliably receives any message sent by a faulty node w in phases 0 and 1. Furthermore, if $|F_u^*| \geq 1$ or $|F_u| < 2f$, then u reliably receives messages sent by every node $x \in V$ in phases 0 and 1.*

Proof. This follows from **Lemma 4.6** and **Lemma 4.5**.

□

Lemma 4.8 (Fault Detection). *Let $u \in V - W$ be a non-faulty node, and let S be the set of non-faulty nodes that reliably receive $val_u[u]$ in phase 0.1. If $S \neq V - W$, then there are necessarily f nodes in W , and every node $x \in S$ learns the identity of all the faulty nodes (i.e. $|F_x^*| = f$ and $F_x = F_x^* = W$), and knows the input values of all nodes after phase 0.2 ends.*

Proof. Suppose there exists a non-faulty node $v \in V - W$, such that $v \notin S$. Since v does not reliably receive $val_u[u]$ in phase 0.1, this means that there must be exactly $2f$ internally disjoint paths $P_1 \dots P_{2f}$ from u to v , such that there is a faulty node each on exactly f of the paths in $P_1 \dots P_f$ that forwarded $\overline{val_u[u]}$ in phase 0.1. Since the labeling is arbitrary, we can assume that the faulty nodes are on paths $P_1 \dots P_f$. Let w_i denote the faulty node on path P_i . Since there is exactly one faulty node on each path $P_1 \dots P_f$, each w_i on path P_i must be the first node that forwards $\overline{val_u[u]}$. Now for any node $x \in S$, x reliably receives u 's input value in phase 0.1, and by **Lemma 4.4**, x receives the trusted messages $(recvd, (input_u, \overline{val_u[u]}, [u, w_i]_{P_i}))$ in phase 0.2. Hence x sets $F_x^* = \{w_i \mid 1 \leq i \leq f\}$. Now since $|F_x^*| = f$, x sets $F_x = F_x^*$, and by **Corollary 4.7**, x can reliably receive messages from every node, and therefore, x learns the input values of all the nodes by the end of phase 0.2. \square

Lemma 4.9. *Let u, v be two non-faulty nodes, such that v does not reliably receive $val_u[u]$ in phase 0.1, then after phase 0.2, for any non-faulty node x , we have $u \notin F_x$ and $|F_x| \leq 2f$.*

Proof. There are two cases to consider.

1. **x reliably receives $val_u[u]$ in phase 0.1 :** If x reliably receives $val_u[u]$, then by **Lemma 4.8**, x knows the identity of all the faulty nodes. That is, $|F_x^*| = f$ and $F_x = F_x^* = W$. Since u is not a faulty node, by **Lemma 4.5**, $u \notin F_x$.
2. **x does not reliably receive $val_u[u]$ in phase 0.1 :** If x does not reliably receive $val_u[u]$ in phase 0.1, then consider a set of $2f$ paths $P_1 \dots P_{2f}$, and let Q_i be the subpath $(u, x)_{P_i}$. By the update rules in phase 0.2, x picks a node w_i on each path Q_i and sets $F_x = F_x \cap \{w_i \mid 1 \leq i \leq 2f\}$. Therefore, $|F_x| \leq 2f$ and since, by construction, $u \notin Q_i$, therefore, $u \notin F_x$.

\square

Lemma 4.10. *Let u, v be non-faulty nodes, such that $val_u[v] = \perp$ at the end of phase 0.1. Then, for any non-faulty node $x \in V - W$, at the end of phase 1, either $|F_x^*| = f$, or $|F_x| = 2f$.*

Proof. The proof is by contradiction. Let x be a non-faulty node such that at the end of phase 1, $|F_x^*| \neq f$, and $F_x \neq 2f$. Note that since there are no rules for updating F_x that make the set larger, by **Lemma 4.9**, we have that by the end of phase 1, $|F_x| \leq 2f$. By **Lemma 4.5**, it is also not possible that $|F_x^*| > f$, since there are at most f faulty nodes. Thus, we can conclude that at the end of phase 1, we have $|F_x^*| < f$ and $|F_x| < 2f$. Now,

since we have $|F_x^*| < f$, it must be the case that F_x^* is not updated in phase 1, which means that F_x is also not updated in phase 1. So if $|F_x| < 2f$ at the end of phase 1, then it must be the case that $|F_x| < 2f$ at the end of phase 0.2. But by **Lemma 4.9**, at the end of phase 0.2, x knows that v is a non-faulty node, and since $|F_x| < 2f$ in phase 1, then by **Corollary 4.7**, x reliably receives $(faulty, F_v^*)$ in phase 1, and sets $F_x = F_x^* = F_v^*$. By **Lemma 4.8**, $|F_v^*| = f$, hence $|F_x^*| = f$. This contradicts our assumption. \square

Let $F_{u,r}$ denote the state of F_u at the end of round r of phase 2, and let $F_{u,0}$ be the state of F_u at the end of phase 1. Analogously define $F_{u,r}^*$ and $K_{u,r}$.

Definition 4.11. We will call a non-faulty node u , an “participating node” in round r of phase 2, if $|F_{u,r-1}^*| < f$.

Note that only participating nodes in round r of phase 2 originate a message in round r of phase 2.1.

Lemma 4.12. Let u be an arbitrary non-faulty node. Then, for any integer $0 \leq r \leq n$, the following three conditions hold.

- $W \subseteq F_{u,r}$
- $F_{u,r}^* \subseteq W$
- If a non-faulty node v is a participating node in round $r+1$ of phase 2, then $K_{u,r} \supseteq (F_{v,0} \cup \{v\})$.

Proof. The proof is by induction on r . For the base case, by **Lemma 4.5**, $W \subseteq F_{u,0}$ and $F_{u,0}^* \subseteq W$. For the third condition, we consider the following cases.

- In phase 0.1, all non-faulty nodes reliably received all input values : In this case, $K_{u,0} = V$. Hence, $K_{u,0} \supseteq (F_{v,0} \cup \{v\})$.
- After phase 0.1 there exists nodes a non-faulty node x and faulty node y such that $val_x[y] = \perp$: This case is not possible by **Corollary 4.7**
- After phase 0.1, there exist non-faulty nodes x and y , such that $val_x[y] = \perp$: Suppose $v \notin K_{u,0}$ (i.e. u does not reliably receive $val_v[v]$ in phase 0.1), then by **Lemma 4.8**, $|F_{v,0}^*| = f$. Hence, by definition, v is not a participating node in round 1 of phase 2. Since we have assumed that v is a participating node in round 1 of phase 2, $v \in K_{u,0}$. Now consider any node $z \in F_{v,0}$. Since no rule for updating the set F_v makes it larger, it must be the case that z is in the set F_v after phase 0.2. If z is a non-faulty node,

then by **Lemma 4.9**, every non-faulty node reliably receives $val_z[z]$ in phase 0.1. If z is faulty node, then by **Corollary 4.7** u reliably receives $val_z[z]$ in phase 0.1. Therefore, $z \in K_{u,0}$. Since z was an arbitrary node in $F_{v,0}$, $K_{u,0} \supseteq F_{v,0}$. Since we showed in the previous paragraph that $v \in K_{u,0}$, therefore, $K_{u,0} \supseteq (F_{v,0} \cup \{u\})$.

For the induction hypothesis, assume that the three conditions of the lemma hold for all rounds $0 \leq r \leq k$ (for some $k < n$). We consider updates made in round $k + 1$ of phases 2.1, 2.2 and 2.3 and show that all three conditions hold after every phase.

- *Phase 2.1* : If node u is not a participating node in round $k + 1$ of phase 2.1, then u does not update the sets F_u , F_u^* and K_u . Therefore, in that case, the claim trivially holds after round $k + 1$ of phase 2.1

So let us suppose that node u is a participating node in round $k + 1$ of phase 2. By the induction hypothesis, we have $K_{z,k} \supseteq F_{u,0} \cup \{u\}$ for all non-faulty nodes z . Therefore, if node u reliably receives a set $K_{w,k} \not\supseteq F_{u,0} \cup \{u\}$, then node w must be faulty node. Hence $(F_{u,k} \cup \{w\}) \subseteq W$.

- *Phase 2.2* : In phase 2.2, node u only potentially updates the set F_u^* . Correctness of the update follows from an argument similar to the correctness of the update in phase 0.2 in the proof of **Lemma 4.5**.
- *Phase 2.3* : In phase 2.3, u only updates F_u and F_u^* if a node $v \notin F_{u,k}$ sent a message (*faulty*, F_v^*) (with respect to the updated sets F_u and F_u^*) in phase 1. By the induction hypothesis, $W \subseteq F_{u,k}$. Hence if $v \notin F_{u,k}$, then v must be a non-faulty node. Node v only sends such a message if $|F_v^*| = f$. By **Lemma 4.5**, $F_v^* \subseteq W$. Since $|F_v^*| = f$, $F_v^* = W$. Hence, the update $F_u = F_u^* = F_v^*$ is correct, and the first two conditions of the lemma hold after round $k + 1$ of phase 2.3.

Note that no rule for updating F_v^* makes the set smaller. Therefore, if a node v is a participating node in round $k + 2$ of phase 2, then v must be a participating node in round $k + 1$ of phase 2. We will show, by contradiction, that $K_{u,k+1} \supseteq (F_{v,0} \cup \{v\})$. Suppose $K_{u,k+1} \not\supseteq (F_{v,0} \cup \{v\})$. By the induction hypothesis, $K_{x,k} \supseteq (F_{v,0} \cup \{v\})$, for all non-faulty nodes x . Hence, if $K_{u,k+1} \not\supseteq (F_{v,0} \cup \{v\})$, then $|F_{u,k+1}^*| < f$, u sets $K_{u,k+1} = \cap_{x \in V} K[u][x]_k$, and there exists a faulty node w that sent a set $K_{w,k} \not\supseteq (F_{v,0} \cup \{v\})$. We consider the following cases

- *In phase 0.1, all non-faulty nodes reliably received all input values* : In this case, $F_{x,0} = V - \{x\}$ for all non-faulty nodes x , as argued in the base case. Since

$F_{v,0} \cup \{v\} = V$, therefore, by the induction hypothesis, $K_{x,k} \supseteq V$ for all non-faulty nodes x . Furthermore, since $|F_{u,k+1}^*| < f$, the set F_u is not updated in the first $k+1$ rounds of phase 2. Hence $F_{u,k} = F_{u,0} = V - \{u\}$. Hence, if node u reliably receives a set $K_{w,k} \not\supseteq V$, u sets $K[u][w] = V$. Therefore for all nodes x , $K[u][x]_k = V$. Therefore,

$$K_{u,k+1} = \cap_x K[u][x] \quad (4.1)$$

$$= \cap_x V \quad (4.2)$$

$$= V \quad (4.3)$$

$$= F_{v,0} \cup \{v\} \quad (4.4)$$

- After phase 0.1 there exists nodes a non-faulty node x and faulty node y such that $val_x[y] = \perp$: This is not possible as discussed in the base case.
- After phase 0.1, there exist nodes x and y , such that $val_x[y] = \perp$: By the induction hypothesis, $W \subseteq F_{v,k}$ and $F_{v,k}^* \subseteq W$. Hence, by **Corollary 4.7**, node v reliably receives $K_{w,k}$ in round $k+1$. But, since $K_{w,k} \not\supseteq (F_{v,k} \cup \{v\})$, in round $k+1$ of phase 2.1, v adds the faulty node w to the set F_v^* . By **Lemma 4.9**, at the end of phase 0.2, y knows the actual set of faulty nodes, i.e. $|F_y^*| = f$. Therefore in phase 1, y must have sent the message (*faulty*, F_y^* , []). Since v learned that w is a faulty node, $|F_{v,k+1}^*| \geq 1$. All updates made so far to the sets F_v and F_v^* are correct. Hence, by **Lemma 4.6**, v can reliably receive messages from node y . So in round $k+1$ of phase 2.3, v sets $F_{v,k+1} = F_{v,k+1}^* = F_y^*$. Since, $|F_y^*| = f$, $|F_{v,k+1}^*| = f$. Therefore v is not a participating node in round $k+2$ of phase 2, which is a contradiction.

□

Corollary 4.13. *Let u be an arbitrary non-faulty node. Then u reliably receives any message sent by a faulty node, w . Furthermore, if $|F_u^*| \geq 1$ or $|F_u| < 2f$, then u reliably receives messages from every node $x \in V$.*

Proof. This follows from **Lemma 4.12** and **Lemma 4.6**. □

Lemma 4.14. *Let u, v be non-faulty nodes, such that $val_u[v] = \perp$ at the end of phase 0.1. If there is a round $1 \leq r \leq n$ of phase 2 in which the number of participating nodes does not decrease, then the following conditions hold*

- *If x is a participating node in round r , then x does not receive a message $(Kset_y, K_{y,r-1})$, such that $K_{y,r-1} \not\supseteq (F_{x,r} \cup \{x\})$*

- If x is a participating node in round r , and y is any node in V , then y reliably receives $(Kset_x, K_{x,r-1})$ in round r

Proof. We prove both claims below.

- Suppose that in round r of phase 2.1, a non-faulty node x receives a set $K_{w,r-1} \not\supseteq (F_{x,r} \cup \{x\})$. Note that by **Lemma 4.8**, node v knows the identity of all the faulty nodes by the end of phase 0. Hence, v must have sent $(faulty, F_v^*)$ in phase 1. Since, node x receives a set $K_w \not\supseteq (F_{x,r} \cup \{x\})$ in round r of phase 2, x sets $F_x^* = F_{x,r}^* \cup \{w\}$. Since $|F_x^*| \geq 1$, by **Corollary 4.13**, x can reliably receive messages from every node. Hence, in phase 2.3, x sets $F_{x,r} = F_{x,r}^* = F_v^*$. Since $|F_v^*| = f$, therefore, $|F_{x,r}^*| = f$. Therefore, by definition, node x is not a participating node in round $r + 1$. However, by assumption, the number of participating nodes does not decrease in round r . So this is impossible.
- Suppose that in round r of phase 2.1, a participating node x sends a message m , and m is not reliably received by a node y . Then in phase 2.2, x learns the identity of the faulty nodes, i.e. $|F_x^*| = f$ (the argument is identical to **Lemma 4.8**). Therefore, x is not a participating node in round $r + 1$. Again, by assumption, the number of participating nodes does not decrease in round r . So this is impossible.

□

Lemma 4.15. *Let u, v be non-faulty nodes, such that $val_u[v] = \perp$ at the end of phase 0.1, then in the decision phase, for any pair of non-faulty nodes x, y , we have $D_x = D_y = D$ (for some set $D \subseteq V$). Furthermore, we have, $|D| \geq 2f + 1$.*

Proof. Consider an arbitrary non-faulty node x . If there is no participating node in a round $n + 1$ of phase 2, then it must be the case that $|F_{x,n}^*| = f$ and $F_{x,n} = F_{x,n}^*$. Therefore in round n of phase 2.3, x sets $K_{x,n} = V$. Consider the decision phase. By **Corollary 4.13**, x reliably receives messages from every node in the decision phase. Hence, x computes

$$D_x = \cap_{y \notin F_x} K[x][y] \tag{4.5}$$

$$= \cap_{y \notin W} K[x][y] \tag{4.6} \quad (\text{By } \mathbf{Lemma 4.12}, \text{ since } |F_x^*| = f)$$

$$= \cap_{y \notin W} K_y \tag{4.7}$$

$$= \cap_{y \notin W} V \tag{4.8}$$

$$= V \tag{4.9}$$

By assumption the graph G has at least $2f + 1$ nodes. Hence, $|D_x| \geq 2f + 1$.

Now let us suppose that there is at least one participating node in round $n + 1$ of phase 2. Since there are n rounds of phase 2, if the number of participating node decreases in every round, then there would be no participating nodes in round $n + 1$ of phase 2. Since we are assuming that there is at least one participating node in round $n + 1$ of phase 2, there must be a round $1 \leq r \leq n$, in which the number of participating nodes stays the same.

Let x and x' be participating nodes in round r of phase 2. We will show that $K_{x,r} = K_{x',r}$, by showing that $K[x][y] = K[x'][y]$ for all nodes y . For any node y , there are three possible cases.

- *Node y is a faulty node* : In this case, by **Corollary** 4.13, every non-faulty node reliably receives $K_{y,r-1}$. By **Lemma** 4.14, $K_{y,r-1} \supseteq (F_z \cup \{z\})$ for all nodes z that are participating in round r of phase 2. Hence, we have $K[x][y] = K[x'][y] = K_{y,r-1}$.
- *Node y is a participating node in round r of phase 2* : By **Lemma** 4.14, every participating node in round r of phase 2 reliably receives $K_{y,r-1}$. By **Lemma** 4.12, $K_{y,r-1} \supseteq (F_z \cup \{z\})$ for all participating nodes z in phase 2. Hence, we have $K[x][y] = K[x'][y] = K_{y,r-1}$.
- *Node y is a non-faulty node that is not participating in round r of phase 2* : In this case, node y does not send any message with the tag **Kset_y**. Therefore, no non-faulty node z reliably receives any message with the tag **Kset_y**. By the update rules in phase 2, we have $K[x][y] = K[x'][y] = V$.

Then,

$$K_{x,r} = \cap_{y \in V} K[x][y]_r \tag{4.10}$$

$$= \cap_{y \in V} K[x'][y]_r \quad (\text{Because } K[x][y]_r = K[x'][y]_r) \tag{4.11}$$

$$= K_{x',r} \tag{4.12}$$

Therefore, $K_{x,r} = K_{x',r}$.

Now consider round $r + 1$ of phase 2. We will show that for any nodes x, x' that remain participating nodes in round $r + 2$ of phase 2, we have $K_{x,r+1} = K_{x',r+1}$. For any node y , there are again three cases.

- *Node y is a faulty node* : In this case, by **Corollary** 4.13, every non-faulty node reliably receives $K_{y,r}$. Since nodes x and x' remain participating nodes in round $r + 2$ of phase

2, it must be the case that $K_{y,r} \supseteq (F_{x,r} \cup \{x\})$ and $K_{y,r} \supseteq (F_{x',r} \cup \{x'\})$. Hence, by the update rules in phase 2, $K[x][y] = K[x'][y] = K_{y,r}$.

- *Node y is a participating node* : In this case, $K_{y,r} = K_{x,r} = K_{x',r}$, as argued earlier. Nodes x and x' either reliably receive $K_{y,r}$ they do not. In either case, $K[x][y]_r \supseteq K_{x,r}$ and $K[x'][y]_r \supseteq K_{x',r}$.
- *Node y is a non-faulty node that is not participating in round r of phase 2* : As argued earlier, in this case $K[x][y]_r = K[x'][y]_r = V$.

And, we have

$$K_{x,r+1} = \bigcap_{y \in V} K[x][y]_{r+1} \quad (4.13)$$

$$= K_{x,r} \cap_{y \in W} K[x][y]_r \quad (\text{for } y \notin W, K[x][y]_r \supseteq K_{x,r}) \quad (4.14)$$

$$= K_{x',r} \cap_{y \in W} K[x][y]_r \quad (K_{x,r} = K_{x',r}) \quad (4.15)$$

$$= K_{x',r} \cap_{y \in W} K[x'][y]_r \quad (\text{for } y \in W, K[x][y]_r = K[x'][y]_r) \quad (4.16)$$

$$= K_{x',r+1} \quad (4.17)$$

We can now repeat this argument inductively, and say that for all nodes x, x' that are participating in round $n + 1$ of phase 2, $K_{x,n} = K_{x',n} = K$ (for some set $K \subseteq V$).

By **Lemma 4.12**, $K \supseteq (F_{x,0} \cup \{x\})$. By **Lemma 4.10**, $|F_{x,0}| = 2f$. Therefore, $|K| \geq 2f + 1$. For a node z , if $K_z = K$ in the decision phase, then whether or not z reliably receives any other message or not, z computes K as the decision set D_z . Otherwise if $K_z \neq K$, it must be the case that z was not a participating node in round $n + 1$ of phase 2. Hence, $|F_{z,n}| = f$ and $K_{z,n} = V$. Furthermore by **Corollary 4.13**, z can reliably receive messages from every node. Therefore, z reliably receives K in the decision phase from some node $x' \notin F_x$ and computes $D_z = K$. \square

Theorem 4.16. *The algorithm described in 4.1 correctly achieves consensus.*

Proof. Termination is trivially satisfied, since the algorithm runs a bounded number of steps. For validity and agreement, we consider two cases.

- *There exist non-faulty nodes u, v such that u does not reliably receive the input value of v in phase 0.1* : In this case, at the end of phase 0.1, we have, $val_u[v] = \perp$. Hence, by **Lemma 4.15**, all non-faulty nodes compute the same decision set D , and $|D| \geq 2f + 1$.
- *For all non-faulty nodes u, v , node u reliably receives the input value of node v* : Consider an arbitrary node z . There are two cases.

- *Node z is a participating node in round $n + 1$ of phase 2 :* Since all nodes reliably receive all values in phase 0.1, we have $K_{z,0} = V$. Now we will show $F_{z,n} = V - \{z\}$ and $K_{z,n} = V$. Note that the set F_z is only updated in phase 0, if either z does not reliably receive an input value, or the size of F_z^* becomes f . Since, by assumption, all nodes reliably receive all input values in phase 0.1, and z is a participating node in round $n + 1$ of phase 2, neither of these conditions is true. Therefore, the set F_z is not updated in phase 0. Furthermore, the set F_z is only updated in phase 1, or phase 2, if the size of the set F_z^* becomes f . Since z is a participating node in round $n + 1$, by definition, we have $|F_{z,n}^*| < f$. Therefore, the set F_z is not updated in phase 1, and in any round of phase 2. Hence, $F_{z,n} = V$. Now since z is a participating node in round $n + 1$ of phase 2, in any round r , if z reliably received a message $(K_{set_y}, K_{y,r-1})$, then by **Lemma 4.14**, it must be the case that $K_{y,r-1} \supseteq F_{z,r} \cup \{z\} = V$. Hence, $K_{z,r} = V$, for all rounds r . In particular, we have, $K_{z,n} = V$.

In the decision phase, node z computes

$$D_z = \cap_{y \notin F_{z,n}} K[z][y]_n \quad (4.18)$$

$$= \cap_{y \in \{z\}} K[z][y]_n \quad (\text{Because } F_{z,n} = V - \{z\}) \quad (4.19)$$

$$= K[z][z]_n \quad (4.20)$$

$$= V \quad (\text{As argued earlier}) \quad (4.21)$$

- *Node z is not a participating node in round $n + 1$ of phase 2 :* If z is not a participating node in round $n + 1$ of phase 2, then it must be the case that $|F_{z,n}^*| = f$, and hence by **Corollary 4.13**, z reliably receives messages from every node and sets $K_{z,n} = V$ in phase 2.3 of round n of phase 2.

Therefore, for every non faulty node z , we have $K_z = V$ in the decision set, and hence, every node computes V as the decision set.

So in every execution of the algorithm, all nodes compute the same decision set D , and $|D| \geq 2f + 1$.

Now if x is any node in V , and $val_u[x] \neq \perp$ and $val_v[x] \neq \perp$, then this must mean that u and v both reliably receive the input value of x . Hence, $val_u[x] = val_v[x] = val_x[x]$, and every node computes the same multiset $S = \{val_v[v] \mid v \in D\}$. Since S contains $2f + 1$ values, the majority value in S has to be the value of a non-faulty node, so the algorithm satisfies validity. Since every node computes the decision using the same steps, agreement is satisfied too.

□

We showed that Byzantine consensus is achievable under our model on any arbitrary graph that is $2f$ -connected, and has at least $2f + 1$ nodes. The algorithm presented in this chapter can also be used for the case of a single fault, but we mention both algorithms separately, as we think they illustrate different techniques to tackle the problem.

Chapter 5: Other Results

In this chapter we present two other results without proof. Our first claim is a necessity claim about the connectivity of the underlying communication graph.

Claim 5.1. *Let $G = (V, E)$ be any arbitrary undirected graph that is at most $\lfloor \frac{3f}{2} \rfloor$ -connected. Then there exists no algorithm for achieving consensus on graph G , under the presence of f faulty nodes.*

We believe that an argument similar to the proof of **Lemma 2.3** can be made for this claim as well. Our next claim says that for even numbers f , there exists a graph with connectivity $\frac{3f}{2} + 1$, on which consensus is achievable.

Claim 5.2. *Let $G_1 = G_2 = K_{f/2}$, and $G_3 = K_{3f/2+1}$, where K_n is the complete graph on n nodes. Define G to be the union of G_1, G_2, G_3 with additional edges between every pair of nodes in G_1 and G_3 , and every pair of nodes in G_2 and G_3 . We claim that there exists an algorithm that achieves consensus on graph G .*

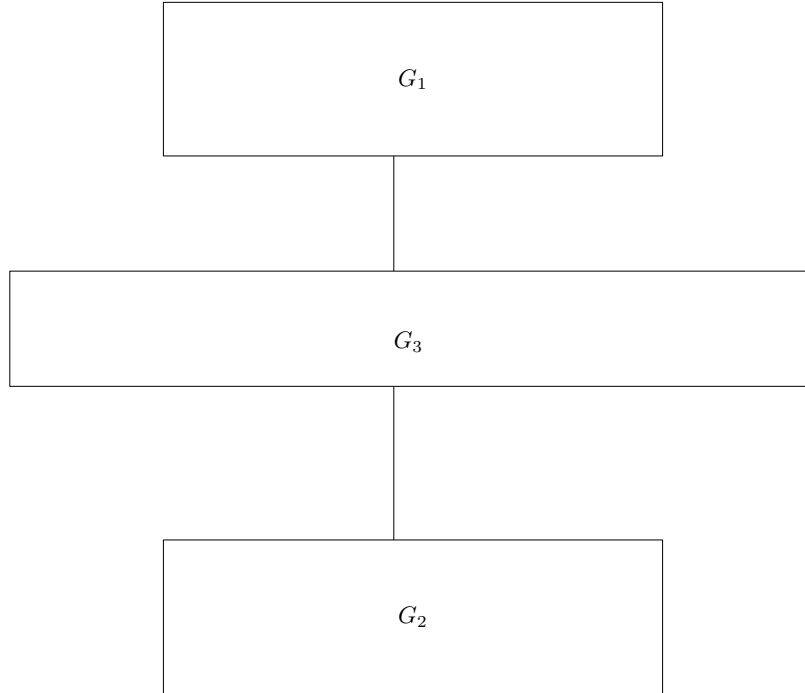


Figure 5.1: $(\frac{3f}{2} + 1)$ -connected graph G

The graph G in the above claim is shown in Figure 5.1. Each block represents one of the cliques G_1, G_2, G_3 , and an edge between two blocks represents an edge between every pair of nodes in the two subgraphs.

Chapter 6: Conclusion

We studied the problem of exact consensus in directed and undirected graphs with local-broadcast channels, under the presence of Byzantine faults nodes. We proved conditions on the degree and connectivity of the graphs under which consensus is impossible. We also showed two different algorithms for solving consensus under the presence of f Byzantine faulty nodes. Both algorithms illustrate different algorithmic and proof techniques for attacking the problem of consensus. We show that under our model, binary-valued consensus is achievable on any graph that is $2f$ -connected, and that no algorithm exists for achieving consensus on a graph that is at most f -connected. In contrast, under the point-to-point model even binary-valued consensus is impossible on any graph with connectivity less than $2f + 1$ or nodes less than $3f + 1$. We also show that under our model consensus is impossible on any graph in which the minimum node degree is less than $2f$. We believe that there exist graphs with connectivity at most $\lfloor \frac{3f}{2} \rfloor + 1$ on which binary-valued consensus is achievable. However, the following problems remain open.

Problem 6.1. *Does there exist an algorithm for achieving binary-valued consensus on an arbitrary graph that is $(\lfloor \frac{3f}{2} \rfloor + 1)$ -connected and has minimum degree $2f$.*

Problem 6.2. *Does there exist an algorithm for achieving multi-valued consensus on arbitrary graph with connectivity $2f$.*

References

- [1] M. J. Fischer, N. A. Lynch, and M. Merritt, “Easy impossibility proofs for distributed consensus problems,” *Distributed Computing*, vol. 1, no. 1, pp. 26–39, 1986.
- [2] A. Clement, F. Junqueira, A. Kate, and R. Rodrigues, “On the (limited) power of non-equivocation,” in *Proceedings of the 2012 ACM symposium on Principles of distributed computing*. ACM, 2012, pp. 301–308.
- [3] L. Tseng and N. H. Vaidya, “Fault-tolerant consensus in directed graphs,” in *Proceedings of the 2015 ACM Symposium on Principles of Distributed Computing*. ACM, 2015, pp. 451–460.
- [4] P. Bansal, P. Gopal, A. Gupta, K. Srinathan, and P. K. Vasishta, “Byzantine agreement using partial authentication,” in *International Symposium on Distributed Computing*. Springer, 2011, pp. 389–403.
- [5] C.-Y. Koo, V. Bhandari, J. Katz, and N. H. Vaidya, “Reliable broadcast in radio networks: The bounded collision case,” in *Proceedings of the twenty-fifth annual ACM symposium on Principles of distributed computing*. ACM, 2006, pp. 258–264.
- [6] H. Zhang and S. Sundaram, “Robustness of information diffusion algorithms to locally bounded adversaries,” in *American Control Conference (ACC), 2012*. IEEE, 2012, pp. 5855–5861.
- [7] D. Ravikant, V. Muthuramakrishnan, V. Srikanth, K. Srinathan, and C. P. Rangan, “On byzantine agreement over $(2, 3)$ -uniform hypergraphs,” in *International Symposium on Distributed Computing*. Springer, 2004, pp. 450–464.
- [8] A. Jaffe, T. Moscibroda, and S. Sen, “On the price of equivocation in byzantine agreement,” in *Proceedings of the 2012 ACM symposium on Principles of distributed computing*. ACM, 2012, pp. 309–318.
- [9] C. Li, M. Hurfin, Y. Wang, and L. Yu, “Towards a restrained use of non-equivocation for achieving iterative approximate byzantine consensus,” in *Parallel and Distributed Processing Symposium, 2016 IEEE International*. IEEE, 2016, pp. 710–719.
- [10] M. Backes, F. Bendun, A. Choudhury, and A. Kate, “Asynchronous mpc with a strict honest majority using non-equivocation,” in *Proceedings of the 2014 ACM symposium on Principles of distributed computing*. ACM, 2014, pp. 10–19.
- [11] T. Böhme, F. Göring, and J. Harant, “Menger’s theorem,” *Journal of Graph Theory*, vol. 37, no. 1, pp. 35–36, 2001.