FEDERATED BYZANTINE AGREEMENT SYSTEM AND STELLAR CONSENSUS PROTOCOL

HIDENORI SHINOHARA

Contents

1. Prerequisites

Definition 1.0.1 (Network). The network delivers any messages among nodes connected to it, but it may arbitrarily delay or reorder messages.

Definition 1.0.2 (Well-behaved). A node is considered well-behaved if and only if they follow the protocol and is connected to the network.

Remark 1.0.3. A few things that may make it easier to understand FBA and SCP:

- V and v are often used as the set of nodes and a node because graph theorists often call nodes vertices. Also, n is a common variable name for a natural number, so it is often convenient to use v for a vertex.
- Pay particular attention to the difference between quorums and quorum slices. They sound very similar, and they are related, but they are different.
- B is often used as a subset of nodes. The letter B is used because when we think of a subset, we are often trying to model the set of befouled nodes.
- q and Q are often used for *quorum slices*, instead of quorums. The function Q is for *quorum slices*, not quorums. The letter q is often used as a *quorum slice*, and the letter U is often used as a *quorum*. I have no idea why that is the case.
- The letter x is often used as a value, and the letter a is often used as a statement.

2. Federated Byzantine Agreement System

2.1. Quorums.

Definition 2.1.1 (Federated Byzantine Agreement System). Let V be a set and $Q : V \to 2^{2^V} \setminus \{\emptyset\}$ be a function such that $\forall v \in V, \forall q \in Q(v), v \in q$. Then we call the pair $\langle V, Q \rangle$ a federated Byzantine agreement system, or FBAS for short. Each q in $Q(v)$ is called a quorum slice for each $v \in V$.

Remark 2.1.2.

- For each node v, $Q(v)$ is a set of subsets of V. For instance, node v_1 may trust v_2, v_3, v_4 and may have $\{v_1, v_2, v_3, v_4\} \in Q(v_1) \subset 2^V$.
- Note that we explicitly exclude $\emptyset$ from the co-domain. In other words, we want $Q(v) \neq \emptyset$ for all $v \in V$. If $Q(v) = \emptyset$ for some $v \in V$, it satisfies $\forall q \in Q(v), v \in q$. As we will see, each $q \in Q(v)$ is a list of nodes that v trusts. If v has no list of nodes that it trusts, v cannot really do anything. Thus we want $Q(v) \neq \emptyset$ for all $v \in V$.
- Consider the case when $V = \emptyset$. Note that $2^\emptyset = \{\emptyset\}$ and thus $2^{2^\emptyset} = \{\emptyset, \{\emptyset\}\}$. Thus V forms an FBAS where Q is the map $\emptyset \mapsto \{\{\emptyset\}\}$.

Definition 2.1.3 (Quorum). Let $\langle V, Q \rangle$ be an FBAS. $U \subset V$ is called a quorum if and only if $U \neq \emptyset$ and $\forall v \in U, \exists q \in Q(v), q \subset U$.

Theorem 2.1.4. *In an FBAS $\langle V, Q \rangle$, the union of two quorums is a quorum.*

Proof. Let U_1, U_2 be two quorums. Let $v \in U_1 \cup U_2$. Then $v \in U_i$ for $i = 1$ or $i = 2$. Then $q \subset U_i$ for some $q \in Q(v)$. Therefore, $q \subset U_1 \cup U_2$, so $U_1 \cup U_2$ is indeed a quorum. □

Corollary 2.1.5. *The set of quorums of a given FBAS is closed under union.*

Theorem 2.1.6. *In an FBAS $\langle V, Q \rangle$, V is a quorum.*

Proof. For any $v \in V$, for any $q \in Q(v)$, $q \subset V$. Therefore, V is indeed a quorum. □

Example 2.1.7. One might wonder if the intersection of quorums is always a quorum. However, this is not true in general.

Let $V = \{v_1, \ldots, v_4\}$ and

- $Q(v_1) = \{\{v_1, v_2, v_3\}, \{v_1, v_2, v_4\}, \{v_1, v_3, v_4\}\}$,
- $\vdots$
- $Q(v_4) = \{\{v_1, v_2, v_4\}, \{v_1, v_3, v_4\}, \{v_2, v_3, v_4\}\}$.

In other words, $Q(v_i) = \{U \subset V \mid |U| = 3, v_i \in U\}$.

Then $U_1 = \{v_1, v_2, v_3\}$ is a quorum, and $U_2 = \{v_2, v_3, v_4\}$ is a quorum. However, $U_1 \cap U_2 = \{v_2, v_3\}$ is not a quorum because the size of any quorum slice is 3.

Although the intersection of two quorums is not a quorum in general, we do care whether the intersection of two quorums is empty.

Definition 2.1.8 (Quorum Intersection). Let $\langle V, Q \rangle$ be an FBAS. We say $\langle V, Q \rangle$ enjoys quorum intersection if and only if $U_1 \cap U_2 \neq \emptyset$ for any pair of quorums U_1, U_2.

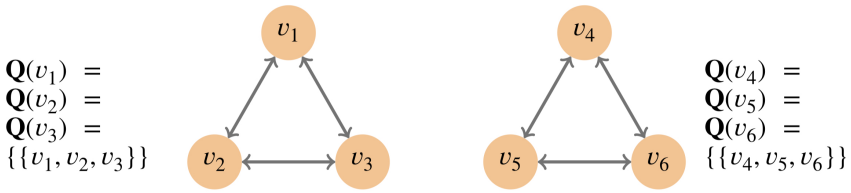

FIGURE 1. FBAS Without Quorum Intersection (Figure 6, White Paper)

Remark 2.1.9. Figure 1 is an example of an FBAS where v_1, v_2, v_3 trust each other, and v_4, v_5, v_6 trust each other. Notice that this FBAS does not enjoy quorum intersection because $\{v_1, v_2, v_3\}, \{v_4, v_5, v_5\}$ are quorums without any intersection. Even though we have not defined how voting works, it is not hard to see that v_1, v_2, v_3 might "go with" a statement while v_4, v_5, v_6 "go with" a contradictory statement. For instance, consider a case where a malicious user with \$100 in their account tries to double spend their money as following:

- The malicious user tells v_1, v_2, v_3 that they are sending the \$100 to Person A. The nodes have no reason to doubt the transaction, so they will "go with" it. Person A thinks that they received \$100, so they give the malicious user \$100 worth of products.
- Similarly, the malicious user tells v_4, v_5, v_6 that they are sending the \$100 to Person B. Again, the nodes have no reason to doubt the transaction, so they will "go with" it. Person B thinks that they received \$100, so they give the malicious user \$100 worth of products.

The malicious user should not be able to obtain \$200 worth of products by spending their \$100 twice. However, the network would not be able to prevent this due to the way each node chose its quorum slice. This example shows how quorum intersection is related to safety.

I have not defined safety yet!

Definition 2.1.10 (Delete). Let $\langle V, Q\rangle$ be an FBAS and $B \subset V$. Then the FBAS $\langle V, Q\rangle^B$ is defined to be $\left\langle V \setminus B, Q^B\right\rangle$ where $\forall v \in V \setminus B, Q^B(v) = \{q \setminus B \mid q \in Q(v)\}$.

Remark 2.1.11. One may think that this is related to fail-stop behaviors where B is the set of nodes that stopped responding. In general, however, this is not true as we also remove nodes from quorum slices. One can think of this as the *alternate* universe where the nodes from B simply did not even exist from the beginning.

Theorem 2.1.12. *Definition 2.1.10 is well-defined. In other words, if $\langle V, Q\rangle$ is an FBAS and $B \subset V$, then $\langle V, Q\rangle^B$ is an FBAS.*

Proof. If $V = B$, then we are done because of Remark 2.1.2.

Suppose otherwise. Let $v \in V \setminus B$ be given. By the definition of an FBAS, $Q(v) \neq \emptyset$. Therefore, $Q^B(v) = \{q \setminus B \mid q \in Q(v)\}$ is nonempty.

Let $q' \in Q^B(v)$ be given arbitrarily. Then $q' = q \setminus B$ for some $q \in Q(v)$. Then $v \in q$ by the definition of an FBAS, so $v \in q'$.

Thus Q^B is indeed a quorum function. Therefore, $\langle V, Q\rangle^B$ is an FBAS. □

Theorem 2.1.13. *Let U be a quorum in FBAS $\langle V, Q\rangle$, let $B \subset V$ be a set of nodes, and let $U' = U \setminus B$. If $U' \neq \emptyset$, then U' is a quorum in $\langle V, Q\rangle^B$.*

Proof. Since $U' \neq \emptyset$, it suffices to show that $\forall v \in U', \exists q \in Q^B(v), q \subset U'$. Let $v \in U'$. Then $v \in U$. Since U is a quorum in $\langle V, Q \rangle$, we can find $q \in Q(v)$ such that $q \subset U$. Then $q' = q \setminus B \in Q^B(v)$, and $q' = q \setminus B \subset U \setminus B = U'$. Therefore, U' is a quorum in $\langle V, Q \rangle^B$. □

Remark 2.1.14. One can think of this theorem as "*A quorum in the 'original' universe is a quorum in the 'alternate' universe.*"

Definition 2.1.15 (Quorum Intersection Despite B). Let $\langle V, Q \rangle$ be an FBAS and $B \subset V$ be a set of nodes. We say $\langle V, Q \rangle$ enjoys quorum intersection despite B if and only if $\langle V, Q \rangle^B$ enjoys quorum intersection.

Remark 2.1.16. Quorum intersection despite B is related to the system-level safety when nodes in B act arbitrarily.

Insert Figure 7 from the white paper.

Definition 2.1.17 (Quorum Availability Despite B). Let $\langle V, Q \rangle$ be an FBAS and $B \subset V$ be a set of nodes. We say $\langle V, Q \rangle$ enjoys quorum availability despite B if and only if $V \setminus B$ is a quorum in $\langle V, Q \rangle$ or $B = V$.

Theorem 2.1.18. *Let $\langle V, Q \rangle$ be an FBAS and $B \subset V$. $\langle V, Q \rangle$ enjoys quorum availability despite B if and only if $\forall v \in V \setminus B$, there exists a quorum U_v such that $v \in U_v \subset (V \setminus B)$.*

Proof. If $V = B$, we are done. Suppose otherwise.

$$\begin{aligned} \forall v \in V \setminus B, \exists \text{a quorum } U_v, v \in U_v \subset (V \setminus B) &\implies \bigcup_{v \in V \setminus B} U_v \text{ is a quorum in } \langle V, Q \rangle \\ &\implies V \setminus B \text{ is a quorum in } \langle V, Q \rangle \end{aligned}$$

by Theorem 2.1.4. On the other hand, if $V \setminus B$ is a quorum in $\langle V, Q \rangle$, then $\forall v \in V \setminus B, \exists$a quorum $U_v, v \in U_v \subset (V \setminus B)$ because we can let $U_v = V \setminus B$ for each v. □

Remark 2.1.19. Theorem 2.1.18 shows that $\langle V, Q \rangle$ enjoys quorum availability despite B if all nodes in $V \setminus B$ can find a quorum without B. This is related to the liveness of the system. If $\langle V, Q \rangle$ enjoys quorum availability despite B, then regardless of what happens to nodes in B, nodes in $V \setminus B$ can keep going.

Definition 2.1.20 (v-blocking). Let $\langle V, Q \rangle$ be an FBAS. Let $v \in V$. A subset $B \subset V$ is called v-blocking if and only if $\forall q \in Q(v), q \cap B \neq \emptyset$.

Remark 2.1.21. Intuitively, if a subset $B \subset V$ is v-blocking, then one may think of it as "*v can't really get by without B.*" The following theorem can be interpreted as "*If v can't get by without B, v can't get by without C for any $C \supset B$.*"

Theorem 2.1.22. *Let $\langle V, Q \rangle$ be an FBAS. Let $v \in V$. Then*

- *The union of two v-blocking sets is v-blocking.*
- *Any superset of a v-blocking set is v-blocking.*

Proof. It suffices to only prove the second statement. If $B \subset B'$ and B is v-blocking, $q \cap B' \supset q \cap B \neq \emptyset$ for any $q \in Q(v)$. □

Theorem 2.1.23. *Let $\langle V, Q \rangle$ be an FBAS. Let $A \subsetneq V$ and U_1, U_2 be a partition of $V \setminus A$. Let $v \in U_1$. If U_2 is not v-blocking in $\langle V, Q \rangle$, then U_2 is not v-blocking in $\langle V, Q \rangle^A$.*

Proof. Since U_2 is not v-blocking in $\langle V, Q\rangle$, there exists $q_v \in Q(v)$ such that $q_v \cap U_2 = \emptyset$.

$$\begin{aligned}(q_v \setminus A) \cap U_2 &= (q_v \cap U_2) \setminus (A \cap U_2) \\ &= q_v \cap U_2 = \emptyset.\end{aligned}$$

Thus U_2 is not v-blocking in $\langle V, Q\rangle^A$. □

Theorem 2.1.24. *Let $\langle V, Q\rangle$ be an FBAS. Let $B \subset V$. $\langle V, Q\rangle$ enjoys quorum availability despite B if and only if B is not v-blocking for any $v \in V \setminus B$.*

Proof.

$$\begin{aligned}\forall v \in V \setminus B, \neg(B \text{ is } v\text{-blocking}) &\iff \forall v \in V \setminus B, \neg(\forall q \in Q(v), q \cap B \neq \emptyset) \\ &\iff \forall v \in V \setminus B, \exists q \in Q(v), q \cap B = \emptyset \\ &\iff \forall v \in V \setminus B, \exists q \in Q(v), q \subset V \setminus B \\ &\iff V = B \text{ or } V \setminus B \text{ is a quorum in } \langle V, Q\rangle \\ &\iff \langle V, Q\rangle \text{ enjoys quorum availability despite } B\end{aligned}$$

□

2.2. **Dispensable Sets.**

Definition 2.2.1 (Dispensable Set). Let $\langle V, Q\rangle$ be an FBAS and $B \subset V$ be a set of nodes. B is called a dispensable set, or DSet, if and only if $\langle V, Q\rangle$ enjoys both quorum intersection despite B and quorum availability despite B.

We will first show some basic properties of DSets.

Theorem 2.2.2. *Let $\langle V, Q\rangle$ be an FBAS. Then*

- *V is a DSet.*
- *If $\forall v \in V, Q(v) = \{V\}$, then $\emptyset$ and V are the only DSets of $\langle V, Q\rangle$.*

Proof.

- $\langle V, Q\rangle^V$ enjoys quorum intersection because there is no quorum. $\langle V, Q\rangle$ enjoys quorum availability despite V because $V = V$.
- Suppose $\forall v \in V, Q(v) = \{V\}$. As shown above, V is a DSet of $\langle V, Q\rangle$. The empty set is a DSet because
  - $\langle V, Q\rangle$ enjoys quorum intersection despite $\emptyset$ because the only quorum is V.
  - $\langle V, Q\rangle$ enjoys quorum availability despite B because $V \setminus \emptyset = V$ is a quorum.

  Let $\emptyset \subsetneq S \subsetneq V$ be given. Then $V \neq S$ and $V \setminus S$ is not a quorum for it is nonempty and does not contain any quorum slice. Therefore, $\langle V, Q\rangle$ does not enjoy quorum availability despite B, so no nonempty, proper subset of V is a DSet.

□

Definition 2.2.3 (Intact and Befouled). Let $\langle V, Q\rangle$ be an FBAS and $v \in V$. v is said to be intact if and only if there exists a DSet B containing all ill-behaved nodes and $v \notin B$. v is said to be befouled if and only if v is not intact.

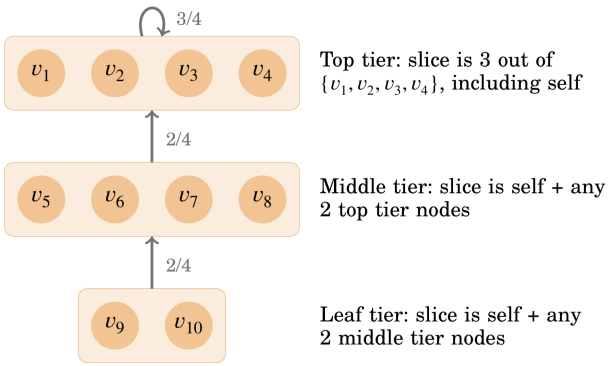

FIGURE 2. Tiered Quroum Example (P.5 of the white paper)

Example 2.2.4.

proofread!

We will use Figure 2 as an example.

- The smallest DSet containing v_5, v_6 in Figure 2 is $\{v_5, v_6, v_9, v_{10}\}$.

– First, we will show that $B = \{v_5, v_6\}$ is not a DSet. By definition,

$$Q^B(v_9) = \{\{v_9\}, \{v_9, v_7\}, \{v_9, v_8\}, \{v_9, v_7, v_8\}\}$$
$$Q^B(v_{10}) = \{\{v_{10}\}, \{v_{10}, v_7\}, \{v_{10}, v_8\}, \{v_{10}, v_7, v_8\}\}$$

This implies that $U_9 = \{v_9\}$ and $U_{10} = \{v_{10}\}$ are both quorums. Then $U_9 \cap U_{10} = \emptyset$, so $\langle V, Q\rangle^B$ does not enjoy quorum intersection. Therefore, B is not a DSet.
Next, we will consider $C = \{v_5, v_6, v_1\}$. Then we can use the same argument as above. $U_9 = \{v_9\} \in Q^C(v_9)$ and $U_{10} = \{v_{10}\} \in Q^C(v_{10})$, and the intersection is empty. Therefore, C is not a DSet. It is easy to see that this argument works for the case of $\{v_5, v_6, v_i\}$ for any $i = 1, 2, 3, 4$.
We will consider $D = \{v_5, v_6, v_9\}$. Similarly, $U = \{v_{10}\} \in Q^D(v_{10})$ is a quorum. Moreover, $U' = \{v_1, v_2, v_3, v_4\}$ is a quorum. Then $U \cap U' = \emptyset$, so $\langle V, Q\rangle^D$ does not enjoy quorum intersection. It is easy to see that a similar argument shows that $\{v_5, v_6, v_{10}\}$ is not a DSet.
Finally, we will show that $E = \{v_5, v_6, v_9, v_{10}\}$ is a DSet. $V \setminus E$ is a quorum in $\langle V, Q\rangle$ because every node in $V \setminus E$ has a quorum slice consisting of nodes in $V \setminus E$. If a quorum in $\langle V, Q\rangle^E$ contains v_7 or v_8, then it must contain some of v_1, v_2, v_3, v_4. If a quorum in $\langle V, Q\rangle^E$ contains at least one of v_1, v_2, v_3, or v_4, then it must contain at least three of v_1, v_2, v_3, v_4. Therefore, any intersection of two quorums in $\langle V, Q\rangle^E$ contains at least two of v_1, v_2, v_3, v_4 by the pigeon hole principle.
Therefore, E is indeed a smallest DSet containing v_5 and v_6.

– We showed that $B = \{v_5, v_6\}$ is not a DSet because $\langle V, Q\rangle$ does not enjoy quorum intersection despite B. What this means is that if both v_5 and v_6 either stop responding or are malicious, then it is not possible to guarantee safety for v_9 and v_{10}. For instance, consider the following situation:
  * Both v_5 and v_6 tell v_9 and v_{10} that $Q(v_5) = Q(v_6) = \{\{v_5, v_6\}\}$ convincing that $\{v_5, v_6, v_9\}$ and $\{v_5, v_6, v_{10}\}$ are both quorums.
  * Both v_5 and v_6 tell v_9 that they want to process a certain transaction. This transaction does not contradict what v_9 knows about v_5. Moreover, everyone in the quorum $\{v_5, v_6, v_9\}$ is in favor of this transaction. Thus there is no reason for v_9 to not believe this transaction.
  * Both v_5 and v_6 tell v_{10} that they want to process a certain transaction that contradicts the transaction they told v_5 about. For the same reason, there is no reason for v_{10} to not believe this transaction.
  * Then the network processes contradicting transactions. This can let v_5 double-spend some money, for instance.
  * One can verify that this is possible by looking into the definition of accepting, confirming and such that are introduced in later chapters.

- $B = \{v_1\}$ is a DSet.
  – First, we will check if $\langle V, Q\rangle$ enjoys quorum intersection despite B.
  Consider $\langle V, Q\rangle^B$. Any quorum containing v_9 and/or v_{10} must contain at least two of v_5, v_6, v_7, v_8. Any quorum containing at least one of $v_5, \cdots, v_8$ must contain at least one of v_2, v_3, v_4. Any quorum containing at least one of v_2, v_3, v_4 must

contain at least two of v_2, v_3, v_4. This is because $Q(v_i)^B = \{\{v_2, v_3, v_4\}, \{v_i, v_j\}, \{v_i, v_k\}\}$ where $\{i, j, k\} = \{2, 3, 4\}$.
Therefore, the intersection of any two quorums must contain at least one of v_2, v_3, v_4 by the pigeon hole principle.
Next, we need to check if $\langle V, Q\rangle$ enjoys quorum availability despite B. $V \setminus B$ is indeed a quorum in $\langle V, Q\rangle$ because each node in $V \setminus B$ has a quorum slice that does not contain v_1.

– We showed that B is indeed a DSet. What this means is that even if v_1 stops responding or becomes malicious, the rest of the network can make progress safely. For instance, suppose that v_1 becomes malicious and tries to double-spend money. v_1 might tell v_5 that it wants to process a certain transaction. Similarly, v_1 might tell v_6 that it wants to process a contradicting transaction. However, every quorum slice of v_5 and v_6 contains at least one tier-1 node that is not v_1. Suppose that v_5 asks v_2 what it thinks, and v_6 asks v_3 what it thinks. Then every quorum slice of v_2 and v_3 contains 3 tier-1 nodes. By the pigeon hole principle, at least one tier-1 node that is not v_1 gets asked what it thinks about the contradicting transactions from v_5. The tier-1 node does not agree with them and v_1's attempt to double-spend money fails.

Theorem 2.2.5. *If B_1 and B_2 are DSets in an FBAS $\langle V, Q\rangle$ enjoying quorum intersection, then $B = B_1 \cap B_2$ is a DSet, too.*

Proof. If $B_1 = V$ or $B_2 = V$, then we are done. Suppose otherwise.

First, we will show that $\langle V, Q\rangle$ enjoys quorum availability despite B. By Definition 2.1.17, it suffices to show that $V = B$ or $V \setminus B$ is a quorum in $\langle V, Q\rangle$. Since we assumed that $B_1 \neq V$ and $B_2 \neq V$, $B \neq V$. Therefore, we will show that $V \setminus B$ is a quorum in $\langle V, Q\rangle$. By basic set theory, $V \setminus B = V \setminus (B_1 \cap B_2) = (V \setminus B_1) \cup (V \setminus B_2)$. Since B_1 is a DSet, $V = B_1$ or $V \setminus B_1$ is a quorum in $\langle V, Q\rangle$. Since we assumed that $V \neq B_1$ earlier, $V \setminus B_1$ is a quorum in $\langle V, Q\rangle$. Similarly, $V \setminus B_2$ is a quorum in $\langle V, Q\rangle$. By Theorem 2.1.4, the union $(V \setminus B_1) \cup (V \setminus B_2) = V \setminus B$ is a quorum in $\langle V, Q\rangle$.

Next, we will show that $\langle V, Q\rangle$ enjoys quorum intersection despite B. Let U_a, U_b be quorums in $\langle V, Q\rangle^B$. We want to show that $U_a \cap U_b \neq \emptyset$. We will do so by proving a stronger statement, which is $(U_a \cap U_b) \setminus B_1 \neq \emptyset$. In other words, we will show that $(U_a \setminus B_1) \cap (U_b \setminus B_1) \neq \emptyset$.

Since B_1 is a DSet, $\langle V, Q\rangle$ enjoys quorum intersection despite B_1. In other words, $\langle V, Q\rangle^{B_1}$ enjoys quorum intersection. Therefore, it suffices to show that $U_a \setminus B_1$ and $U_b \setminus B_1$ are both quorums in $\langle V, Q\rangle^{B_1}$. By Theorem 2.1.13, $U_a \setminus B_1$ and $U_b \setminus B_1$ are quorums in $(\langle V, Q\rangle^B)^{B_1}$ if $U_a \setminus B_1 \neq \emptyset$ and $U_b \setminus B_1 \neq \emptyset$. Since $(\langle V, Q\rangle^B)^{B_1} = \langle V, Q\rangle^{B_1}$, it suffices to show that $U_a \setminus B_1 \neq \emptyset$ and $U_b \setminus B_1 \neq \emptyset$.

We will first show that $U_a \setminus B_1 \neq \emptyset$. By basic set theory,

$$\begin{aligned} U_a &= U_a \setminus B \\ &= U_a \setminus (B_1 \cap B_2) \\ &= (U_a \setminus B_1) \cup (U_a \setminus B_2) \end{aligned}$$

because $U_a \cap B = \emptyset$.

This implies that $(U_a \setminus B_1) \cup (U_a \setminus B_2) \neq \emptyset$. If $U_a \setminus B_1$ is nonempty, we are done. Suppose $U_a \setminus B_2$ is nonempty. We will show that this implies that $U_a \setminus B_1 \neq \emptyset$. We will do so by first finding two quorums in $\langle V, Q\rangle^{B_2}$ whose intersection is a subset of $U_a \setminus B_1$. Since $\langle V, Q\rangle^{B_2}$ enjoys quorum intersection, the intersection of such two quorums must be nonempty, which in turn shows that $U_a \setminus B_1$ is nonempty.

- We claim that $U_a \setminus B_2$ is a quorum in $\langle V, Q\rangle^{B_2}$. U_a is a quorum in $\langle V, Q\rangle^{B}$. Since $U_a \setminus B_2$ is assumed to be nonempty, $U_a \setminus B_2$ is a quorum in $(\langle V, Q\rangle^{B})^{B_2}$ by Theorem 2.1.13. Since $(\langle V, Q\rangle^{B})^{B_2} = \langle V, Q\rangle^{B_2}$, $U_a \setminus B_2$ is a quorum in $\langle V, Q\rangle^{B_2}$.
- $V \setminus B_2 \neq \emptyset$ is a quorum in $\langle V, Q\rangle$ because $\langle V, Q\rangle$ must enjoy quorum availability despite B_2 and $B_2 \neq V$. Similarly, $V \setminus B_1 \neq \emptyset$ is a quorum in $\langle V, Q\rangle$. Because $\langle V, Q\rangle$ enjoys quorum intersection, $(V \setminus B_1) \cap (V \setminus B_2) \neq \emptyset$. In other words, $(V \setminus B_1) \setminus B_2 \neq \emptyset$. By applying Theorem 2.1.13 to the fact that $V \setminus B_1$ is a quorum in $\langle V, Q\rangle$ and $(V \setminus B_1) \setminus B_2 \neq \emptyset$, we can conclude that $(V \setminus B_1) \setminus B_2$ is a quorum in $\langle V, Q\rangle^{B_2}$.

Since $\langle V, Q\rangle^{B_2}$ enjoys quorum intersection, $(U_a \setminus B_2) \cap ((V \setminus B_1) \setminus B_2) \neq \emptyset$.

$$\begin{aligned}
\emptyset &\neq (U_a \setminus B_2) \cap ((V \setminus B_1) \setminus B_2) \\
&= (U_a \cap (V \setminus B_1)) \setminus B_2 \\
&\subset U_a \cap (V \setminus B_1) \\
&= U_a \setminus B_1.
\end{aligned}$$

Thus, $U_a \setminus B_1 \neq \emptyset$.

The same argument will show that $U_b \setminus B_1 \neq \emptyset$. □

Remark 2.2.6. Theorem 2.2.5 states that the intersection of two DSets is a DSet if the FBAS enjoys quorum intersection. One might wonder if the union of two DSets is a DSet when the FBAS enjoys quorum intersection. However, this is not true in general. Consider the FBAS $\langle V, Q\rangle$ where $V = \{v_1, v_2, v_3, v_4\}$ and $Q(v_i) = \{U \subset V \mid v_i \in U, |U| = 3\}$. This FBAS enjoys quorum intersection by the pigeon-hole principle because each quorum contains at least 3 elements. Then $B = \{v_1\}$ is a DSet because

- Quorum intersection despite B
  - Every quorum slice in $\langle V, Q\rangle^{B}$ contains at least 2 nodes because every quorum slice in $\langle V, Q\rangle$ contains exactly 3 nodes. This implies that any quorum in $\langle V, Q\rangle^{B}$ must contain at least 2 nodes. By the pigeon-hole principle, every pair of quorums in $\langle V, Q\rangle^{B}$ must intersect.
- Quorum availability despite B
  - $V \setminus B = \{v_2, v_3, v_4\}$ is a quorum in $\langle V, Q\rangle$ because $\{v_2, v_3, v_4\} \in Q(v_i)$ for each $i = 2, 3, 4$.

Similarly, $C = \{v_2\}$ is a DSet. However, $B \cup C = \{v_1, v_2\}$ is not a DSet because $B \cup C \neq V$ and $V \setminus (B \cup C) = \{v_3, v_4\}$ is not a quorum in $\langle V, Q\rangle$.

Theorem 2.2.7. *In an FBAS with quorum intersection, the set of befouled nodes is a DSet.*

Proof. Let $\langle V, Q\rangle$ be an FBAS with quorum intersection. Let B be the intersection of all DSets that contain all ill-behaved nodes. We will show that B is the set of befouled nodes by showing that $V \setminus B$ is the set of intact nodes.

$$
\begin{aligned}
v \in V \setminus B &\iff v \notin B \\
&\iff \exists \text{ DSet } B_v \text{ that contains all ill-behaved nodes and } v \notin B_v \\
&\iff v \text{ is intact}
\end{aligned}
$$

Therefore, $V \setminus B$ is precisely the set of intact nodes, and thus B is the set of befouled nodes.

By applying Theorem 2.2.5 repeatedly, we can conclude that B is a DSet. □

Corollary 2.2.8. *In an FBAS with quorum intersection, the set of intact nodes is either empty or a quorum.*

Proof. The set B of befouled nodes is a DSet by Theorem 2.2.7. By the definition of a DSet, B^c is either empty or a quorum. Therefore, the set of intact nodes, if not empty, is a quorum. □

Theorem 2.2.9. *Let $\langle V, Q \rangle$ be an FBAS and let $B \subset V$ be the set of befouled nodes. If B is a DSet, B is not v-blocking for any intact v.*

Proof. By Definition 2.2.3, a node $v \in V$ is intact if and only if $v \notin B$. By Theorem 2.1.24, $\langle V, Q \rangle$ enjoys quorum availability despite B if and only if B is not v-blocking for any $v \in V \setminus B$. Since B is a DSet, $\langle V, Q \rangle$ enjoys quorum availability despite B. Thus B is not v-blocking for any intact v. □

2.3. **Voting, Accepting, and Ratifying.**

Definition 2.3.1 (Vote). A node v votes for a statement a if and only if v asserts

- a is valid,
- a is consistent with all statements v has accepted,
- v has never voted for a statement that contradicts a,
- v promises never to vote for a statement that contradicts a in the future.

Definition 2.3.2 (Vote Against a). When a node v votes for a statement that contradicts a, we say v votes against a.

Definition 2.3.3 (Accept). Let $\langle V, Q \rangle$ be an FBAS, and let $v \in V$. v accepts a statement a if and only if

- It has never accepted a statement contradicting a, and
- It determines that either
  - There exists a quorum U such that $v \in U$ and each member of U either voted for a or broadcast that it has accepted a, or
  - There exists a v-blocking set B such that every member of B broadcast that it has accepted a.

When v accepts a, it must vote for the statement "an intact node accepted a." For simplicity, we will often write "$accept(a)$" to mean "an intact node accepted a."

Remark 2.3.4.

- As you can see, the definitions of voting and accepting have a circular dependency.
- It is possible for a node to accept a statement after voting for a contradictory statement by the second condition for accepting.
- FBAS does not tell us how to determine if two statements are contradictory. In SCP, each statement is in a specific format and that tells us how to tell if two statements are contradictory. For instance, two nomination statements $nominate(a)$ and $nominate(b)$ are not contradictory with each other regardless of what a and b are.

  This makes sense because FBAS cannot construct a general way to check whether two statements are contradictory quickly. For instance, FBAS cannot tell you if "*n is an even integer greater than 2*" and "*n is the sum of two primes*" are contradictory because it is Goldbach's conjecture which remains unproven.

Definition 2.3.5 (Ratify). A quorum U_a ratifies a statement a if and only if every member of U_a votes for a. A node v ratifies a if and only if v is a member of a quorum U_a that ratifies a.

Theorem 2.3.6. *Let $\langle V, Q \rangle$ be an FBAS. If a node $v \in V$ ratifies a statement a, then it must accept a.*

Proof. If a node v ratifies a, then it is a member of a quorum $U \subset V$ that ratifies a. Thus every member of U votes for a. This implies that v also votes for a. By Definition 2.3.1, v has never accepted a statement contradicting a. By Definition 2.3.3, v accepts a. □

Remark 2.3.7. Theorem 2.3.6 shows that ratifying is a stronger condition than accepting.

Theorem 2.3.8. *If an FBAS enjoys quorum intersection and contains no ill-behaved node, then two contradictory statements cannot be both ratified.*

Proof. Suppose the statement is false and let $a, \bar{a}$ denote two contradictory statements ratified in such an FBAS. Let $U_a, U_{\bar{a}}$ denote quorums ratifying such statements, respectively. By the definition of quorum intersection, $U_a \cap U_{\bar{a}} \neq \emptyset$. Let $v \in U_a \cap U_{\bar{a}}$. This implies that v voted for both a and $\bar{a}$. However, the definition of voting(Definition 2.3.1) explicitly prohibits that. In other words, v must be ill-behaved, which is a contradiction to our assumption. □

Theorem 2.3.9. *Let $\langle V, Q \rangle$ be an FBAS and $B \subset V$. Suppose that B contains all the ill-behaved nodes and $\langle V, Q \rangle^B$ enjoys quorum intersection. Let $v_1 \neq v_2 \in V \setminus B$. If v_1 ratifies a statement a, then v_2 cannot ratify any statement that contradicts a.*

Proof. Suppose that the theorem is false and let U_1, U_2 be quorums of v_1, v_2 that ratify $a, \bar{a}$, respectively, where a and $\bar{a}$ are contradictory. Since $v_1 \in U_1 \setminus B$, $U_1 \setminus B \neq \emptyset$. By Theorem 2.1.13, $U_1' = U_1 \setminus B$ is a quorum in $\langle V, Q \rangle^B$. Similarly, $U_2' = U_2 \setminus B$ is a quorum in $\langle V, Q \rangle^B$. Since $\langle V, Q \rangle^B$ enjoys quorum intersection, $U_1' \cap U_2' \neq \emptyset$. Choose $v \in U_1' \cap U_2'$ arbitrarily. Then $v \in U_1 \cap U_2$. In order for U_1, U_2 to ratify $a, \bar{a}$, respectively, v must vote for both a and $\bar{a}$. However, this is against the definition of voting. v must be an ill-behaved node, so $v \in B$, which is a contradiction because $v \in U_1' \cap U_2' \subset U_1' = U_1 \setminus B$ and B contains all the ill-behaved nodes. □

Theorem 2.3.10. *Let $\langle V, Q \rangle$ be an FBAS with quorum intersection. Then two intact nodes in V cannot ratify contradictory statements.*

Proof. Let $v \neq v'$ be two intact nodes in V. Let $B \subset V$ be the set of befouled nodes. Then $v \notin B$ and $v' \notin B$. Since $\langle V, Q \rangle$ is an FBAS with quorum intersection, B is a DSet by Theorem 2.2.7. By the definition of a DSet (Definition 2.2.1), $\langle V, Q \rangle^B$ enjoys quorum intersection. By Theorem 2.3.9, v, v' cannot ratify contradictory statements. □

Lemma 2.3.11. *Let $\langle V, Q \rangle$ be an FBAS enjoying quorum intersection and B be the set of befouled nodes. If a is accepted by an intact node in V, then a is ratified by some intact node in $\langle V, Q \rangle^B$.*

Proof. Suppose that a is accepted by one or more intact nodes in V. Since V is finite, there has to be an intact node v such that no intact nodes in V accepted a before v.

By the definition of accepting(Definition 2.3.3), v accepted a because either

- There was a quorum U of v such that every element of U either voted for a or broadcast that it has accepted a, or
- There existed a v-blocking set such that every element in it broadcast that it has accepted a.

We claim that it could not have been the second one. On the contrary, suppose that it was the second one. Since no intact nodes in V accepted a before v, such a v-blocking set must have only had befouled nodes. Therefore, such a v-blocking set must be a subset of B. Since $\langle V, Q \rangle$ enjoys quorum intersection, B is a DSet by Theorem 2.2.7. By Theorem 2.2.9, B is not v-blocking. By taking the contrapositive of Theorem 2.1.22, we conclude that no subset of B is v-blocking.

Therefore, it must have been the first case. In other words, there must have existed a quorum U of v such that, before v accepted a, every member of U either voted for a or

broadcast that it has accepted a. Since no intact node accepted a before v, every intact node in U must have voted for a before v accepted a. In other words, every node in $U \setminus B$ voted for a. By Theorem 2.1.13, $U \setminus B$ is a quorum in $\langle V, Q\rangle^B$. Thus $U \setminus B$ ratified a in $\langle V, Q\rangle^B$, and thus v ratified a in $\langle V, Q\rangle^B$. Finally, v is indeed an intact node in $\langle V, Q\rangle^B$ because $\langle V, Q\rangle^B$ contains no ill-behaved nodes.

In conclusion, v is an intact node in $\langle V, Q\rangle^B$ and v ratified a in $\langle V, Q\rangle^B$. □

Theorem 2.3.12. *Two intact nodes in an FBAS $\langle V, Q\rangle$ enjoying quorum intersection cannot accept contradictory statements.*

By Theorem 2.3.6, ratifying is a stronger condition than accepting. Therefore, Theorem 2.3.12 is a stronger version of Theorem 2.3.10.

Proof. Suppose otherwise. Let $a, \bar{a}$ be contradictory statements accepted by two intact nodes in $\langle V, Q\rangle$. Let B denote the set of befouled nodes. By Lemma 2.3.11, $a, \bar{a}$ are ratified by some intact nodes in $\langle V, Q\rangle^B$. By the definition of a DSet(Definition 2.2.1), $\langle V, Q\rangle$ enjoys quorum intersection despite B.

This means that $\langle V, Q\rangle^B$ enjoys quorum intersection and two contradictory statements are ratified by some intact nodes in $\langle V, Q\rangle^B$. However, this is a direct contradiction to Theorem 2.3.10. Hence, two contradictory statements cannot be accepted by two intact nodes in $\langle V, Q\rangle$. □

2.4. **Confirmation.**

Definition 2.4.1 (Irrefutable). A statement a is irrefutable in an FBAS if no intact node can ever vote against it.

I don't use this definition anywhere. That sounds wrong.

Definition 2.4.2 (Confirm). A quorum U_a in an FBAS confirms a statement a if and only if every element in U_a broadcasts that it has accepted a. A node confirms a if and only if it is in such a quorum.

Theorem 2.4.3. *Ratifying is stronger than confirming, and confirming is stronger than accepting.*

Proof. Let $\langle V, Q\rangle$ and $v \in V$ be given. Suppose that v ratifies a statement a. Then there exists a quorum U such that $v \in U$ and every member in U votes for a. For any $u \in U$,

- u has never accepted a statement contradicting a by the definition of voting (Definition 2.3.1), and
- U is a quorum such that $u \in U$ and every memeber of U voted for a.

Therefore, u accepts a. In other words, every member of U accepts a. U confirms a and thus v confirms a. Thus ratifying is stronger than confirming.

The definition of confirming(Definition 2.4.2) shows that a node must first accept a statement before confirming. Therefore, confirming is stronger than accepting. □

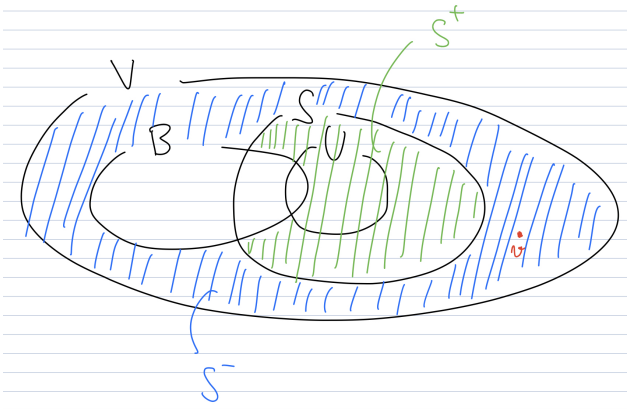

FIGURE 3. Lemma 2.4.4

Lemma 2.4.4. *Let $\langle V, Q\rangle$ be an FBAS with quorum intersection. Let B denote the set of befouled nodes. Let U be a quorum containing an intact node, and let S be a set containing U. Let S^+ be the set of intact nodes in S, and let S^- be the set of intact nodes not in S. Either $S^- = \emptyset$, or $\exists v \in S^-$ such that S^+ is v-blocking. (See Figure 3.)*

Proof. Note that $S^+ = S \setminus B$ and $S^- = (V \setminus S) \setminus B = (V \setminus B) \setminus S^+$. If $\exists v \in S^-$ such that S^+ is v-blocking, then we are done.

Suppose that $\forall v \in S^-$, S^+ is not v-blocking in $\langle V, Q\rangle$. We want to show that $S^- = \emptyset$.

- S^+ and S^- form a partition of $V \setminus B$,

- S^+ is not v-blocking in $\langle V, Q\rangle$ for any arbitrary $v \in S^-$,

By Theorem 2.1.23, S^+ is not v-blocking in $\langle V, Q\rangle^B$ for any $v \in S^-$. Since $S^- = (V \setminus B) \setminus S^+$, S^+ is not v-blocking in $\langle V, Q\rangle^B$ for any $v \in (V \setminus B) \setminus S^+$. By applying Theorem 2.1.24 to the FBAS $\langle V, Q\rangle^B$ and subset S^+, $\langle V, Q\rangle^B$ enjoys quorum availability despite S^+.

By the definition of quorum availability(Definition 2.1.17), $(V \setminus B) \setminus S^+$ is a quorum in $\langle V, Q\rangle^B$, or $V \setminus B = S^+$. Suppose $(V \setminus B) \setminus S^+$ is a quorum in $\langle V, Q\rangle^B$. This leads to two contradictory claims:

- Claim 1: $\langle V, Q\rangle$ enjoys quorum intersection despite B.
  - Since $\langle V, Q\rangle$ enjoys quorum intersection, B is a DSet by Theorem 2.2.7. By the definition of a DSet(Definition 2.2.1), $\langle V, Q\rangle^B$ enjoys quorum intersection.
- Claim 2: $\langle V, Q\rangle$ does not enjoy quorum intersection despite B.
  - $U \setminus B$ is nonempty since U contains an intact node. Thus $U \setminus B$ is a quorum in $\langle V, Q\rangle^B$ by Theorem 2.1.13. We also assumed that $(V \setminus B) \setminus S^+$ is a quorum in $\langle V, Q\rangle^B$.

$$\begin{aligned}(U \setminus B) \cap ((V \setminus B) \setminus S^+) &= (U \setminus B) \cap S^- \\ &\subset (S \setminus B) \cap S^- \\ &\subset S^+ \cap S^- \\ &= \emptyset.\end{aligned}$$

    Therefore, $\langle V, Q\rangle^B$ does not enjoy quorum intersection.

Therefore, $(V \setminus B) \setminus S^+$ must not be a quorum in $\langle V, Q\rangle^B$, so $V \setminus B$ must be S^+. In other words, S^+ contains all the intact nodes, so $S^- = \emptyset$, which is exactly what we wanted to show. □

Theorem 2.4.5. *If an intact node in an FBAS $\langle V, Q\rangle$ with quorum intersection confirms a statement a, then every intact node will accept and confirm a once sufficient messages are delivered.*

Proof. Let B denote the set of befouled nodes. When an intact node confirms a, some quorum containing such an intact node confirms a. In other words, there exists a quorum $U \not\subset B$ such that every node in U broadcast that it accepted a. Some nodes may decide to accept a upon hearing that nodes in U broadcast that it accepted a. This may, in turn, make more nodes accept a. Thus we may experience a gradual increase in the number of nodes that accept a over time. Since V only contains finitely many nodes, there will be a point at which the number of nodes that accept a stops increasing. Let S be the set of nodes that accept a at that point. We claim that S contains all intact nodes.

- U is a quorum containing an intact node.
- $U \subset S \subset V$ because every node in U accepted a in the beginning.
- Let $S^+ = S \setminus B$ be the set of intact nodes in S, and let $S^- = (V \setminus S) \setminus B$ be the set of intact nodes not in S.

By Lemma 2.4.4, S^- is empty, or S^+ is v-blocking for some $v \in S^-$. Suppose S^- is nonempty. Then v accepts a because S^+ is v-blocking and every element of S^+ broadcast that it has accepted a. This is a contradiction because we assumed that the number of nodes that accept a stopped increasing. Therefore, S^- must be empty. If S^- is empty, then that

implies that every intact node accepted and confirmed a assuming sufficient messages are delivered because

- Since S^- is empty, S^+ contains all intact nodes in V. $S^+ \subset S$, so S contains all intact nodes. Since every node in S accepted a, every intact node accepted a.
- Since $\langle V, Q \rangle$ enjoys quorum intersection, B is a DSet by Theorem 2.2.7. By the definition of a DSet, $V \setminus B$ is a quorum. In other words, the set of all intact nodes is a quorum.
- Since $V \setminus B$ is a quorum in which every node accepted a, every intact node confirmed a.

□

3. Stellar Consensus Protocol

3.1. Nomination Protocol.

Nomination is done through voting, accepting, and confirming a special type of statement in the form of *nominate* x.

Definition 3.1.1 (Nomination Statement). *nominate* x is a special statement which is short for

- *Value* x *passes application validity checks, and*
- *I have never confirmed a nomination statement.*

Definition 3.1.2 (Nominate). A node is said to nominate a value x if and only if it votes for the statement *nominate* x before any other node.

Definition 3.1.3 (Renominate). A node is said to renominate a value x if and only if it votes for the statement *nominate* x after some other node votes for it.

Definition 3.1.4 (Candidate). A node v considers a value x to be a candidate if and only if v has confirmed the statement *nominate* x. Alternatively, we say that v has a candidate value x.

Definition 3.1.5 (Produce). A node is said to produce a candidate value x when it confirms *nominate* x after nominating it.

Remark 3.1.6. This approach is slightly different from the white paper's. The white paper seems to define some special "rule" on voting for a nomination statement. However, that seems to change the definition of voting (Definition 2.3.1), and I was unable to convince myself that we could simply apply FBA arguments after changing the definition of voting.

Instead, I decided to define a nomination statement to be a statement containing some special instructions. This approach should achieve the same thing because a well-behaved node is allowed to vote for a statement only if it is consistent with history.

Theorem 3.1.7. *The set of intact nodes will eventually produce at least one candidate value.*

The white paper mentions that the set of intact nodes *can* eventually produce at least one candidate value. However, we should be able to change it to *will* as long as nodes without candidate values continue renominating other candidate's values.

Proof. Suppose that, at some point in time, an intact node confirms a nominate statement. Then we are done. Suppose otherwise.

Let v be an intact node and x be a value that passes application validity checks. We assume that v is capable of finding such a value. Then v will vote for *nominate* x. By Definition 1.0.1, the message will eventually get delivered to all intact nodes. Since the value passes application validity checks and no intact node has confirmed a nominate statement, every intact node will eventually vote for *nominate* x. By the definition of a DSet(Definition 2.2.1), the set of all intact nodes is a quorum, so all the intact nodes will accept and confirm *nominate* x after sufficient messages have been delivered. This is a contradiction because we assumed that no intact node confirms a nominate statement.

Therefore, the set of intact nodes can eventually produce at least one candidate value. □

Theorem 3.1.8. *Let* $\langle V, Q \rangle$ *be an FBAS that enjoys quorum intersection. If any intact node considers* x *to be a candidate value, then eventually every intact node will consider* x *to be a candidate value.*

Proof. Suppose that an intact node considers x to be a candidate value. In other words, an intact node confirmed the statement *nominate* x. By Theorem 2.4.5, all intact nodes will eventually confirm it, and thus all intact nodes will eventually consider x to be a candidate value. □

Remark 3.1.9. Note that Theorem 3.1.8 does not hold if the FBAS does not enjoy quorum intersection. In Figure 4, each node is intact because

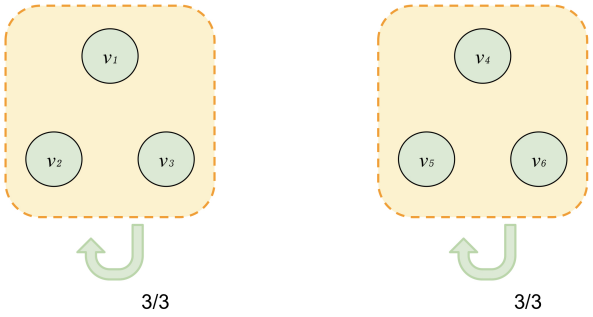

FIGURE 4. Nomination Procol And Quorum Intersection

- $B = v_4, v_5, v_6$ is a DSet such that $v_1, v_2, v_3 \notin B$. Thus v_1, v_2, v_3 are intact.
- $B = v_1, v_2, v_3$ is a DSet such that $v_4, v_5, v_6 \notin B$. Thus v_4, v_5, v_6 are intact.

Suppose the nodes vote, accept, and confirm as following:

- v_1, v_2, v_3 vote, accept, and confirm *nominate* x before hearing about *nominate* y.
- v_4, v_5, v_6 vote, accept, and confirm *nominate* y before hearing aobut *nominate* x.

In such a case, v_1, v_2, v_3 will never have the same set of candidates as v_4, v_5, v_6 because they can no longer vote for nomination statements. The same argument applies to any FBAS without quorum intersection. If an FBAS does not enjoy quorum intersection, there exists a pair of disjoint quorums Q_1, Q_2. If every node in Q_1 votes, accepts and confirms *nominate* x and every node in Q_2 votes, accepts and confirms *nominate* y before hearing the other nomination statement, they can never have the same set of candidates.

Lemma 3.1.10. *In an FBAS that enjoys quorum intersection, an intact node cannot accept a statement that no intact node voted for.*

We will use this lemma to prove Theorem 3.1.11.

Proof. On the contrary, suppose that it is possible. Let x be a statement that no intact node voted for and v be the first intact node that accepts it.

In order for an intact node v to accept x, we need

- The existence of a quorum U containing v such that every member of U either voted for or accepted x, or
- A v-blocking set consisting of nodes that accepted x.

However, neither of them is possible because

- We assumed that no intact node ever voted for x. Therefore, v never voted for x. Thus the first condition can never be satisfied.

- Let B be such a v-blocking set. The set of intact nodes is a quorum by Corollary 2.2.8. Therefore, B must contain at least one intact node w. However, this implies that w accepted x before v. This is a contradiction because we specifically picked v such that v is the first node to accept the statement. Therefore, the second condition can never be satisfied.

Therefore, it is impossible for v to accept x. □

Theorem 3.1.11. *If an FBAS enjoys quorum intersection, the set of possible candidate values eventually stops growing.*

Proof. By applying Theorem 3.1.7 and Theorem 3.1.8, there exists a point T in time where all intact nodes have at least one candidate value.

Let S denote the set of all nomination statements that intact nodes voted for before T. S is finite because T is a specific point in time.

We claim that S is the set of possible candidate values, and a nomination statement outside S can never be confirmed.

We will first show that a nomination statement outside S cannot be accepted by an intact node. No intact node will ever vote for a nomination statement after T. Therefore, S is the set of all nomination statements that intact nodes will ever vote for. By Lemma 3.1.10, no intact node will ever accept a statement not in S.

The set of intact nodes is a quorum by Corollary 2.2.8. Since the FBAS enjoys quorum intersection, any quorum intersects the set of intact nodes. In other words, any quorum contains an intact node. Therefore, it is impossible for any quorum to confirm a statement not in S. Hence, S is the set of possible candidate values. □

Theorem 3.1.12. *If an FBAS enjoys quorum intersection, all intact nodes will have the same composite value after sufficient messages have been delivered.*

proofread this!

Proof. By Theorem 3.1.11, there exists a finite set S containing the set of possible candidate values. For each value $x \in S$, there exist two possibilities:

- Within a finite amount of time, an intact node considers x a candidate value. By Theorem 3.1.8, all intact nodes will consider x a candidate value after sufficient messages have been delivered. In other words, within a finite amount of time, all intact nodes will consider x a candidate value.
- No intact node ever considers x to be a candidate value at any point.

Since

- There are only finitely many values in S, and
- Each value either becomes a candidate value at every intact node within a finite amount of time or never becomes a candidate value,

every intact node will have the same composite value within a finite amount of time. □

Definition 3.1.13 (Weight, Neighbors, and Priority). Let H be a cryptographic hash function whose range can be interpreted as a set of integers $\{0, \cdots, h_{\max} - 1\}$. Let $G_i(m) = H(i, x_{i-1}, m)$ be a slot-specific hash function for slot i, where x_{i-1} is the value chosen for the

slot preceding i. Let $N \neq P$ are arbitrary constants. Given a slot i, a round number n and node v, we define

$$\begin{aligned}\text{weight}(v, v') &= \frac{|\{q \in Q(v) \mid v' \in q\}|}{|Q(v)|} \\ \text{neighbors}(v, n) &= \{v' \mid G_i(N, n, v') < h_{\max} \cdot \text{weight}(v, v')\} \\ \text{priority}(n, v') &= G_i(P, n, v')\end{aligned}$$

for each neighbor v'.

Remark 3.1.14. In Stellar Core, N and P are always 1 and 2.

Definition 3.1.15 (Leader). Let a slot i, round number n, and node v be given. Then the node $v_0 \in \text{neighbors}(v, n)$ that maximizes $\text{priority}(n, v_0)$ among the nodes it can communicate with is considered to be the leader by v. Node v votes to nominate the same values as v_0. Only if $v = v_0$, v can introduce a new value to nominate.

Example 3.1.16 (Weight, Neighbors, and Priority). We will calculate the weight, neighbors, and priority for v_5 in Figure 2 as an example. Since each quorum slice consists of v_5 along with two nodes from $\{v_1, v_2, v_3, v_4\}$ as in

$$Q(v_5) = \{\{v_1, v_2, v_5\}, \{v_1, v_3, v_5\}, \{v_1, v_4, v_5\}, \cdots, \{v_3, v_4, v_5\}\},$$

$Q(v_5)$ has $\binom{4}{2} = 6$ slices.

We will first calculate the weight of v_1.

$$\begin{aligned}\text{weight}(v_5, v_1) &= \frac{|\{q \in Q(v_5) \mid v_1 \in q\}|}{|Q(v_5)|} \\ &= \frac{|\{v_1, v_2, v_5\}, \{v_1, v_3, v_5\}, \{v_1, v_4, v_5\}|}{6} \\ &= \frac{3}{6} = \frac{1}{2}.\end{aligned}$$

By symmetry, $\text{weight}(v_5, v_j) = \frac{1}{2}$ for each $j = 1, 2, 3, 4$. $\text{weight}(v_5, v_5) = 1$ because every quorum slice in $Q(v_5)$ contains v_5. Finally, $\text{weight}(v_5, v_j) = 0$ for all $j = 6, 7, 8, 9, 10$ because no quorum slice in $Q(v_5)$ contains any of $v_6, v_7, \cdots, v_{10}$. Therefore, we obtain the following table:

j	$\text{weight}(v_5, v_j)$
1, 2, 3, 4	1/2
5	1
6, 7, 8, 9, 10	0

For this example, we will suppose that $h_{max} = 100$. Let N, P, i, n be fixed. Then we will calculate the neighbors. First, we will start with v_1, v_2, v_3, v_4.

Suppose

$$\begin{aligned}G_i(N, n, v_1) &= 41 \\ G_i(N, n, v_2) &= 72 \\ G_i(N, n, v_3) &= 19 \\ G_i(N, n, v_4) &= 84.\end{aligned}$$

The condition for a node v' to be in neighbors(v_5, n) is $G_i(N, n, v') < 100 \cdot \text{weight}(v_5, v')$. Therefore, $v_1, v_3 \in \text{neighbors}(v_5, n)$. (e.g., $G_i(N, n, v_1) = 41 < 50 = 100 \cdot 1/2 = 100 \cdot \text{weight}(v_5, v_1)$.)

Moreover, $v_5 \in \text{neighbors}(v_5, n)$ since $\text{weight}(v_5, v_5) = 1$. Finally, $v_j \in \text{neighbors}(v_5, n)$ for each $j = 6, 7, \cdots, 10$ because $\text{weight}(v_5, v_j) = 0$.

Therefore, we have

$$\text{neighbors}(v_5, n) = \{v_1, v_3, v_5\}.$$

This is a reasonable choice of neighbors because

- v_5 trusts $v_1, \cdots, v_4$, so it is a good thing that we have v_1, v_3 in neighbors(v_5, n).
- v_5 trusts v_5.
- Since v_5 does not trust $v_6, \cdots, v_{10}$, v_5 has no quorum slice containing any of them. Thus it is a good thing that neighbors(v_5, n) does not contain any of them.

Finally, suppose

$$\begin{aligned}\text{priority}(n, v_1) &= G_i(P, n, v_1) = 17\\ \text{priority}(n, v_3) &= G_i(P, n, v_3) = 86\\ \text{priority}(n, v_5) &= G_i(P, n, v_5) = 25.\end{aligned}$$

Then v_5 will simply nominate the same value as v_3.

Remark 3.1.17.

- weight is not symmetric in general. In other words, $\text{weight}(v_i, v_j) \neq \text{weight}(v_j, v_i)$ in general.
- neighbors(v_i) is a set of nodes calculated locally at each v_i. In general, neighbors$(v_i) \neq$ neighbors(v_j) for any $i \neq j$.
- priority(n, v) is *global* in a sense that the values of priority(n, v) calculated at node w and w' must be identical for it only depends on n, the hash function G_i and the constant P.

Example 3.1.18 (Leader Election). We will again use the FBAS in Figure 2 as an example. Suppose that each node's neighbors is as follows:

j	neighbors(v_j)
1	$\{v_1, v_3\}$
2	$\{v_2, v_4\}$
3	$\{v_2, v_3, v_4\}$
4	$\{v_1, v_2, v_4\}$
5	$\{v_2, v_5\}$
6	$\{v_1, v_3, v_6\}$
7	$\{v_1, v_2, v_3, v_7\}$
8	$\{v_3, v_8\}$
9	$\{v_6, v_7, v_8, v_9\}$
10	$\{v_{10}\}$

Suppose that priority$(n, v_j) = G_i(P, n, v_j)$ for each j is as follows:

j	1	2	3	4	5	6	7	8	9	10
priority(n, v_j)	26	3	60	89	18	56	35	19	61	27

Note that the priority of each node is *global*. For instance, priority(n, v_1) calculated at v_2 and v_6 are the same. Thus it makes sense to have a table like this.

Then each node's leader (the node whose nominations it will renominate) will be as follows:

j	1	2	3	4	5	6	7	8	9	10
v_j's leader	v_3	v_4	v_4	v_4	v_5	v_3	v_3	v_3	v_9	v_{10}

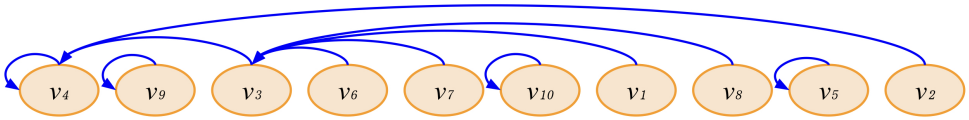

FIGURE 5. Leader Relation Graph

As you can see, this is a directed graph such that the only cycles are self-loops. In this particular case, v_4, v_9, v_{10}, v_5 will produce new values, and other nodes will simply renominate those values.

Example 3.1.19 (Leader Election With A Failed Node). Suppose that, in the previous example, node v_3 stops responding due to a power outage. Since the leader is the node in the neighbors that maximizes the priority among the nodes that it can communicate with, no node will pick v_3 as their leader. Under this circumstance, each node will pick the following nodes as their leader:

j	1	2	3	4	5	6	7	8	9	10
v_j's leader	v_3	v_4	v_4	v_4	v_5	v_3	v_3	v_3	v_9	v_{10}

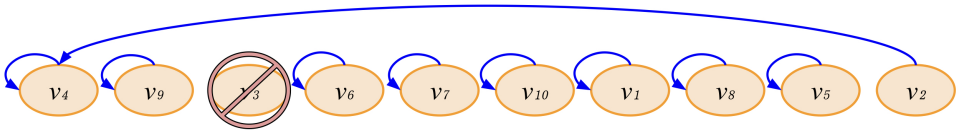

FIGURE 6. Leader Relation Graph

In this case, all the nodes except for v_3, v_2 will produce new values while v_2 will simply renominate the what v_4 votes for.

This is actually really bad because almost all *nominate* statements will only have one vote. v_2 will vote for all the *nominate* statements that v_4 votes for as long as v_2 has not confirmed anything. Actually, v_2 is the *only* one who will vote for any of the *nominate* statements from v_4. Since $\{v_2, v_4\}$ is not a quorum, none of the *nominate* statements from v_4 will be accepted by any node. A similar argument can be applied to every other node to show that no *nominate* statement will be accepted.

3.2. **Ballot Protocol.**

Definition 3.2.1 (Ballot). A ballot b is a pair of the form $b = \langle n, x \rangle$ where $x \neq \perp$ is a value and $n \geq 1$ is a counter.

Definition 3.2.2 (Order). For any ballots b_1, b_2, $b_1 \leq b_2$ if and only if $(b_1.n, b_1.x) \leq (b_2.n, b_2.x)$ in dictionary order.

Definition 3.2.3 (Null ballot). $\mathbf{0} = \langle 0, \perp \rangle$ is a special null invalid ballot that is less than any other ballots.

Definition 3.2.4 (Compatability). For a pair of ballots b_1, b_2, we define $\sim, \nsim, \lesssim, \lnsim$ as following:

$$
\begin{aligned}
b_1 \sim b_2 &\iff b_1.x = b_2.x \\
b_1 \nsim b_2 &\iff b_1.x \neq b_2.x \\
b_1 \lesssim b_2 &\iff b_1 \leq b_2 \wedge b_1 \sim b_2 \\
b_1 \lnsim b_2 &\iff b_1 \leq b_2 \wedge b_1 \nsim b_2.
\end{aligned}
$$

Example 3.2.5. While this may seem relatively straightforward, I found this unexpectedly complicated. Here are some basic properties:

- $b_1 \lesssim b_2 \lesssim b_3 \implies b_1 \lesssim b_2$.
- $b_1 \lesssim b_2 \lnsim b_3 \implies b_1 \lnsim b_2$.
- $b_1 \lnsim b_2 \lesssim b_3 \implies b_1 \lnsim b_2$.
- $b_1 \lnsim b_2 \implies b_1 < b_2$.
- $b_1 \sim b_2 \implies b_1 \lesssim b_2 \vee b_2 \lesssim b_1$.
- $b_1 \nsim b_2 \implies b_1 \lnsim b_2 \vee b_2 \lnsim b_1$.

One might mistakenly assume that the following may hold:

- $b_1 \lesssim b_2 \lesssim b_3 \implies b_1 \lesssim b_3$.
  - This is not true in general. For instance, $b_1 = (1, x), b_2 = (2, y), b_3 = (3, x)$ would be a counterexample.
- $b_1 < b_2 \implies b_1 \lnsim b_2$.
  - This is not true in general. For instance, $b_1 = (1, x), b_2 = (2, x)$ would be a counterexample.

Definition 3.2.6 (Commit). For a given ballot b, *commit* b is a shorthand for "I have confirmed $\overline{\textit{commit } b'}$ for all $b' \lnsim b$."

$\overline{\textit{commit } b'}$ denotes a statement that contradicts *commit* b'.

Remark 3.2.7 (Abort). For a given ballot b, we will often write *abort* b instead of $\overline{\textit{commit } b}$ for readability.

Example 3.2.8. Let a, b, c, d, e be statements with $a < b < c < d < e$. The following table shows what statements a node needs to confirm to abort before voting for *commit* $(3, c)$:

(n, x)	$x = a$	$x = b$	$x = c$	$x = d$	$x = e$
$n = 1$	*abort*	*abort*		*abort*	*abort*
$n = 2$	*abort*	*abort*		*abort*	*abort*
$n = 3$	*abort*	*abort*			
$n = 4$					

There are a few things that may be worth mentioning:

- $(2, c)$ is *not* $\lesssim (3, c)$ because $(2, c) \sim (3, c)$.
- $(3, a) \lesssim (3, c)$ because $(3, a) \leq (3, c)$ and $(3, a) \nsim (3, c)$.
- $(3, d)$ is *not* $\lesssim (3, c)$ even though $(3, d) \nsim (3, c)$. This is because $(3, d)$ is *not* $\leq (3, c)$.

Definition 3.2.9 (Prepare). Let b be a ballot. A node is said to vote, accept, or confirm that b *is prepared* if and only if it votes, accepts, or confirms *abort* b' for all $b' \lesssim b$, respectively.

Remark 3.2.10. By the definition of committing and preparing, a node can vote for *commit* b if and only if it has confirmed that b is prepared.

Remark 3.2.11. The approach presented here is slightly different from the white paper's. In the white paper, the fact that *commit* b is valid to vote for only if b is confirmed prepared is given as a (seemingly arbitrary) "rule." However, I was not very convinced by this approach because it would mean that we need to alter the definition of voting. More specifically, the definition of voting states that one votes for a statement *if and only if* the four conditions have been met. However, this new "rule" may prevent a node from voting even when all of the four conditions.

In the approach presented above, the "rule" regarding voting for a *commit* b is embedded in the commit statement itself. Therefore, we can safely apply all of our FBAS arguments.

Definition 3.2.12 (Externalize). A node *externalizes* a value x if and only if it confirms *commit* $\langle n, x \rangle$ for some $n \geq 1$.

Theorem 3.2.13. *Let $b = (b.n, b.x)$ be given. In an FBAS with a quorum intersection, if an intact node confirms that b is prepared, then every intact node will eventually confirm that b is prepared.*

Proof. By Definition 3.2.9, confirming that b is prepared is equivalent to confirming *abort* b' for all $b' \lesssim b$. For each $b' \lesssim b$, Theorem 2.4.5 states that all intact nodes will eventually confirm *abort* b'. Therefore, all intact nodes will eventually confirm that b is prepared. □

Theorem 3.2.14. *Let $\langle V, Q \rangle$ be an FBAS with quorum intersection. Let $v \in V$ be an intact node. If v accepts commit b, all intact nodes will eventually confirm commit b' for some $b' \sim b$.*

Proof. Let w be an intact node and suppose w accepts *commit* b' for some ballot b'. If $b \sim b'$, we are done. Suppose otherwise.

Since v, w accept b, b', they must have confirmed that b, b' are prepared, respectively. Without loss of generality, $b \lesssim b'$. By Definition 3.2.9, w must have confirmed *abort* b. *abort* b and *commit* b are defined to be contradictory, and Theorem 2.3.12 states that no intact node can accept contradictory statements. Therefore, this is a contradiction, so $b \sim b'$.

Since a node must first accept a statement before confirming it, the argument above shows that if an intact node confirms *commit* b', then $b' \sim b$.

Now, we need to show that all intact nodes will eventually confirm *commit* b' for some $b' \sim b$.

This "liveness" property requires the concrete ballot protocol.

□

3.2.1. *Concrete Ballot Protocol.*

Definition 3.2.15 (Ballot State). The tuple $(\varphi, b, p', p, c, h, z, M)$ where

- $\varphi \in$ {PREPARE, CONFIRM, EXTERNALIZE},
- b, p', p, c, h are ballots,
- z is a value,
- M is a set of messages

is called a ballot state, and each node must maintain one during the ballot protocol.

Definition 3.2.16 (Meaning of Variables).

FInish the rest.

- p', p are the two highest ballots accepted as prepared such that $p' \lesssim p$, where $p' = \mathbf{0}$ or $p = p' = \mathbf{0}$ if there are no such ballots.

Definition 3.2.17 (Ballot Message). Any message that a well-behaved node sends is one of the following messages:

- (PREPARE, $v, i, b, p, p', c.n, h.n, D$),
- (CONFIRM, $v, i, b, p.n, p'.n, c.n, h.n, D$),
- (EXTERNALIZE, $v, i, c.x, c.n, h.n, D$),

and the phase in the message must match the phase in the state.

Definition 3.2.18 (PREPARE). (PREPARE, $v, i, b, p, p', c.n, h.n, D$) encodes

- $\{abort\ b' \vee accept(abort\ b') \mid b' \lesssim b\}$
- $\{accept(abort\ b') \mid b' \lesssim p\}$
- $\{accept(abort\ b') \mid b' \lesssim p'\}$
- $\{commit\ b' \mid c.n \neq 0 \wedge c \lesssim b' \lesssim h\}$

Add CONFIRM, EXTERNALIZE. But when I think about it, this part may not be super necessary? Can we prove all the invariants without defining what messages look like?

Definition 3.2.19. The state such that

- $\varphi =$ PREPARE
- $b = p = p' = c = h = \mathbf{0}$
- $z = \perp$
- $M = \emptyset$

is called the initial ballot state, and every well-behaved node must have this state when the ballot protocol starts.

There are exactly two ways in which a well-behaved node is allowed to change the state:

- When a node receives a new nomination message.
- When a node receives a new ballot message.

We will call them Transition Type 1 and Type 2, respectively.

Definition 3.2.20 (Transition Type 1). When $h = \mathbf{0}$, a node may update z in response to a NOMINATE message. If z was $\perp$, then $b \leftarrow \langle 1, z \rangle$.

Definition 3.2.21 (Transition Type 2). Upon receiving a new ballot message m, node v adds m to M and updates its state as following:

(1) If $\varphi = \text{PREPARE}$ and m lets v accept new ballots as prepared, update p and p'. Afterwards, if either $p \gtrapprox h$ or $p' \gtrapprox h$, then set $c \leftarrow \mathbf{0}$.
(2) If $\varphi = \text{PREPARE}$ and m lets v confirm new higher ballots prepared, then raise h to the highest such ballot and set $z \leftarrow h.x$.
(3) If $\varphi = \text{PREPARE}, c = \mathbf{0}, b \leq h$, and neither $p \gtrapprox h$ nor $p' \gtrapprox h$, then set c to the lowest ballot satisfiying $b \leq c \lesssim h$.
(4) If $\varphi = \text{PREPARE}$ and v accepts *commit* for one or more ballots, set c to the lowest such ballot, then set h to the highest ballot such that v accepts all $\{\textit{commit } b' \mid c \lesssim b' \lesssim h\}$, and set $\varphi \leftarrow \text{CONFIRM}$. Also, set $z \leftarrow h.x$ after updating h, and unless $h \lesssim b$, set $b \leftarrow h$.
(5) If $\varphi = \text{CONFIRM}$ and the received message lets v accept new ballots prepared, raise p to the highest accepted prepared ballot such that $p \sim c$.
(6) If $\varphi = \text{CONFIRM}$ and v accepts more commit messages or raises b, then let h' be the highest ballot such that v accepts all $\{\textit{commit } b' \mid b \lesssim b' \lesssim h'\}$ (if any). If there exists such an h' and $h' > h$, then set $h \leftarrow h'$, and, if necessary, raise c to the lowest ballot such that v accepts all $\{\textit{commit } b' \mid c \lesssim b' \lesssim h\}$.
(7) If $\varphi = \text{CONFIRM}$ and v confirms *commit* c' for any c', set c and h to the lowest and highest such ballots, set $\varphi \leftarrow \text{EXTERNALIZE}$, externalize $c.x$ and terminate.
(8) If $\varphi \in \{\text{PREPARE}, \text{CONFIRM}\}$ and $b < h$, then set $b \leftarrow h$.
(9) If $\varphi \in \{\text{PREPARE}, \text{CONFIRM}\}$ and $\exists S \subseteq M_v$ such that the set of senders $\{v_{m'} \mid m' \in S\}$ is v-blocking and $\forall m' \in S, b_{m'}.n > b_v.n$, then set $b \leftarrow \langle n, z\rangle$, where n is the lowest coutner for which no such S exists. Repeat the previous steps after updating b.

We will now prove invariants of states in the concrete ballot protocol.

Lemma 3.2.22. *If v has confirmed any ballot as prepared, $h \neq \mathbf{0}$.*

Proof. It suffices to show that it is impossible that

(a) v has confirmed a ballot as prepared, and
(b) $h = \mathbf{0}$.

This invariant holds in the initial state because v has not confirmed anything initially.

Suppose that the invariant holds in the current state. Then there are $3 = 2^2 - 1$ possible ways in which the invariant can be violated.

(i) (a) currently holds and (b) does not. After some actions caused by a new message, (b) becomes true.
(ii) (a) does not hold and (b) currently holds. After some actions caused by a new message, (a) becomes true.
(iii) Neither (a) nor (b) holds. After some actions caused by a new message, both become true.

Since none of the actions sets $h \leftarrow \mathbf{0}$, (i) and (iii) are impossible. Moreover, if $h = \mathbf{0}$ and a new message lets v confirm a ballot as prepared, Step (2) forces v to update the value of h to a nonzero value. It is possible that v performs some other actions after Step (2), but no action sets $h \leftarrow \mathbf{0}$, so (b) will never hold. In other words, (ii) is impossible. □

Lemma 3.2.23. *If $\varphi = \text{PREPARE}$ and $h \neq \mathbf{0}$, then h is the highest ballot confirmed as prepared.*

Proof. It suffices to show that, at any given moment,

(a) $\varphi \neq$ PREPARE, or
(b) $h = \mathbf{0}$, or
(c) h is the highest ballot confirmed as prepared.

This invariant holds in the initial state since $h = \mathbf{0}$. Suppose that the invariant is true in the current state. There are $7 = 2^3 - 1$ possible ways in which the invariant can be violated.

(i) (a), (b), (c) all hold in the current state, but all of them become false after one message.
(ii) Only (a) and (b) hold in the current state, but they become false after one message.
(iii) Only (a) and (c) hold in the current state, but they become false after one message.
(iv) Only (b) and (c) hold in the current state, but they become false after one message.
(v) Only (a) holds in the current state, but it becomes false after one message.
(vi) Only (b) holds in the current state, but it becomes false after one message.
(vii) Only (c) holds in the current state, but it becomes false after one message.

First, it is easy to see that, if $\varphi \neq$ PREPARE, then no message can set $\varphi \leftarrow$ PREPARE. Therefore, (i), (ii), (iii) and (v) are impossible.

By taking the contrapositive of Lemma 3.2.22, we know that if $h = \mathbf{0}$, then v has not confirmed any ballot as prepared. Thus (iv) is impossible.

We will now examine option (vi). Suppose $\varphi =$ PREPARE$, h = \mathbf{0}$. The only way for v to set h to a ballot that is not $\mathbf{0}$ while having $\varphi =$ PREPARE is Step (2) in Transition Type 2(3.2.21). However, Step (2) sets h to the highest ballot confirmed as prepared, so (c) becomes true through the step. Therefore, (vi) is impossible.

Similarly, consider option (vii). Then we have $\varphi =$ PREPARE$, h \neq \mathbf{0}$, and h is the highest ballot confirmed as prepared. For (c) to become false, we need to either update the value of h, or confirm another ballot as prepared while keeping $\varphi =$ PREPARE. This means we must take Step (2) in Transition Type 2(3.2.21), and that means we will update h to the highest ballot confirmed as prepared. Thus (c) continues to hold and (vii) is impossible.

Hence, we confirm that the invariant always holds. □

Theorem 3.2.24. *Suppose $\varphi =$ PREPARE. h is the highest ballot confirmed as prepared, or $\mathbf{0}$ if none.*

Proof. Lemma 3.2.22 and 3.2.23. □

Lemma 3.2.25. *If $\varphi =$ PREPARE, then v has confirmed c as prepared.*

Proof. It suffices to show that, at any given moment,

(a) $\varphi \neq$ PREPARE, or
(b) v has confirmed c as prepared.

Note that every node confirms $\mathbf{0}$ as prepared vacuously because $\{\mathit{abort}\ b' \mid b' \lesssim \mathbf{0}\} = \emptyset$. Thus the invariant holds initially since (b) holds in the initial state. Suppose that the invariant holds in some state. Since no action sets $\varphi \leftarrow$ PREPARE, there is no state in which (a) becomes true after an action. Thus we only need to consider the possibility where only (b) holds in the current state, and (b) becomes false after an action. In other words, $\varphi =$ PREPARE and v has confirmed c as prepared in the current state, but after an action, v has not confirmed c as prepared. Since the FBA does not have a way to "un-confirm" a

statement, this means c changed to something v has not confirmed as prepared. There are exactly two actions that modify c while keeping $\varphi = \text{PREPARE}$:

- Step (1) may set $c \leftarrow \mathbf{0}$. As discussed above, every node confirms $\mathbf{0}$ as prepared vacuously, so (b) cannot become false after Step (1).
- Step (3) may update c such that $c \lesssim h$. Lemma 3.2.23 shows that v must have confirmed h as prepared. In other words, v must have confirmed abort b' for all $b' \lesssim h$. Since $b' \lesssim c \wedge c \lesssim h \implies b' \lesssim h$, v must have confirmed the new c as prepared. In other words, (b) does not become false after Step (3).

Therefore, if $\varphi = \text{PREPARE}$, then v has confirmed c as prepared. □

Lemma 3.2.26. *$c.n \leq h.n \leq b.n$.*

Proof. This holds in the initial state since $c.n = h.n = b.n = 0$. The only way to violate this invariant is to start with a state where only (b) holds, and after an action (b) no longer holds. There are several actions that modify at least one of $c.n, h.n, b.n$:

- Transition Type 1: Suppose $c.n \leq h.n \leq b.n$. If Transition Type 1 is applied, $h = \mathbf{0}$. Thus $c.n = h.n = 0$. Transtion Type 1 may update $b \leftarrow \langle 1, z \rangle$. Since $0 \leq 1$, (b) continues to hold after Transition Type 1.
- Transition Type 2: Step (9) requires the node to repeat the previous steps if some action takes place. Thus Step (8) is the last step that any node tries to apply upon receiving any new message. Since Step (8) ensures that $h \leq b$, we know that $h.n \leq b.n$ always. We will now check if $c.n \leq h.n$ continues to hold.
  - Step (2) modifies h. Suppose that v had confirmed at least one ballot prepared. By Theorem 3.2.24, h is the highest ballot confirmed prepared. Step (2) raises h to a higher ballot, so $c.n \leq h.n$ continues to hold.
    Now we will consider the case where v had not confirmed any ballot prepared. By Theorem 3.2.24, $h = \mathbf{0}$. Step (2) raises h to a higher ballot, so $c.n \leq h.n$ continues to hold.
  - Step (3), (4) and (6) explicitly set c such that $c \lesssim h$, so $c.n \leq h.n$ continues to hold.
  - Step (7) explicitly require $c \leq h$, so $c.n \leq h.n$.

Therefore, $c.n \leq h.n \leq b.n$. □

Remark 3.2.27. Lemma 3.2.26 may not hold while executing some steps. For instance, Step (2) may update h such that $b < h$, and Lemma 3.2.26 may not hold until we get to Step (8). Here, we only consider the state after completing all the actions associated to the new message.

Theorem 3.2.28. *Suppose $\varphi = \text{PREPARE}$. If $c \neq \mathbf{0}$, then c is the lowest ballot for which v has voted commit and not accepted abort.*

Proof.

Prove this! PREPARE messages encode a vote to *commit* $c, \cdots, h$ if $c \neq \mathbf{0}$. So, v has voted *commit* c for sure if $c \neq \mathbf{0}$. How do I know that it is the lowest such ballot? The accepted part is easy because as soon as v accepts commit, φ becomes CONFIRM.

□

Theorem 3.2.29. *If φ = CONFIRM, then h is the highest ballot for which v accepted commit.*

Proof. It suffices to show that, at any given moment,

(a) $\varphi \neq$ CONFIRM, or
(b) h is the highest ballot for which v accepted *commit*.

The invariant holds initially since (a) holds in the initial state. Suppose that the invariant holds in some state. There are $3 = 2^2 - 1$ ways in which this can be violated:

(i) Only (a) currently holds, and (a) becomes false.
(ii) Only (b) currently holds, and (b) becomes false.
(iii) Both (a) and (b) currently hold, and both become false.

We will consider each possibility separately:

(i) Suppose that only (a) holds currently. In other words, $\varphi \neq$ CONFIRM and h is not the highest ballot for which v accepted *commit*.
Finish this.

(ii)
Finish this.

(iii)
Finish this.

□

Theorem 3.2.30. *If $h \neq \mathbf{0}$, then $z = h.x$.*

Proof. It suffices to show that, at any given moment,

(a) $h = \mathbf{0}$, or
(b) $z = h.x$.

The invariant holds initially since (a) holds in the initial state. Suppose that the invariant holds in some state. There are $3 = 2^2 - 1$ ways in which this can be violated:

(i) Only (a) currently holds, and (a) becomes false.
(ii) Only (b) currently holds, and (b) becomes false.
(iii) Both (a) and (b) currently hold, and both become false.

Since no action sets $h \leftarrow \mathbf{0}$, (i) and (iii) are impossible. We will prove that (ii) is impossible. We modify z and/or h are Transition Type 1, and Step (2), (4), (6), and (7) in Transition Type 2.

- Transition Type 1: Since we are dealing with Case (ii), we assume $h \neq \mathbf{0}$. Then v will not perform Transition Type 1. In other words, v will not alter the value of z or $h.x$, so (b) cannot become false.
- Step (2) sets $z \leftarrow h.x$, so (b) holds after Step (2).
- Step (4) sets $z \leftarrow h.x$, so (b) holds after Step (4).
- Step (6)
  It looks like I need to know that $b \sim h$. If I know that, I can say that $h' \sim b \sim h$, so $h' \sim h$.

- Step (7)

  It looks like the value doesn't change once we get to COMMIT? I'm not quite sure how I can prove this step.

□

Theorem 3.2.31. *$p = p' = \mathbf{0}$ or $p' \lesssim p$.*

Proof. $p = p' = \mathbf{0}$ in the initial state, so this invariant holds in the initial state. We update p, p' only in Step (1) and (5).

- After Step (1), the invariant must hold because we update p, p' such that they are the two highest ballots accepted as prepared such that $p' \lesssim p$, where $p' = \mathbf{0}$ or $p = p' = \mathbf{0}$ if there are no such ballots.
  - If $p = p' = \mathbf{0}$, then we are done.
  - If $p' = \mathbf{0} \neq p$, then $p' \lesssim p$.
  - Otherwise, $p' \lesssim p$.
- Step (5):

  Interesting! I might need to prove other things first. $p \sim c$ makes me think that $c \neq \mathbf{0}$, but it's not obvious to me!

□

Theorem 3.2.32. *If $p' \neq \mathbf{0}$, then p, p' are the two highest ballots accepted as prepared such that $p' \lesssim p$.*

Proof.

prove this!

□