

Stellar Consensus Protocol

Saravanan Vijayakumaran
sarva@ee.iitb.ac.in

Department of Electrical Engineering
Indian Institute of Technology Bombay

September 25, 2018

Lecture Plan

- Consensus Protocol Terminology
- Related Protocols for Context
 - Paxos
 - PBFT
- Federated Byzantine Agreement Model
- Federated Voting
- Stellar Consensus Protocol (in brief)

Consensus Protocol Terminology

- **Agents:** Parties interested in achieving consensus
- Each agent has an input
- Agents use protocol to agree on one of the inputs
- Each agent decides on a chosen value
- Agent failure modes
 - Stopping failure
 - Byzantine failure
- **Safety**
 - **Agreement:** No two non-faulty agents decide on different values
 - **Validity:** If all non-faulty agents have the same input v , then v is the only possible decision value
- **Liveness**
 - **Termination:** All non-faulty agents eventually decide
- Asynchronous network model
 - Messages may be delayed, duplicated, lost, reordered
 - No corrupted messages

Paxos

Paxos

- Consensus protocol for non-Byzantine agents and asynchronous network
- Proposed by Leslie Lamport in 1989
- Number of agents is known
- Agents act as proposers, acceptors, or learners (multiple roles allowed)
- Proposers propose values
- Acceptors accept a value if requested by a proposer
- Once a majority of acceptors has accepted a value, consensus has been achieved
- Learners are interested in learning about consensus values
- Challenges
 - Messages indicating acceptance may be lost
 - Consensus may be achieved without proposers finding out
 - Multiple proposers may be simultaneously proposing values

Paxos Protocol Phase 1

- Proposal made by proposers have a proposal number n from a totally ordered set
- Phase 1
 - Proposer sends a **prepare** request with number n to all acceptors
 - If acceptor receives a prepare request with number higher than any other previous prepare request, then
 1. it promises to not accept any more proposals with number less than n and
 2. returns highest-numbered proposal value (if any) it has accepted
- Example

Prop. No.	Value	Agent 1	Agent 2	Agent 3
1	7	7	⟨ ⟩	⟨ ⟩
2	8	8	⟨ ⟩	⟨ ⟩
3	9	⟨ ⟩	⟨ ⟩	9

For proposal 4, highest-numbered proposal accepted among all responses is used

Paxos Protocol Phase 2

- Phase 2

- If proposer receives a response to its prepare request from a majority of acceptors, then it **either**
 - sends an **accept** request to each these acceptors with value v which is the highest-numbered proposal among the responses or
 - sends an **accept** request with any value if responses reported no proposals.
- If acceptor receives an accept request for a proposal number n , it accepts the proposal unless it has already responded to a prepare request having number greater than n .

- Example 1

Prop. No.	Value	Agent 1	Agent 2	Agent 3
1	7	7	⟨	⟨
2	8	8	⟨	⟨
3	9	⟨	⟨	9

- For proposal 4, proposer can send accept request with
 - 8 if only agents 1 and 2 respond
 - 9 if only agents 2 and 3 respond

Paxos Protocol Phase 2

- Phase 2

- If proposer receives a response to its prepare request from a majority of acceptors, then it **either**
 - sends an **accept** request to each these acceptors with values v which is the highest-numbered proposal among the responses or
 - sends an **accept** request with any value if responses reported no proposals.
- If acceptor receives an accept request for a proposal number n , it accepts the proposal unless it has already responded to a prepare request having number greater than n .

- Example 2

Prop. No.	Value	Agent 1	Agent 2	Agent 3
1	8	8	⟨	⟨
2	9	9	⟨	9
3	9	⟨	⟨	9

- For proposal 4, proposer can send accept request with only value 9

Paxos Protocol

- Phase 1
 - Proposer sends a **prepare** request with number n to all acceptors
 - If acceptor receives a prepare request with number higher than any other previous prepare request, then
 1. it promises to not accept any more proposals with number less than n and
 2. returns highest-numbered proposal value (if any) it has accepted
- Phase 2
 - If proposer receives a response to its prepare request from a majority of acceptors, then it **either**
 - sends an **accept** request to each these acceptors with values v which is the highest-numbered proposal among the responses or
 - sends an **accept** request with any value if responses reported no proposals.
 - If acceptor receives an accept request for a proposal number n , it accepts the proposal unless it has already responded to a prepare request having number greater than n .
- Learners need messages from a majority of acceptors to find out about consensus value

Proposer Selection

- Lamport describes a method using timeouts
 - Each agent broadcasts its ID and the one with the highest ID is the proposer
- Presence of multiple proposers cannot violate safety but can affect liveness
 - Proposer p completes phase 1 for proposal number n_1
 - Proposer q completes phase 1 for proposal number $n_2 > n_1$
 - Proposer p 's phase 2 messages are ignored
 - Proposer p completes phase 1 for new proposal with number $n_3 > n_2$
 - Proposer q 's phase 2 messages are ignored
 - And so on
- **FLP Impossibility Theorem:** No deterministic consensus algorithm can guarantee all three of safety, liveness, and fault-tolerance in an asynchronous system.

Practical Byzantine Fault Tolerance

PBFT

- Proposed in 1999 as an algorithm for state machine replication
 - Each agent is a replica of a state machine
 - Replicas need to achieve consensus on state transitions
- Assumes Byzantine agent failures and weak synchrony
 - Messages may be delayed, duplicated, lost, reordered
 - Delays do not grow faster than t indefinitely
- Guarantees safety and liveness if at most $\lfloor \frac{n-1}{3} \rfloor$ out of n replicas are faulty
 - For f faulty replicas, $3f + 1$ is the minimum number of replicas required
- Let \mathcal{R} be the set of replicas with cardinality $3f + 1$
- Each replica is identified using an integer in $0, 1, \dots, |\mathcal{R}| - 1$
- The algorithm moves through a sequence of **views**
- Views are numbered sequentially
- In view v , replica with identity $v \bmod |\mathcal{R}|$ is the **primary** and the remaining replicas are **backups**

PBFT Algorithm

- Rough outline
 1. A client sends a request to the primary to invoke a state machine operation
 2. Primary multicasts the request to the backups
 3. Replicas execute the request and send a reply to the client
 4. The client waits for $f + 1$ replies from different replicas with same result
- Three phases in case of non-faulty primary
 - Pre-prepare
 - Prepare
 - Commit
- Pre-prepare phase
 - Primary in view v receives client request m
 - Primary assigns a sequence number n to m
 - Primary multicasts PRE-PREPARE message with m, v, n to all backups
 - Backup accepts PRE-PREPARE message if
 - it is in view v and
 - it has not accepted a PRE-PREPARE message for view v and sequence number n with different request

PBFT Prepare Phase

- Prepare
 - If backup i accepts the PRE-PREPARE message, it enters the prepare phase
 - Multicasts PREPARE message with v, n, m, i to all other replicas
 - Adds both PRE-PREPARE and PREPARE messages to its log
- Define predicate **prepared**(m, v, n, i) to be true if and only if replica i has inserted in its log
 1. a PRE-PREPARE message with m, v, n , and
 2. at least $2f$ PREPARE messages for m, v, n .
- Guarantees that non-faulty replicas agree on total order of requests in a view
 - **Invariant:** If **prepared**(m, v, n, i) is true, then **prepared**(m', v, n, j) is false for any non-faulty replica j where $m' \neq m$
 - **prepared**(m, v, n, i) true \implies at least $f + 1$ non-faulty replicas have sent PREPARE or PRE-PREPARE messages for m, v, n
 - **prepared**(m', v, n, j) true \implies $2f + 1$ replicas have sent PREPARE or PRE-PREPARE messages for m', v, n to j
 - At least one non-faulty replica has sent conflicting PREPAREs or PRE-PREPAREs \implies contradiction

PBFT Commit Phase

- Commit
 - When **prepared**(m, v, n, i) becomes true, replica i multicasts a COMMIT message for m, v, n, i
 - Replicas accept COMMIT messages which match their view and insert them into their logs
 - Replica i executes the operation requested by m when **committed-local**(m, v, n, i) becomes true and all requests with lower sequence number have been executed
- **committed-local**(m, v, n, i) is true if and only if
 1. **prepared**(m, v, n, i) is true and
 2. replica i has accepted $2f + 1$ COMMITs (including its own) for m, v, n
- **committed**(m, v, n) is true if and only if **prepared**(m, v, n, j) is true for all j in some set of $f + 1$ non-faulty replicas
- **Invariant:** If **committed-local**(m, v, n, i) is true for some non-faulty i , then **committed**(m, v, n) is true
- At non-faulty replicas i and j , **committed-local**(m, v, n, i) and **committed-local**(m', v, n, j) cannot both be true for $m \neq m'$

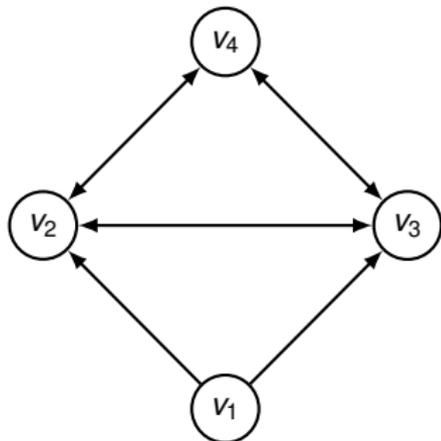
PBFT View Change

- View changes are required when primary replica fails
- View-change algorithm
 1. If client does not receive replies before a timeout, it broadcasts the request to all replicas
 2. If request has already been processed, the replicas resend the reply to client
 3. If request was not received from primary, a backup starts a timer upon receiving the client's request
 4. If the timer expires while waiting for same request from primary, the backup multicasts a view-change message to all replicas
 5. When primary of view $v + 1$ receives $2f$ view-change messages, it multicasts a new-view message and enters view $v + 1$

Federated Byzantine Agreement

Federated Byzantine Agreement

- **Definition:** An **federated Byzantine agreement system (FBAS)** is a pair $\langle \mathbf{V}, \mathbf{Q} \rangle$ comprising of a set of nodes \mathbf{V} and a quorum function $\mathbf{Q} : \mathbf{V} \mapsto 2^{2^{\mathbf{V}}} \setminus \{\emptyset\}$ specifying one or more quorum slices for each node, where a node belongs to all of its own quorum slices, i.e. $\forall v \in \mathbf{V}, \forall q \in \mathbf{Q}(v), v \in q$.
- Example

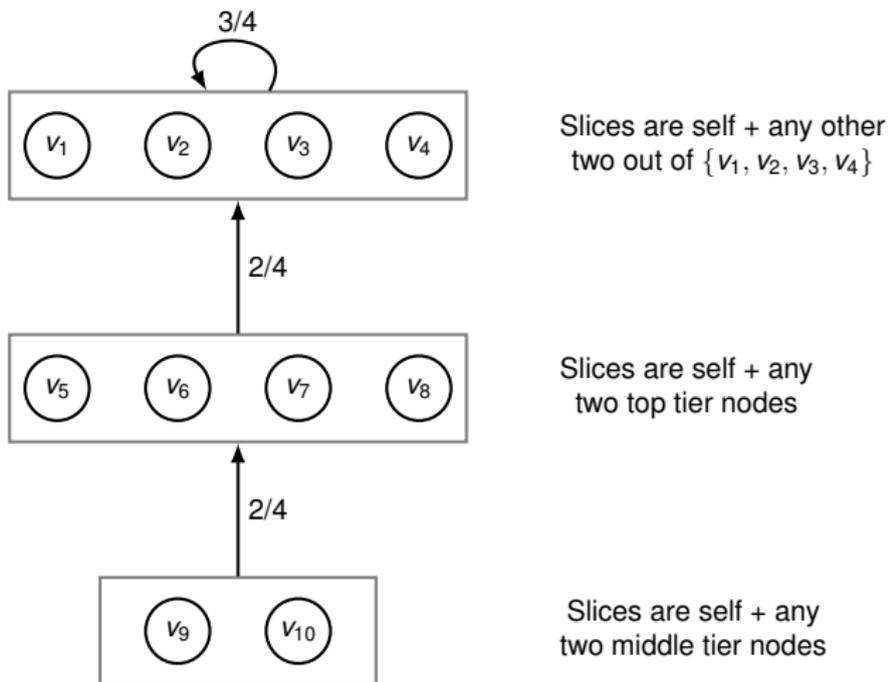


$$\mathbf{Q}(v_1) = \{\{v_1, v_2, v_3\}\}$$

$$\mathbf{Q}(v_2) = \mathbf{Q}(v_3) = \mathbf{Q}(v_4) = \{\{v_2, v_3, v_4\}\}$$

- **Definition:** A set of nodes $\mathbf{U} \subseteq \mathbf{V}$ in FBAS $\langle \mathbf{V}, \mathbf{Q} \rangle$ is a **quorum** iff $\mathbf{U} \neq \emptyset$ and \mathbf{U} contains a slice for each member, i.e. $\forall v \in \mathbf{U}, \exists q \in \mathbf{Q}(v)$ such that $q \subseteq \mathbf{U}$.
- A quorum of nodes is sufficient to reach agreement

Tiered FBAS Example



Possible quorums?

Safety and Liveness

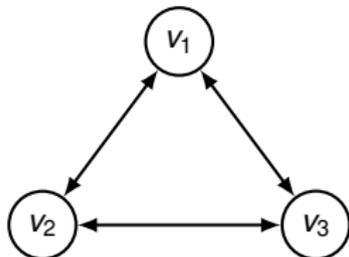
- FBA systems attempt consensus in a slot
- A node applies update x in slot i when
 1. it has applied updates in all previous slots and
 2. it believes all non-faulty nodes will eventually agree on x for slot i .

The node is said to have **externalized** x in slot i .

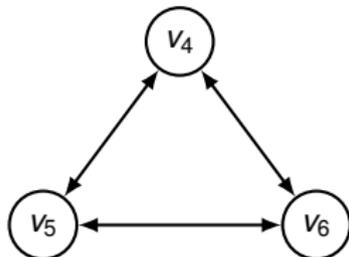
- **Definition:** A set of nodes in an FBAS enjoy **safety** if no two of them ever externalize different values for the same slot
- Well-behaved nodes = obey protocol
- Ill-behaved nodes = Byzantine failures
- Well-behaved nodes can also fail (be blocked or diverge)
- **Definition:** A node in an FBAS enjoys **liveness** if it can externalize new values without the participation of any failed nodes
- Given a specific $\langle \mathbf{V}, \mathbf{Q} \rangle$ and an ill-behaved subset of \mathbf{V} , what is the best any FBA protocol can do?

Quorum Intersection

- **Definition:** An FBAS enjoys **quorum intersection** if and only if any two quorums share a node.
- No protocol can guarantee safety in the absence of quorum intersection
- Example of quorum non-intersection



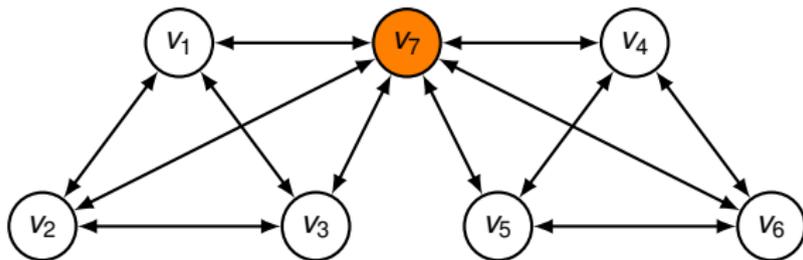
$$\begin{aligned} \mathbf{Q}(v_1) &= \mathbf{Q}(v_2) = \mathbf{Q}(v_3) \\ &= \{\{v_1, v_2, v_3\}\} \end{aligned}$$



$$\begin{aligned} \mathbf{Q}(v_4) &= \mathbf{Q}(v_5) = \mathbf{Q}(v_6) \\ &= \{\{v_4, v_5, v_6\}\} \end{aligned}$$

- $\{v_1, v_2, v_3\}$ and $\{v_4, v_5, v_6\}$ are two disjoint quorums; can approve contradictory statements

Quorum Intersection at Ill-Behaved Nodes



$$\begin{aligned} \mathbf{Q}(v_1) &= \mathbf{Q}(v_2) = \mathbf{Q}(v_3) \\ &= \{\{v_1, v_2, v_3, v_7\}\} \end{aligned}$$

$$\begin{aligned} \mathbf{Q}(v_4) &= \mathbf{Q}(v_5) = \mathbf{Q}(v_6) \\ &= \{\{v_4, v_5, v_6, v_7\}\} \end{aligned}$$

$$\mathbf{Q}(v_7) = \{\{v_1, v_2, v_3, v_7\}, \{v_4, v_5, v_6, v_7\}\}$$

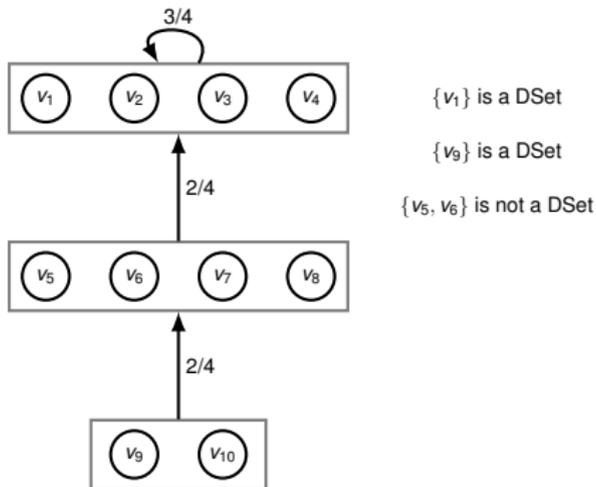
- If v_7 is ill-behaved, the quorums are effectively disjoint
- **Necessary property for safety:** Well-behaved nodes enjoy quorum intersection after deleting ill-behaved nodes
- **Definition:** If $\langle \mathbf{V}, \mathbf{Q} \rangle$ is an FBAS and $B \subseteq \mathbf{V}$ is a set of nodes, to **delete** B is to compute the modified FBAS $\langle \mathbf{V}, \mathbf{Q} \rangle^B = \langle \mathbf{V} \setminus B, \mathbf{Q}^B \rangle$ where $\mathbf{Q}^B = \{q \setminus B \mid q \in \mathbf{Q}(v)\}$

Dispensible Sets

- Safety and liveness of nodes outside a DSet can be guaranteed irrespective of the behaviour of nodes in the DSet
- **Definition:** Let $\langle \mathbf{V}, \mathbf{Q} \rangle$ be an FBAS and $B \subseteq \mathbf{V}$ be a set of nodes. We say B is a dispensible set or DSet if and only if
 1. $\langle \mathbf{V}, \mathbf{Q} \rangle^B$ enjoys quorum intersection, and
 2. either $\mathbf{V} \setminus B$ is a quorum in $\langle \mathbf{V}, \mathbf{Q} \rangle$ or $B = \mathbf{V}$.

Condition 1 = *quorum intersection despite B*

Condition 2 = *quorum availability despite B*



Intact and Befouled Nodes

- **Definition:** A node v in an FBAS is **intact** iff there exists a DSet B containing all ill-behaved nodes such that $v \notin B$
- An optimal FBAS should guarantee safety/liveness for every intact node
- **Definition:** A node v in an FBAS is **befouled** iff it is not intact
- **Theorem:** In an FBAS with quorum intersection, the set of befouled nodes is a DSet
 - Proof follows from a theorem which says that intersection of DSets is a DSet in an FBAS with quorum intersection

Federated Voting

Voting and Ratification

- **Definition:** A node v **votes** for a statement A if and only if
 1. v asserts A is valid and consistent with all statements v has accepted, and
 2. v asserts that it has never voted against A and promises to not vote against A in the future.
- **Definition:** 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 iff v is a member of a quorum U_A that ratifies A .
- Theorems
 - Two contradictory statements A and \bar{A} cannot both be ratified in an FBAS that enjoys quorum intersection and contains no ill-behaved nodes.
 - Let $\langle \mathbf{V}, \mathbf{Q} \rangle$ be an FBAS enjoying quorum intersection despite B where B contains all ill-behaved nodes. Let $v_1, v_2 \notin B$. If v_1 ratifies A , then v_2 cannot ratify \bar{A} .
 - Two intact nodes in an FBAS with quorum intersection cannot ratify contradictory statements.

Accepting Statements

- **Definition:** Let $v \in \mathbf{V}$ be a node in FBAS $\langle \mathbf{V}, \mathbf{Q} \rangle$. A set $B \subseteq \mathbf{V}$ is **v -blocking** iff it overlaps with every one of v 's slices
- **Theorem:** Let $B \subseteq \mathbf{V}$ be a set of nodes in FBAS $\langle \mathbf{V}, \mathbf{Q} \rangle$. $\langle \mathbf{V}, \mathbf{Q} \rangle$ enjoys quorum availability despite B iff B is not v -blocking for any $v \in \mathbf{V} \setminus B$.
- **Corollary:** The DSet of befouled nodes is not v -blocking for any intact v in an FBAS enjoying quorum intersection.
- **Definition:** An FBAS node v **accepts** a statement A iff it has never accepted a statement contradicting A and it determines that either
 1. There exists a quorum U such that $v \in U$ and each each member of U either voted for A or claims to accept A , **or**
 2. each member of a v -blocking set claims to accept A .
- Second condition allows v to vote for one statement but later accept a contradictory one
- **Theorem:** Two intact nodes in an FBAS that enjoys quorum intersection cannot accept contradictory statements.
- Acceptance of a statement at an intact node does not guarantee that all other intact nodes will accept

Confirming Statements

- **Definition:** A quorum U_A in an FBAS **confirms** a statement A if and only if every member of U_A claims to accept A . A node v **confirms** A if and only if it is in such a quorum.
- **Theorem:** Let $\langle \mathbf{V}, \mathbf{Q} \rangle$ be an FBAS enjoying quorum intersection despite B where B contains all ill-behaved nodes. Let $v_1, v_2 \notin B$. If v_1 confirms A , then v_2 cannot confirm \bar{A} .
- **Theorem:** If an intact node in an FBAS $\langle \mathbf{V}, \mathbf{Q} \rangle$ with quorum intersection confirms a statement A , then, whatever subsequently transpires, once sufficient messages are delivered and processed, every intact node will accept and confirm A .
- But the protocol may get stuck before an intact node confirmation
- Need multiple rounds for liveness

Stellar Consensus Protocol

- Two subprotocols
 - Nomination protocol
 - Ballot protocol
- Nodes nominate candidate values for a slot which will converge on a composite value
 - Composite value = Union of transaction sets proposed
- Ballot protocol uses federated voting to commit and abort ballots of composite values

References

- SCP talk <https://www.youtube.com/watch?v=vmwnhZmEZjc>
- SCP white paper <https://www.stellar.org/papers/stellar-consensus-protocol.pdf>
- *Paxos Made Simple*, Leslie Lamport, <https://lamport.azurewebsites.net/pubs/paxos-simple.pdf>
- *How to Build a Highly Available System Using Consensus*, B. W. Lampson, https://doi.org/10.1007/3-540-61769-8_1
- PBFT paper <http://www.pmg.csail.mit.edu/papers/osdi99.pdf>