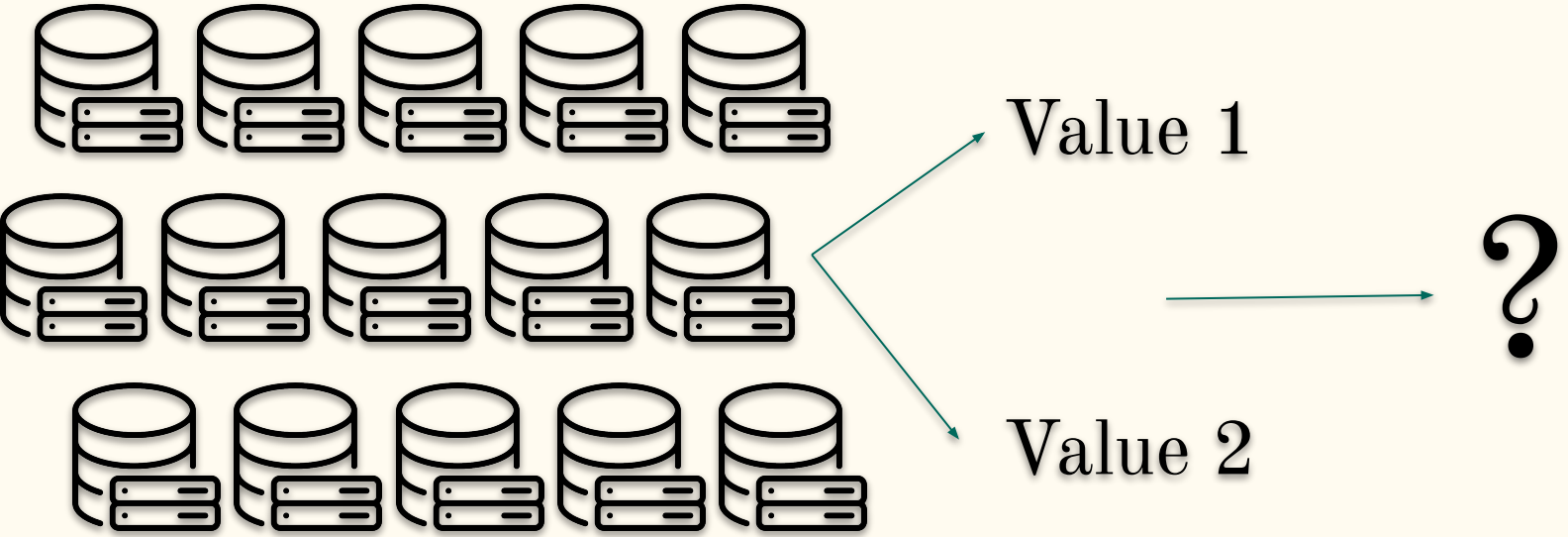
# Formal Specification of Fast Paxos in TLA+

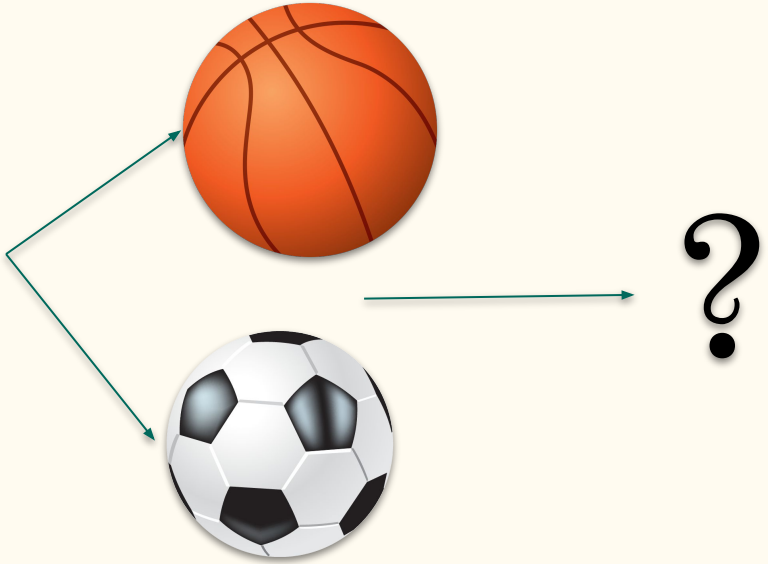
—

By Gaurav Gandhi & Lim Ngian Xin Terry

# What's the need of Consensus?



# An Analogy



# Paxos Overview

Roles:

- Coordinator: Moves the protocol forward when conflict occurs.
- Proposer: Proposes a value.
- Acceptor: Votes on a value to be chosen.
- Learner: Acts as the replication factor for the protocol.

Phases

- Prepare (P1a): “Prepare to receive a proposal from me.”
- Promise (P1b): “I promise to ignore anyone before you.”
- Accept (P2a): “Please accept this value I am proposing.”
- Accepted (P2b): “I accept the value you proposed.”

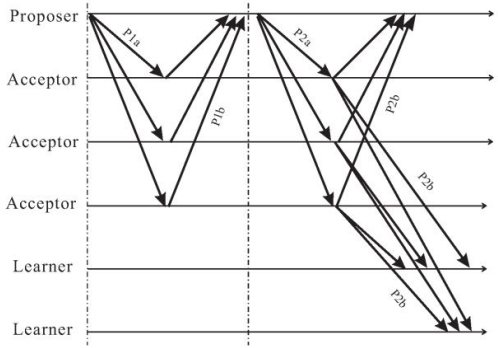
# Our Classic Paxos Model

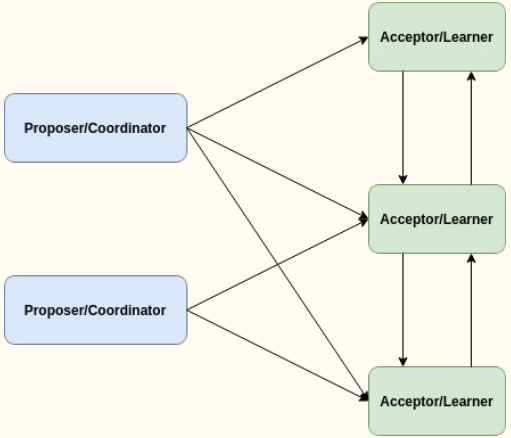
- 2 explicit roles: Proposer & Acceptor
- Proposers are also Coordinators.
- Acceptors are also Learners.
- Quorum $= n/2 + 1$, where $n$ is the number of acceptors.
- All nodes in the system can communicate with one another.

VARIABLES *messages* Set of all messages sent.
VARIABLES *decision* Decided value of an acceptor.
VARIABLES *maxBallot* Maximum ballot an acceptor has seen.
VARIABLES *maxVBallot* Maximum ballot an acceptor has accepted.
VARIABLES *maxValue* Maximum value an acceptor has accepted.

$$
\begin{aligned}
&P1aMessage \triangleq [type : \{\text{“P1a”}\},\\
&\qquad ballot : Ballots \setminus \{0\}]\\
&P1bMessage \triangleq [type : \{\text{“P1b”}\},\\
&\qquad ballot : Ballots,\\
&\qquad acceptor : Replicas,\\
&\qquad maxVBallot : Ballots,\\
&\qquad maxValue : Values \cup \{none\}] \ (maxVBallot = 0) \equiv (maxValue = none)\\
&P2aMessage \triangleq [type : \{\text{“P2a”}\},\\
&\qquad ballot : Ballots,\\
&\qquad value : Values \cup \{any\}]\\
&P2bMessage \triangleq [type : \{\text{“P2b”}\},\\
&\qquad ballot : Ballots,\\
&\qquad acceptor : Replicas,\\
&\qquad value : Values]\\
&Message \triangleq P1aMessage \cup P1bMessage \cup P2aMessage \cup P2bMessage
\end{aligned}
$$

*Figure 1. Normal operation of the Paxos algorithm*

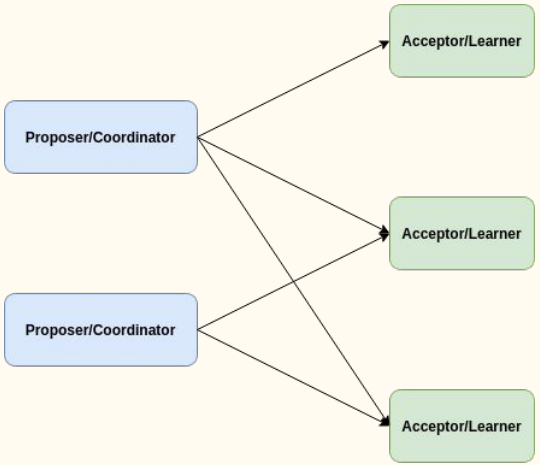


# Prepare Phase (P1a)

- Proposer *broadcasts* a Prepare (P1a) message.
- Message only contains ballot number.
- Message doesn't contain Proposer. Why?
  - Possible for 2 Proposers to use the same ballot number for Prepare (P1a) message.
  - Impossible for more than 1 Proposer to receive a majority of Promise (P1b) replies.
  - Unnecessary to explicitly model which Proposer the message is from.
- Messages can be sent in random ballot order.
  - System is asynchronous. A message may be delayed, dropped or received out of order.

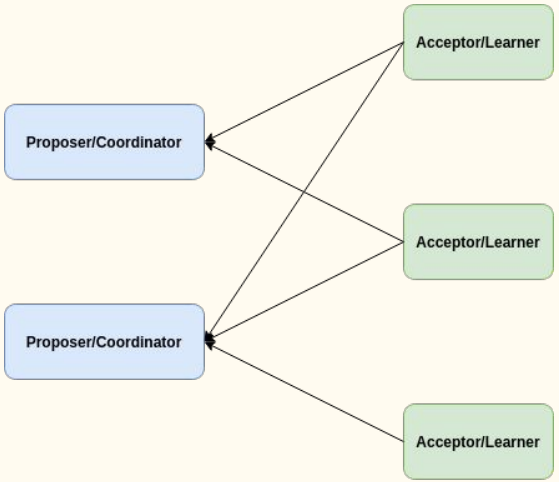
$$
\begin{aligned}
&PaxosPrepare \triangleq \\
&\quad \land \text{UNCHANGED } \langle decision,\ maxBallot,\ maxVBallot,\ maxValue \rangle \\
&\quad \land \exists\, b \in Ballots \setminus \{0\} : \\
&\qquad SendMessage([type \mapsto \text{“P1a”}, \\
&\qquad\qquad ballot \mapsto b])
\end{aligned}
$$



# Promise Phase (P1b)

- If an Acceptor *a* receives a Prepare (P1a) message *m* such that *maxBallot[a]* $<$ *m.ballot*, reply with a Promise (P1b) message.
- If Acceptor had already sent an Accept (P2b) message from a smaller ballot, attach the ballot *maxVBallot[a]* and value *maxValue[a]* to the Promise (P1b) reply.

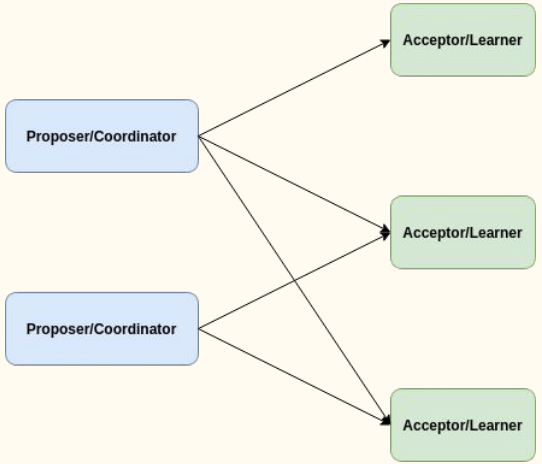
$$
\begin{aligned}
&PaxosPromise \triangleq \\
&\quad \land \text{UNCHANGED } \langle decision, maxVBallot, maxValue \rangle \\
&\quad \land \exists\, a \in Replicas, m \in p1aMessages : \\
&\qquad \land maxBallot[a] < m.ballot \\
&\qquad \land maxBallot' = [maxBallot \text{ EXCEPT } ![a] = m.ballot] \\
&\qquad \land SendMessage([type \mapsto \text{“P1b”}, \\
&\qquad\qquad ballot \mapsto m.ballot, \\
&\qquad\qquad acceptor \mapsto a, \\
&\qquad\qquad maxVBallot \mapsto maxVBallot[a], \\
&\qquad\qquad maxValue \mapsto maxValue[a]])
\end{aligned}
$$



# Accept Phase (P2a)

- If a Proposer receives a quorum of Promise (P1b) replies, *broadcast* an Accept (P2a) message with a proposed value $v$.
- If all Acceptors in the quorum has *not* accepted any values before, $v$ can be anything.
- Else, $v$ is the *maxValue* with the highest associated *maxVBallot* in received Promise (P1b) replies.

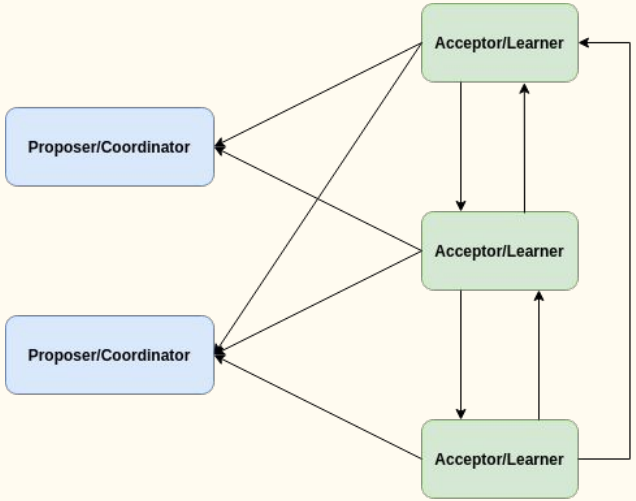
$$
\begin{array}{l}
PaxosAccept \triangleq \\
\quad \land \text{UNCHANGED } \langle decision, maxBallot, maxVBallot, maxValue \rangle \\
\quad \land \exists\, b \in Ballots, q \in Quorums, v \in Values : \\
\qquad \land \forall\, m \in p2aMessages : \neg(m.ballot = b) \\
\qquad \land \text{LET } M \triangleq \{m \in p1bMessages : m.ballot = b \land m.acceptor \in q\} \\
\qquad\quad \text{IN} \quad \land \forall\, a \in q : \exists\, m \in M : m.acceptor = a \\
\qquad\qquad\quad \land \lor \forall\, m \in M : m.maxValue = none \\
\qquad\qquad\qquad\ \lor v = ForcedValue(M) \\
\qquad\qquad\quad \land SendMessage([type \mapsto \text{“P2a”}, \\
\qquad\qquad\qquad\qquad\qquad ballot \mapsto b, \\
\qquad\qquad\qquad\qquad\qquad value \mapsto v])
\end{array}
$$



# Accepted Phase (P2b)

- If an Acceptor $a$ receives an Accept (P2a) message $m$ such that $maxBallot[a] \leqslant m.ballot$, accept the proposed value $v$, and *broadcast* an Accepted (P2b) message with $a$ and $v$ attached.

$$
\begin{aligned}
&PaxosAccepted \triangleq \\
&\quad \land \text{UNCHANGED } \langle decision \rangle \\
&\quad \land \exists\, a \in Replicas,\, m \in p2aMessages : \\
&\qquad \land\, m.value \in Values \\
&\qquad \land\, maxBallot[a] \leq m.ballot \\
&\qquad \land\, maxBallot' = [maxBallot \text{ EXCEPT } ![a] = m.ballot] \\
&\qquad \land\, maxVBallot' = [maxVBallot \text{ EXCEPT } ![a] = m.ballot] \\
&\qquad \land\, maxValue' = [maxValue \text{ EXCEPT } ![a] = m.value] \\
&\qquad \land\, SendMessage([type \mapsto \text{“P2b”}, \\
&\qquad\qquad\qquad ballot \mapsto m.ballot, \\
&\qquad\qquad\qquad acceptor \mapsto a, \\
&\qquad\qquad\qquad value \mapsto m.value])
\end{aligned}
$$



# Decide (End)

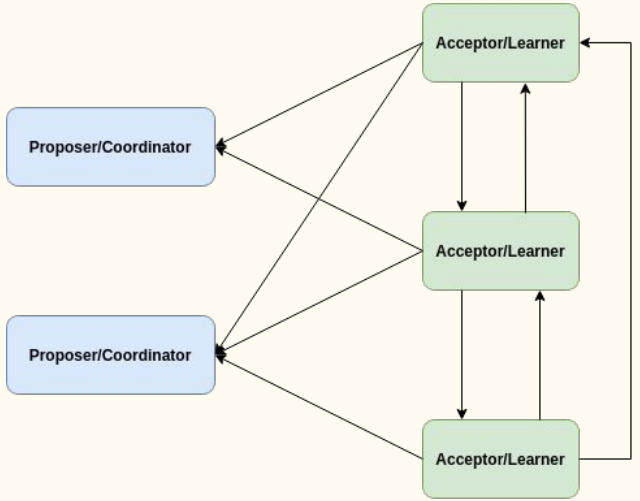
- If an Acceptor receives a quorum of Accepted (P2b) messages for a ballot, it decides on that value. The value is said to be learnt.
- Non-triviality: Only proposed values can be learnt.

$$
\begin{array}{l}
PaxosNontriviality \triangleq \\
\quad \land \lor decision = none \\
\qquad \lor \exists\, m \in p2aMessages : m.value = decision \\
\quad \land \forall\, m \in p1bMessages : \land\, m.maxValue \in Values \lor 0 = m.maxVBallot \\
\qquad\qquad\qquad\qquad\qquad\quad \land\, m.maxValue = none \lor 0 < m.maxVBallot
\end{array}
$$

- Consistency: At most 1 value can be learnt.

$$PaxosConsistency \triangleq decision = none \lor decision = decision'$$

$$
\begin{array}{l}
PaxosDecide \triangleq \\
\quad \land \text{UNCHANGED } \langle messages, maxBallot, maxVBallot, maxValue \rangle \\
\quad \land \exists\, b \in Ballots, q \in Quorums : \\
\qquad \text{LET } M \triangleq \{m \in p2bMessages : m.ballot = b \land m.acceptor \in q\} \\
\qquad \text{IN} \quad \land \forall\, a \in q : \exists\, m \in M : m.acceptor = a \\
\qquad\qquad\quad \land \exists\, m \in M : decision' = m.value
\end{array}
$$



# Fast Paxos Overview

New Phases

- Fast Any (P2a): “Propose and accept any value.”
- Fast Propose (P2b): “I propose and accept this value.”

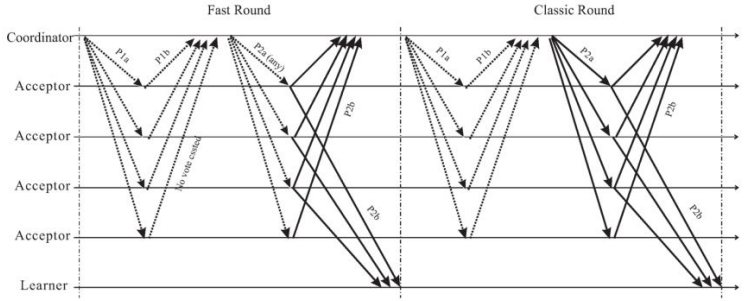
Differences:

- Prepare (P1a) and Promise (P1b) phase no longer needed.
- Two possible cases:
  - Case 1: No collision. In a fast quorum of acceptors, only 1 value was proposed/accepted.
  - Case 2: Collision. In a fast quorum of acceptors, more than 1 value was proposed/accepted.
- When no collision, faster than classic Paxos.
- When collision, fall back to using Classic Paxos.

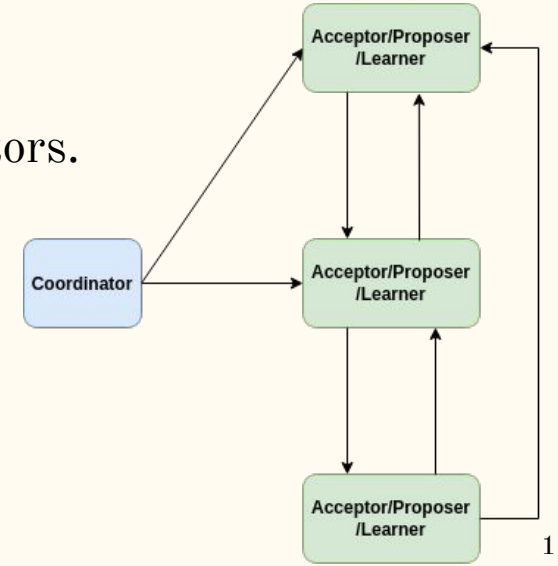
# Our Fast Paxos Model

- 2 explicit roles: Coordinator & Acceptors.
- There is only 1 Coordinator.
- Acceptors are also Proposers and Learners.
- Same messages as Classic Paxos.
- No Prepare (P1a) & Promise (P1b) phase.
- Fast Quorum = $3n/4$, where n is the number of acceptors.

*Figure 3. Fast Paxos operates in rounds with each round consisting of two phases. A round can be a classic round, where the coordinator selects a value to be voted on, or a fast round, where each acceptor is allowed to propose its own value. The dotted arrowed lines means that they can be omitted when a unique coordinator exists in the system.*



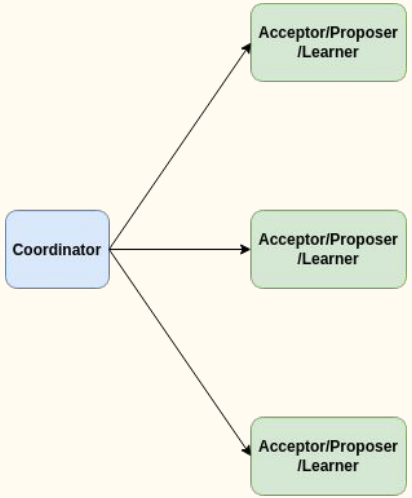

VARIABLES *messages* Set of all messages sent.
VARIABLES *decision* Decided value of an acceptor.
VARIABLES *maxBallot* Maximum ballot an acceptor has seen.
VARIABLES *maxVBallot* Maximum ballot an acceptor has accepted.
VARIABLES *maxValue* Maximum value an acceptor has accepted.
VARIABLES *cValue* Value chosen by coordinator.



# Fast Any Phase (P2a)

- Coordinator initiates a fast round if there are currently no proposed values, by sending an Fast Any (P2a) message to Acceptors.

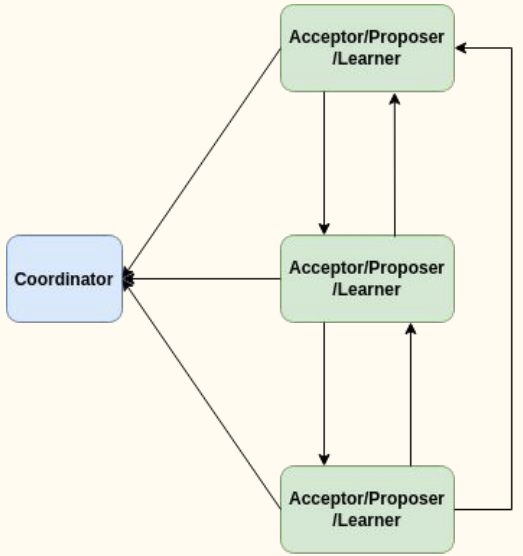
$$
\begin{aligned}
&\mathit{FastAny} \triangleq \\
&\quad \land \text{UNCHANGED } \langle \mathit{decision}, \mathit{maxBallot}, \mathit{maxVBallot}, \mathit{maxValue}, \mathit{cValue} \rangle \\
&\quad \land \exists f \in \mathit{FastBallots} : \\
&\qquad \land \mathit{SendMessage}([\mathit{type} \mapsto \text{“P2a”}, \\
&\qquad\qquad\qquad \mathit{ballot} \mapsto f, \\
&\qquad\qquad\qquad \mathit{value} \mapsto \mathit{any}])
\end{aligned}
$$



# Fast Propose Phase (P2b)

- If an Acceptor *a* receives an Fast Any (P2a) message *m* such that *maxBallot[a]* $\leqslant$ *m.ballot*, it will propose a value *v*, and *broadcast* a Fast Propose (P2b) message with *a* and *v* attached.

$$
\begin{aligned}
&\mathit{FastPropose} \triangleq \\
&\quad \land \text{UNCHANGED } \langle \mathit{decision}, \mathit{cValue} \rangle \\
&\quad \land \exists\, a \in \mathit{Replicas}, m \in \mathit{p2aMessages}, v \in \mathit{Values} : \\
&\qquad \land m.\mathit{value} = \mathit{any} \\
&\qquad \land \mathit{maxBallot}[a] \leq m.\mathit{ballot} \\
&\qquad \land \mathit{maxValue}[a] = \mathit{none} \lor \mathit{maxValue}[a] = v \\
&\qquad \land \mathit{maxBallot}' = [\mathit{maxBallot} \text{ EXCEPT } ![a] = m.\mathit{ballot}] \\
&\qquad \land \mathit{maxVBallot}' = [\mathit{maxVBallot} \text{ EXCEPT } ![a] = m.\mathit{ballot}] \\
&\qquad \land \mathit{maxValue}' = [\mathit{maxValue} \text{ EXCEPT } ![a] = v] \\
&\qquad \land \forall\, n \in \mathit{p2bMessages} : \neg(n.\mathit{ballot} \quad = m.\mathit{ballot} \land n.\mathit{acceptor} = a) \\
&\qquad \land \mathit{SendMessage}([\mathit{type} \mapsto \text{“P2b”}, \\
&\qquad\qquad\qquad \mathit{ballot} \mapsto m.\mathit{ballot}, \\
&\qquad\qquad\qquad \mathit{acceptor} \mapsto a, \\
&\qquad\qquad\qquad \mathit{value} \mapsto v])
\end{aligned}
$$



# Fast Decide (No Collision)

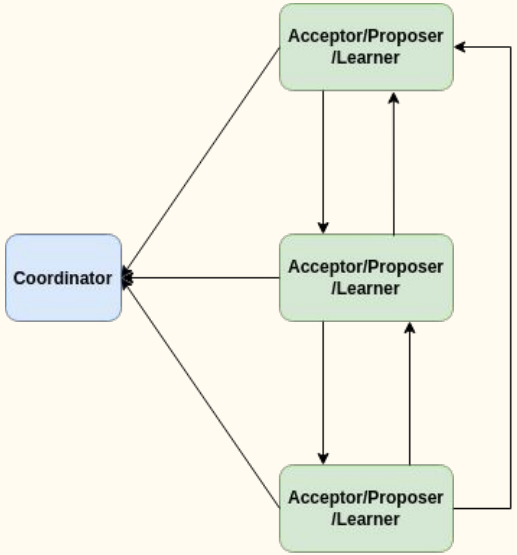
$$
\begin{aligned}
&FastDecide \triangleq \\
&\quad \land \text{UNCHANGED } \langle messages, maxBallot, maxVBallot, maxValue, cValue \rangle \\
&\quad \land \exists\, b \in FastBallots,\ q \in FastQuorums : \\
&\qquad \text{LET } M \triangleq \{m \in p2bMessages : m.ballot = b \land m.acceptor \in q\} \\
&\qquad\quad\ V \triangleq \{w \in Values : \exists\, m \in M : w = m.value\} \\
&\qquad \text{IN } \ \land \forall\, a \in q : \exists\, m \in M : m.acceptor = a \\
&\qquad\quad\ \land 1 = Cardinality(V) \\
&\qquad\quad\ \land \exists\, m \in M : decision' = m.value
\end{aligned}
$$

- If an Acceptor receives a *fast* quorum of Fast Propose (P2b) messages for a ballot of a fast round, and every message, proposes the same value $v$, it decides on $v$. The value $v$ is said to be learnt.
- Non-triviality: Only proposed values can be learnt.

$$
\begin{aligned}
FastNontriviality \triangleq\ & \lor decision = none \\
& \lor \exists\, m \in p2bMessages : m.value = decision \land m.ballot \in FastBallots
\end{aligned}
$$

- Consistency: At most 1 value can be learnt.

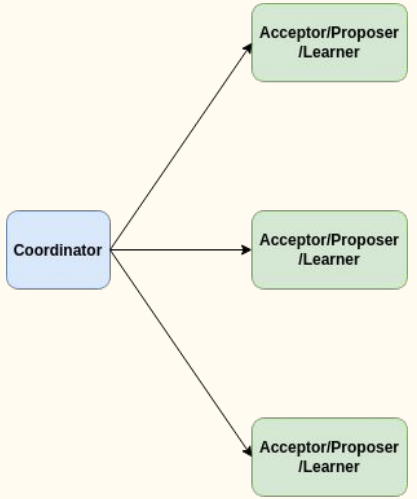
$$PaxosConsistency \triangleq decision = none \lor decision = decision'$$



# Classic Accept Phase (P2a)

- For fast ballot $f$, a fast quorum of Fast Propose (P2b) messages has more than 1 proposed value.
- Resolve collision via following rules:
  - If the messages contain different values, a value $v$ must be selected by the coordinator if the majority of acceptors in the fast quorum proposed $v$.
  - Otherwise, the coordinator is free to select any value proposed.
- After selecting a value $v$, the Coordinator will start a classic round by sending a Classic Accept (P2a) message with ballot $b$ such that $f < b$, and with value $v$.

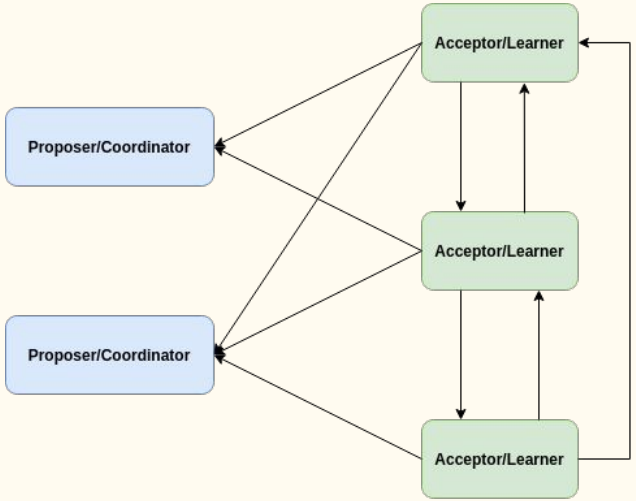
$$
\begin{aligned}
&ClassicAccept \triangleq \\
&\quad \land \text{UNCHANGED } \langle decision, maxBallot, maxVBallot, maxValue \rangle \\
&\quad \land \exists\, b \in ClassicBallots, f \in FastBallots, q \in FastQuorums, v \in Values : \\
&\qquad \land f < b \quad \text{There was a fast round before this classic round.} \\
&\qquad \land cValue = none \lor cValue = v \\
&\qquad \land cValue' = v \\
&\qquad \land \forall\, m \in p2aMessages : m.ballot \neq b \\
&\qquad \land \text{LET } M \triangleq \{m \in p2bMessages : m.ballot = f \land m.acceptor \in q\} \\
&\qquad\qquad V \triangleq \{w \in Values : \exists\, m \in M : w = m.value\} \\
&\qquad\quad \text{IN} \quad \land \forall\, a \in q : \exists\, m \in M : m.acceptor = a \\
&\qquad\qquad\quad \land 1 < Cardinality(V) \quad \text{Collision occured.} \\
&\qquad\qquad\quad \land \text{IF } \exists\, w \in V : IsMajorityValue(M, w) \\
&\qquad\qquad\qquad\quad \text{THEN } IsMajorityValue(M, v) \quad \text{Choose majority in quorum.} \\
&\qquad\qquad\qquad\quad \text{ELSE } v \in V \quad \text{Choose any.} \\
&\qquad\qquad\quad \land SendMessage([type \mapsto \text{“P2a”}, \\
&\qquad\qquad\qquad\qquad\qquad ballot \mapsto b, \\
&\qquad\qquad\qquad\qquad\qquad value \mapsto v])
\end{aligned}
$$



# Classic Accepted Phase (P2b)

- If an Acceptor $a$ receives an Accept (P2a) message $m$ such that $maxBallot[a] \leqslant m.ballot$, accept the proposed value $v$, and *broadcast* an Accepted (P2b) message with $a$ and $v$ attached.

$$
\begin{aligned}
&PaxosAccepted \triangleq \\
&\quad \land \text{UNCHANGED } \langle decision \rangle \\
&\quad \land \exists\, a \in Replicas,\, m \in p2aMessages : \\
&\qquad \land m.value \in Values \\
&\qquad \land maxBallot[a] \leq m.ballot \\
&\qquad \land maxBallot' = [maxBallot \text{ EXCEPT } ![a] = m.ballot] \\
&\qquad \land maxVBallot' = [maxVBallot \text{ EXCEPT } ![a] = m.ballot] \\
&\qquad \land maxValue' = [maxValue \text{ EXCEPT } ![a] = m.value] \\
&\qquad \land SendMessage([type \mapsto \text{“P2b”}, \\
&\qquad\qquad\qquad ballot \mapsto m.ballot, \\
&\qquad\qquad\qquad acceptor \mapsto a, \\
&\qquad\qquad\qquad value \mapsto m.value])
\end{aligned}
$$



# Decide (Collision)

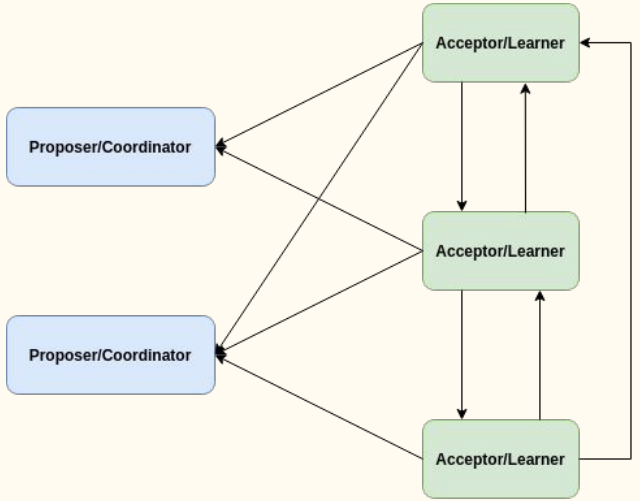
- If an Acceptor receives a quorum of Accepted (P2b) messages for a ballot, it decides on that value. The value is said to be learnt.
- Non-triviality: Only proposed values can be learnt.

$$
\begin{array}{l}
PaxosNontriviality \triangleq \\
\quad \land \lor decision = none \\
\qquad \lor \exists\, m \in p2aMessages : m.value = decision \\
\quad \land \forall\, m \in p1bMessages : \land\, m.maxValue \in Values \lor 0 = m.maxVBallot \\
\qquad\qquad\qquad\qquad\qquad\quad \land\, m.maxValue = none \lor 0 < m.maxVBallot
\end{array}
$$

- Consistency: At most 1 value can be learnt.

$$PaxosConsistency \triangleq decision = none \lor decision = decision'$$

$$
\begin{array}{l}
PaxosDecide \triangleq \\
\quad \land \text{UNCHANGED } \langle messages, maxBallot, maxVBallot, maxValue \rangle \\
\quad \land \exists\, b \in Ballots, q \in Quorums : \\
\qquad \text{LET } M \triangleq \{m \in p2bMessages : m.ballot = b \land m.acceptor \in q\} \\
\qquad \text{IN} \quad \land \forall\, a \in q : \exists\, m \in M : m.acceptor = a \\
\qquad\qquad\quad \land \exists\, m \in M : decision' = m.value
\end{array}
$$



# FLP Impossibility Theorem vs Paxos

- The FLP Impossibility theorem states that in an asynchronous network where messages may be delayed but not lost, there is no consensus algorithm that is guaranteed to terminate in every execution for all starting conditions, if at least one node may experience failure.
- But Paxos does not guarantee termination!