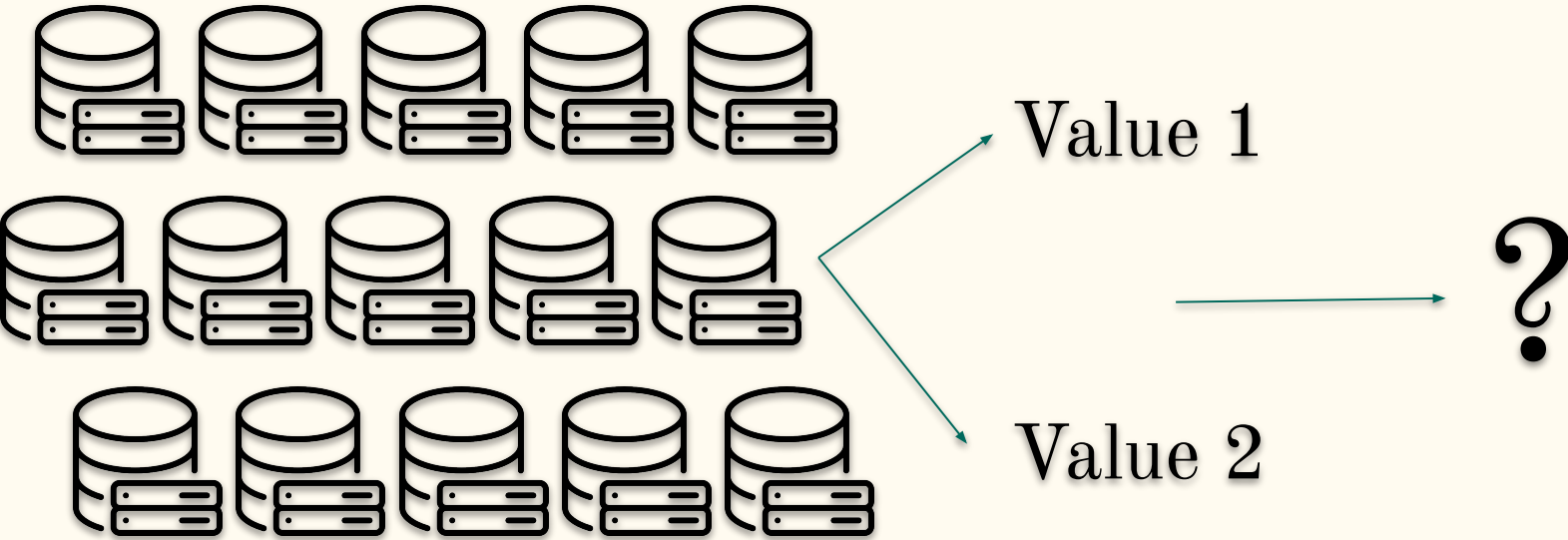


# Formal Specification of Fast Paxos in TLA+

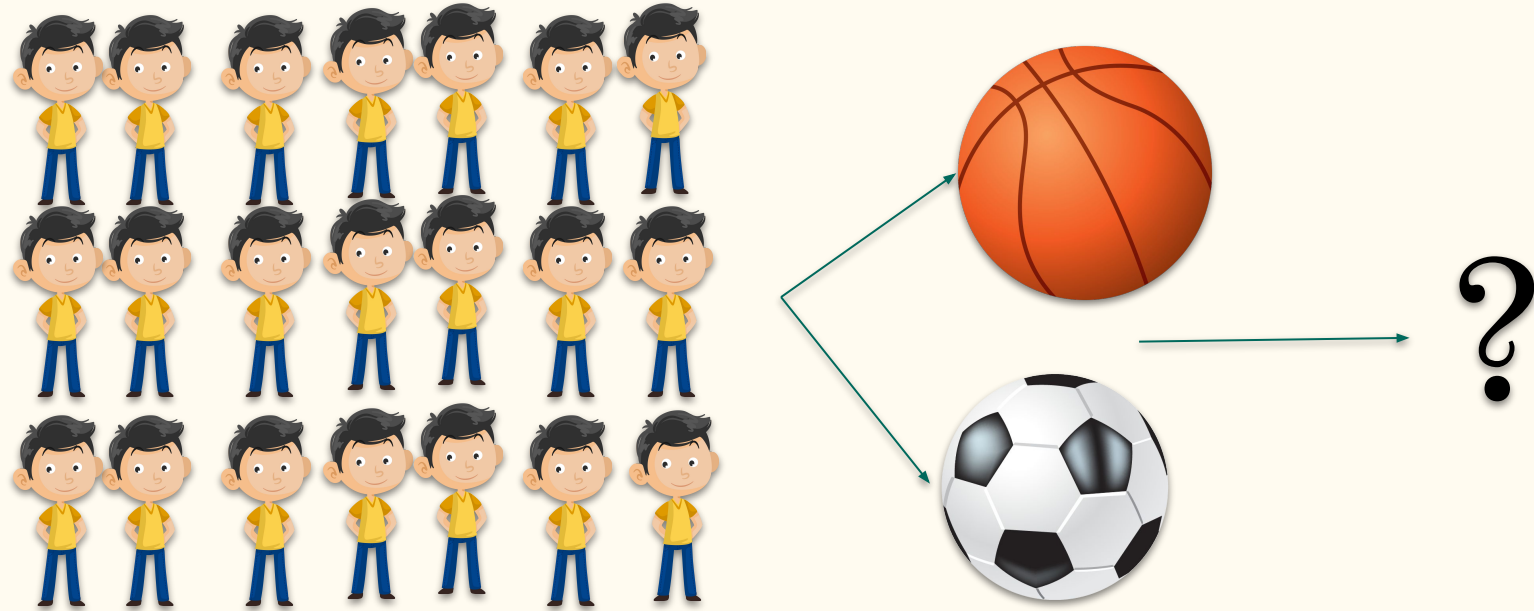
---

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.

$P1aMessage \triangleq [type : \{ "P1a" \},$   
 $ballot : Ballots \setminus \{0\}]$

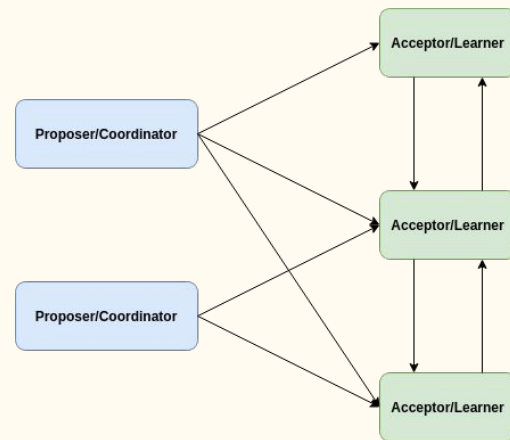
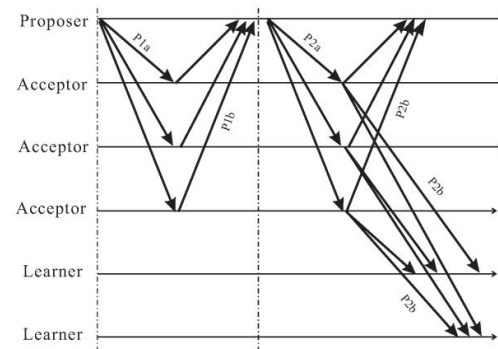
$P1bMessage \triangleq [type : \{ "P1b" \},$   
 $ballot : Ballots,$   
 $acceptor : Replicas,$   
 $maxVBallot : Ballots,$   
 $maxValue : Values \cup \{none\}] \ (maxVBallot = 0) \equiv (maxValue = none)$

$P2aMessage \triangleq [type : \{ "P2a" \},$   
 $ballot : Ballots,$   
 $value : Values \cup \{any\}]$

$P2bMessage \triangleq [type : \{ "P2b" \},$   
 $ballot : Ballots,$   
 $acceptor : Replicas,$   
 $value : Values]$

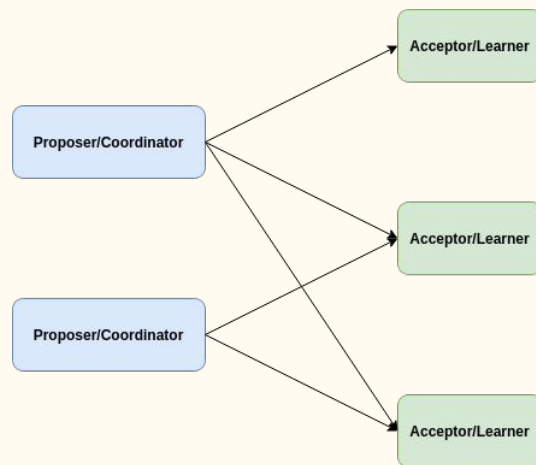
$Message \triangleq P1aMessage \cup P1bMessage \cup P2aMessage \cup P2bMessage$

Figure 1. Normal operation of the Paxos algorithm



# Prepare Phase (P1a)

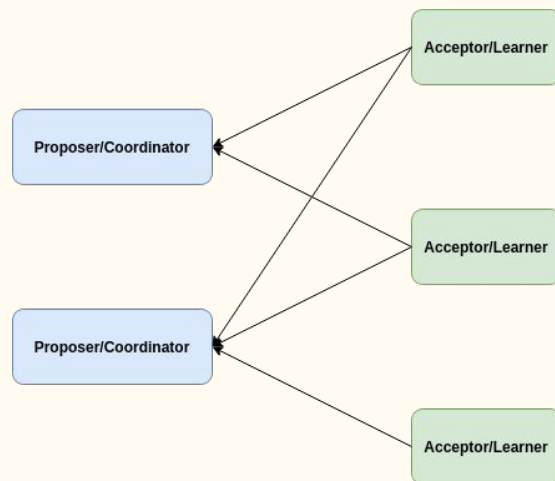
- Proposer *broadcasts* a Prepare (P1a) message.
- Message only contains ballot number.
- Message doesn't contain Proposer. Why?
  - Possible for 2 Proposers to use the same ballot number for Prepare (P1a) message.
  - Impossible for more than 1 Proposer to receive a majority of Promise (P1b) replies.
  - 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.

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


# 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.

$PaxosPromise \triangleq$   
 $\wedge \text{UNCHANGED } \langle decision, maxVBallot, maxValue \rangle$   
 $\wedge \exists a \in Replicas, m \in p1aMessages :$   
 $\wedge maxBallot[a] < m.ballot$   
 $\wedge maxBallot' = [maxBallot \text{ EXCEPT } ![a] = m.ballot]$   
 $\wedge SendMessage([type \mapsto "P1b",$   
 $ballot \mapsto m.ballot,$   
 $acceptor \mapsto a,$   
 $maxVBallot \mapsto maxVBallot[a],$   
 $maxValue \mapsto maxValue[a]])$



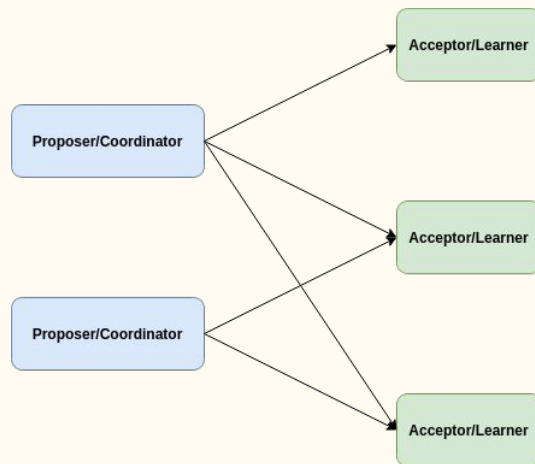
# Accept Phase (P2a)

- 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.

```

ParosAccept  $\triangleq$ 
   $\wedge$  UNCHANGED  $\langle decision, maxBallot, maxVBallot, maxValue \rangle$ 
   $\wedge \exists b \in Ballots, q \in Quorums, v \in Values :$ 
     $\wedge \forall m \in p2aMessages : \neg(m.ballot = b)$ 
     $\wedge$  LET  $M \triangleq \{m \in p1bMessages : m.ballot = b \wedge m.acceptor \in q\}$ 
    IN  $\wedge \forall a \in q : \exists m \in M : m.acceptor = a$ 
       $\wedge \forall m \in M : m.maxValue = none$ 
       $\vee v = ForcedValue(M)$ 
       $\wedge SendMessage([type \mapsto "P2a",$ 
         $ballot \mapsto b,$ 
         $value \mapsto v])$ 

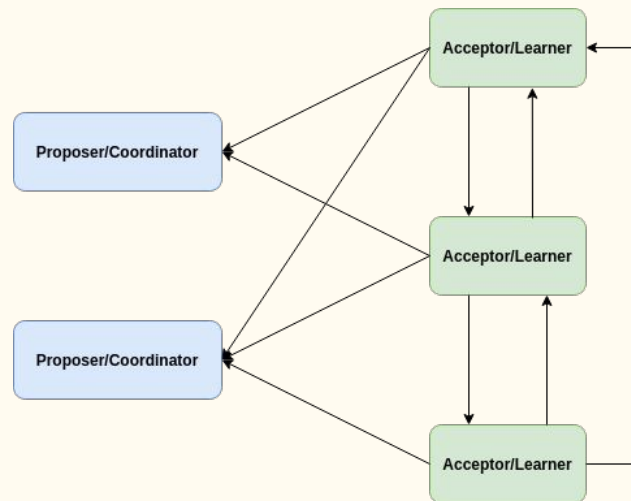
```





# Accepted Phase (P2b)

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

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


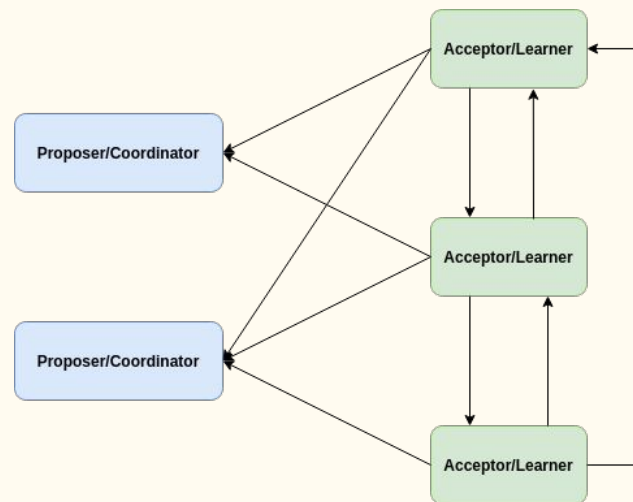
# Decide (End)

- 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.

$$\begin{aligned}
 \text{PaxosNontriviality} &\triangleq \\
 &\wedge \vee \text{decision} = \text{none} \\
 &\quad \vee \exists m \in p2a\text{Messages} : m.\text{value} = \text{decision} \\
 &\wedge \forall m \in p1b\text{Messages} : \wedge m.\text{maxValue} \in \text{Values} \vee 0 = m.\text{maxVBallot} \\
 &\quad \wedge m.\text{maxValue} = \text{none} \vee 0 < m.\text{maxVBallot}
 \end{aligned}$$

- Consistency: At most 1 value can be learnt.

$$\text{PaxosConsistency} \triangleq \text{decision} = \text{none} \vee \text{decision} = \text{decision}'$$

$$\begin{aligned}
 \text{PaxosDecide} &\triangleq \\
 &\wedge \text{UNCHANGED} \langle \text{messages}, \text{maxBallot}, \text{maxVBallot}, \text{maxValue} \rangle \\
 &\wedge \exists b \in \text{Ballots}, q \in \text{Quorums} : \\
 &\quad \text{LET } M \triangleq \{m \in p2b\text{Messages} : m.\text{ballot} = b \wedge m.\text{acceptor} \in q\} \\
 &\quad \text{IN } \wedge \forall a \in q : \exists m \in M : m.\text{acceptor} = a \\
 &\quad \wedge \exists m \in M : \text{decision}' = m.\text{value}
 \end{aligned}$$


# 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.

VARIABLES *messages* Set of all messages sent.

VARIABLES *decision* Decided value of an acceptor.

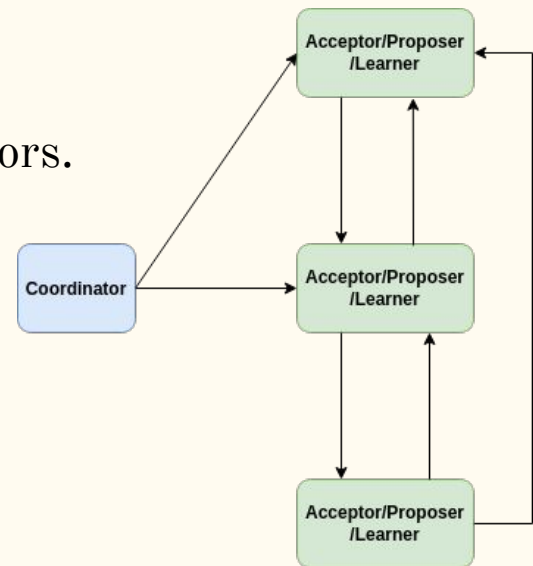
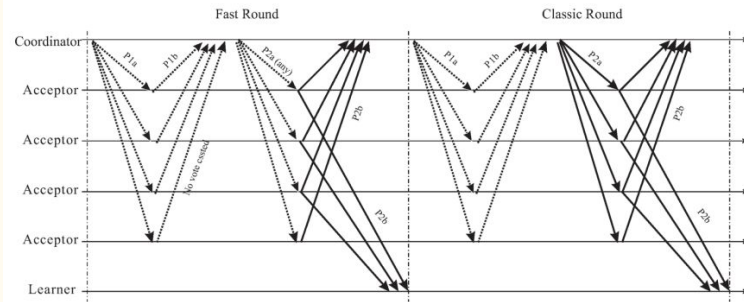
VARIABLES *maxBallot* Maximum ballot an acceptor has seen.

VARIABLES *maxVBallot* Maximum ballot an acceptor has accepted.

VARIABLES *maxValue* Maximum value an acceptor has accepted.

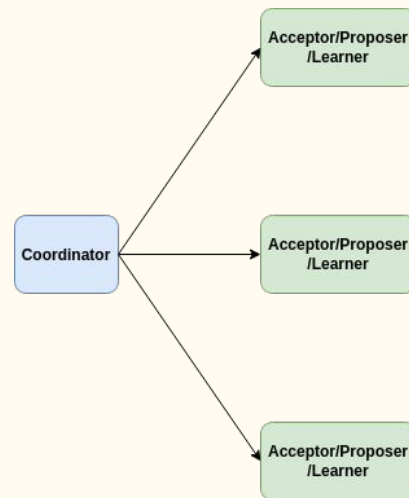
VARIABLES *cValue* Value chosen by coordinator.

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.



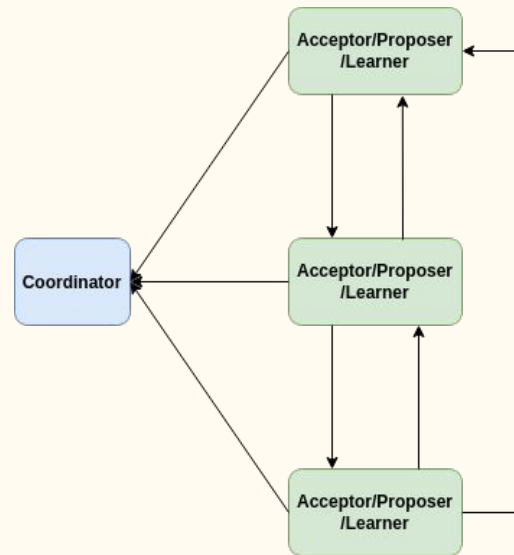
# Fast Any Phase (P2a)

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

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


# Fast Propose Phase (P2b)

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

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


# Fast Decide (No Collision)

- 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.

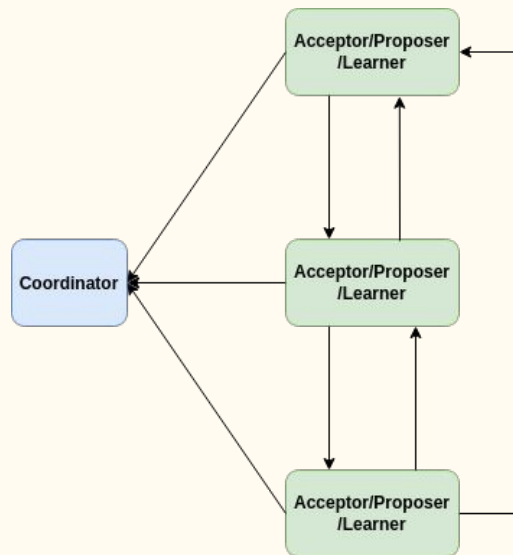
$FastNontriviality \triangleq \vee decision = none$   
 $\vee \exists m \in p2bMessages : m.value = decision \wedge m.ballot \in FastBallots$

- Consistency: At most 1 value can be learnt.

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

$FastDecide \triangleq$

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

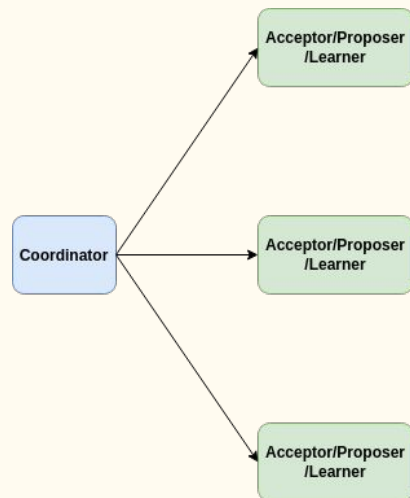


# Classic Accept Phase (P2a)

- 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  $v$ .

```

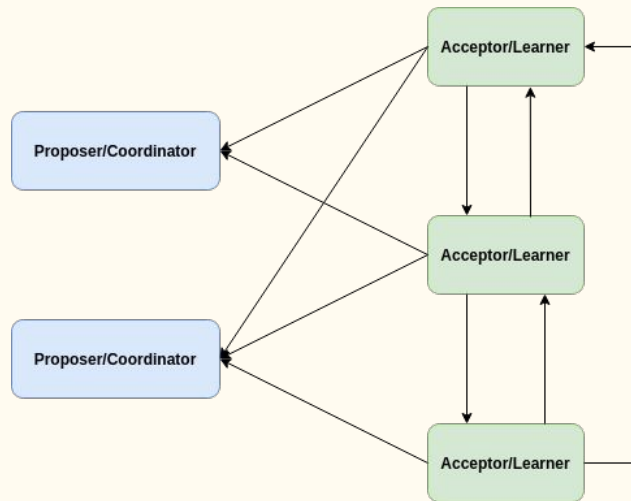
ClassicAccept  $\triangleq$ 
 $\wedge$  UNCHANGED  $\langle decision, maxBallot, maxVBallot, maxValue \rangle$ 
 $\wedge \exists b \in ClassicBallots, f \in FastBallots, q \in FastQuorums, v \in Values :$ 
     $\wedge f < b$  There was a fast round before this classic round.
     $\wedge cValue = none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





# Classic Accepted Phase (P2b)

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

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


# Decide (Collision)

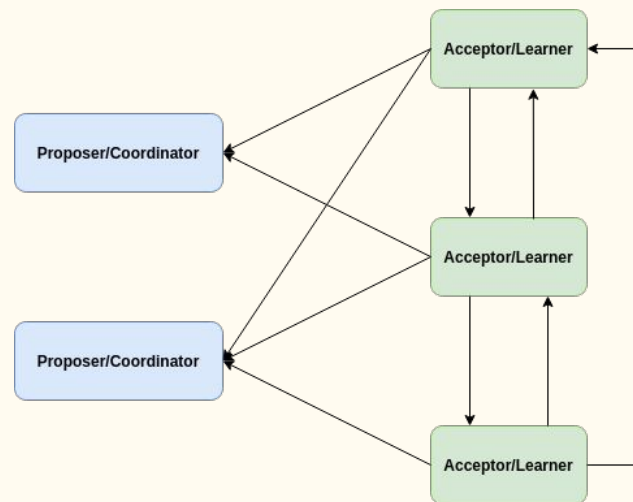
- If an Acceptor receives a quorum of Accepted (P2b) messages for a ballot, it decides on that value. The value is said to be learnt.
- Non-triviality: Only proposed values can be learnt.

$PaxosNontriviality \triangleq$   
 $\wedge \vee decision = none$   
 $\vee \exists m \in p2aMessages : m.value = decision$   
 $\wedge \forall m \in p1bMessages : \wedge m.maxValue \in Values \vee 0 = m.maxVBallot$   
 $\wedge m.maxValue = none \vee 0 < m.maxVBallot$

- Consistency: At most 1 value can be learnt.

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

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



# 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!**