**Paxos algorithm summary**

- **Prepare Phase**
  - **Proposer**
    - Send a **PREPARE** message to **acceptor**
  - **Acceptor**
    - If the **acceptor** has not received a **PREPARE** message with higher number:
      - Send a **PREPARE** message with higher number to **acceptor**
    - Otherwise:
      - Send an **ACCEPT** message to **acceptor**

- **Accept Phase**
  - **Proposer**
    - Wait for responses from a majority of **acceptors**
    - If yes - picks value associated with the highest proposal number from any **acceptor**
    - Otherwise - picks value associated with the highest proposal number from any **acceptor**
  - **Acceptor**
    - If a majority of **acceptors** responded that they agreed on this value, then it is chosen.
    - Otherwise - sends an **ACCEPT** message to **acceptor**

- **Accept Phase**
  - **Proposer**
    - If a majority of **acceptors** responded that they agreed on this value, then it is chosen.
    - Otherwise - sends an **ACCEPT** message to **acceptor**

**Why use proposal numbers?**

- If all requests come from one proposer (leader), then Paxos is trivial
- A leader can fail - Paxos handles the case where multiple proposers might think they are the leader
- Multiple proposers will not lead to inconsistencies
- Each proposer uses a unique proposal number
- Proposals are ordered: newer (higher #) proposals take precedence over older ones
- An acceptor tells it whether it has already accepted a higher numbered proposal
- Why do we need a majority of acceptors?
  - Once a value has been accepted by a majority of acceptors, if any acceptor crashes, at least one acceptor still has the latest (highest) state.

**Questions (Paxos)**

- Why can an acceptor not necessarily accept the first value it receives but must sometimes accept different values?

  [answer from the John Ousterhout video]

  There might not be a majority of proposed values to determine a winner.

  For example:
  
  - 2 acceptors might have value A
  - 3 acceptors might have value B
  - 1 acceptor might have value C

  Therefore, there won’t be one value that all servers can agree on as the majority value.

  An acceptor has the right to change its mind.

  A value that has been accepted does not mean it is ultimately chosen. It just means that it’s the highest numbered proposal at the time it was accepted.

  First check for existing proposed values. Reject old proposals (each proposal has a proposal number) received after newer ones.
Question 1 – Discussion

Why can an acceptor not necessarily accept the first value it receives but must sometimes accept different values?

If each acceptor just accepts a proposed value, it is possible that no acceptors get a majority of any proposed value:
- Acceptors therefore have to be able to accept different values – they may have to change their mind.
- They cannot accept every proposed value because then multiple values could be chosen.
- Once a value has been chosen, a new proposer has to abandon its value and use a previously chosen value.

- We need a 2 phase protocol: phase 1 asks the acceptor for chosen values before proposing a value.
- Any competing proposals have to be aborted.
- This is done by forcing an order: higher numbered (newer) proposals will take precedence over lower-numbered (older) proposals.

Question 2 (Paxos)

When does a proposer have to change the value that it is proposing during the Paxos consensus protocol?

- A proposer sends a value to an acceptor (with a prepare message).
- Multiple proposers may do this concurrently and send different values.
- Acceptors respond to a prepare request from a proposer with the highest numbered proposal that they accepted if another proposal has already been accepted.
- If multiple requests came in concurrently, an acceptor may have seen a higher number. It responds to each proposer with that higher number.
- A proposer must ask for that value to be accepted even if it initially proposed a different value.
- The proposer in the one who figures out the highest accepted proposal from all acceptors and propagates that information to all acceptors.

- This does not violate the requirement of consensus since the algorithm selects one of the proposed values.

Question 3 (Raft)

Raft uses a single leader (one server is elected as a leader). Explain how Raft performs leader election.

Short answer:

Each candidate starts a random timer before proposing itself as a leader & sending election messages to the group.
If you receive a leader proposal message and you have not yet proposed yourself, you will acknowledge that candidate and not vote for yourself.
If a candidate gets majority votes, it becomes the leader.

Question 3 – Longer Answer

Raft uses a single leader (one server is elected as a leader). Explain how Raft performs leader election.

To start an election, a candidate votes for itself and sends a request vote message to all other servers. Other servers that have not yet voted and receive the request acknowledge the candidate to be the leader. Each server that receives a request will vote for at most one candidate.
If the candidate receives a majority of acknowledgements, it becomes the leader.
If the candidate does not win or lose an election, it times out and starts a new election. Randomized timeouts are used to ensure that split votes happen rarely.

To support recovery and avoid stale state, a “term number” is incremented after each election. If the candidate receives a heartbeat from another server and that leader’s term # is at least as large as the candidate’s current term, then the candidate recognizes the leader as legitimate and becomes a follower.

Question 4 (Raft)

An elected leader takes client requests. Each request is essentially a log entry that will be replicated among the servers. When is a log entry committed in Raft?

A log entry is committed once the leader that created the entry has replicated it on a majority of the servers.
Committed means that the log entry is applied to the state machine.

The End