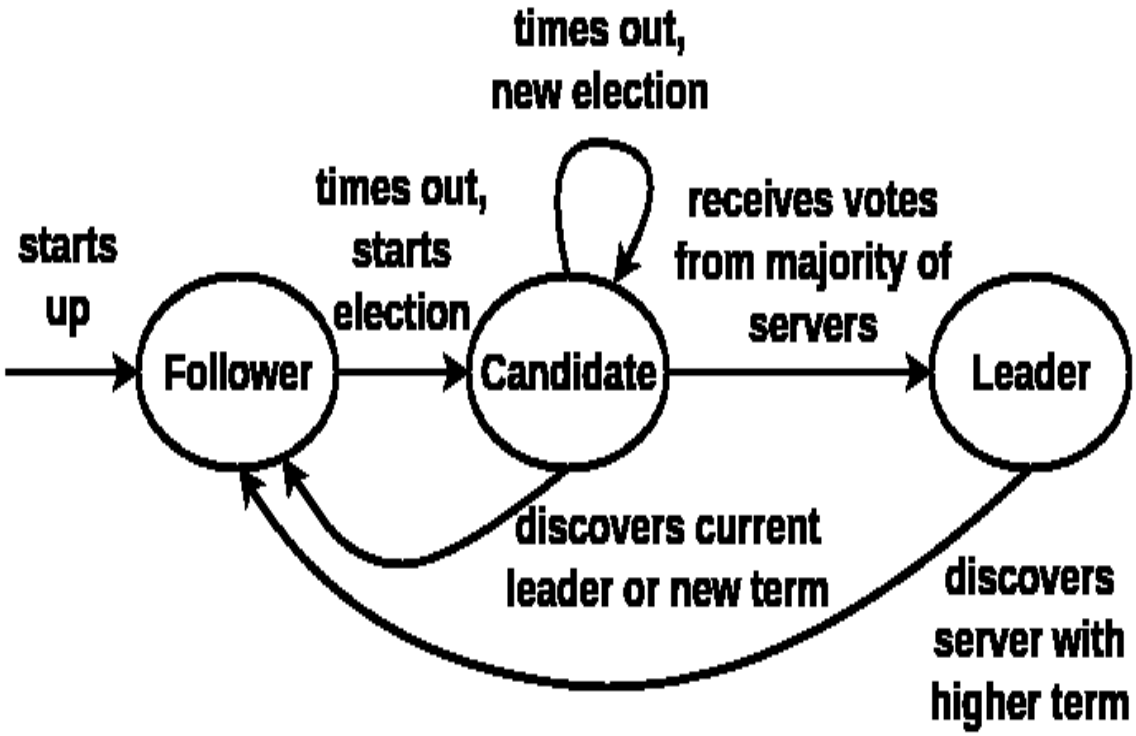
BLOCKCHAIN

Unit3: Class 7

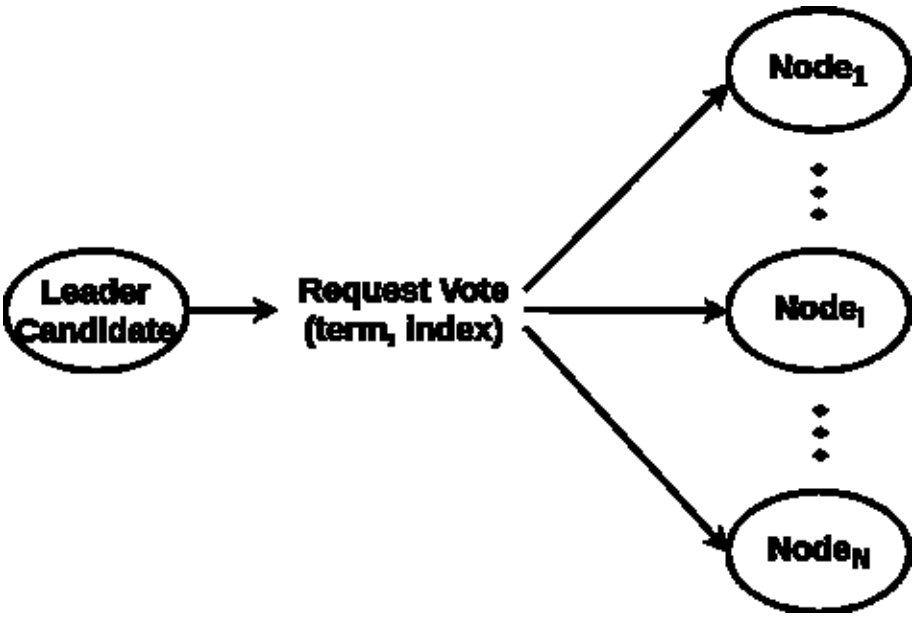
6. Consensus Algorithm – RAFT

- Designed as an alternative to Paxos. Raft is a consensus algorithm for managing a replicated log.
- State machines on a collection of servers compute identical copies of the same state and can continue operating even if some of the servers are down.
- Replicated state machines are used to solve a variety of fault tolerance problems in distributed systems.
- A generic way to distribute a state machine among a set of servers
  - Ensures that every server agrees upon same series of state transitions
- Basic idea:
  - The nodes collectively selects a *leader*; others become *followers*
  - The leader is responsible for state transition log replication across the followers.
- Each node in a replicated state machine(server cluster) can stay in any of the three states, namely, leader, candidate, follower.
- Replicated state machine starts. Initially every node is a follower. Follower times out. Starts an election to get elected as the leader



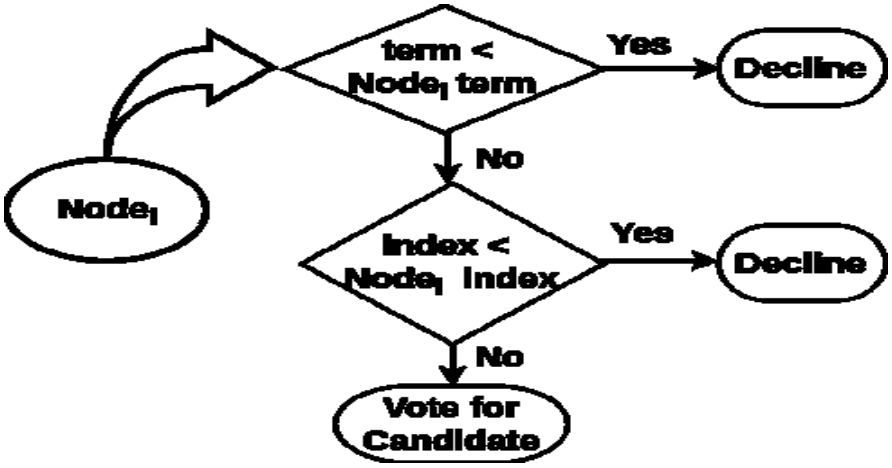
- (re)electing a leader
- committing multiple values to the transaction log
- dealing with replicas failing
- Raft algorithm runs in rounds which is known as term. Each term starts with an election where one or more candidates strive to become a leader. This term number is maintained by every node and is passed while communications between nodes. Every term starts with an election to determine the new leader.
- Initially, we have a set of follower nodes, who look out for a leader. If within certain time interval they do not find one, then the leader election process starts. In this election phase, some of the followers volunteer to become a leader and request votes from all other nodes. Then these candidate nodes send request messages to other followers of the system for vote. Request vote:(term, index), where term: last calculated number known to candidate + 1 and index: committed transaction available to the candidate. This is the first part of Raft algorithm. This Request vote message will be forwarded to all the nodes in the network.

Electing the Leader: Voting Request



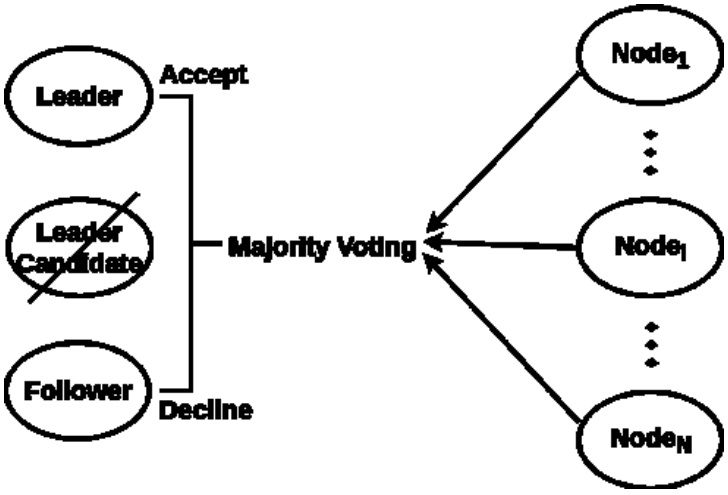
- term: last calculated # known to candidate + 1
- index: committed transaction available to the candidate

Electing the leader: Follower Node's Decision Making



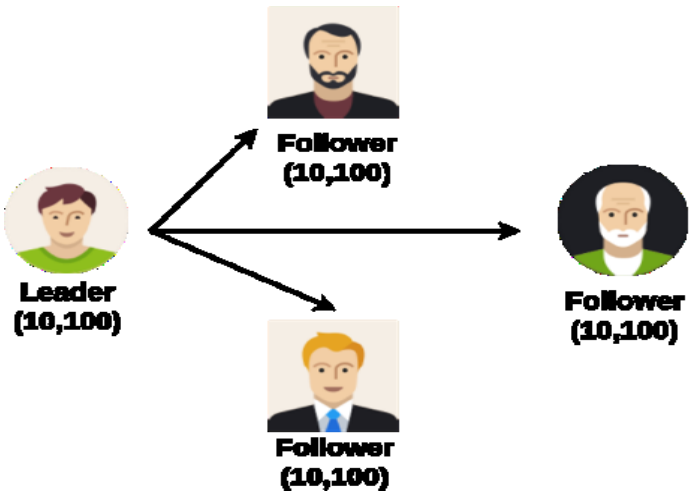
- When a node receives the request, it compares the term and index in the received message with corresponding current known values

Electing the leader: Majority Voting



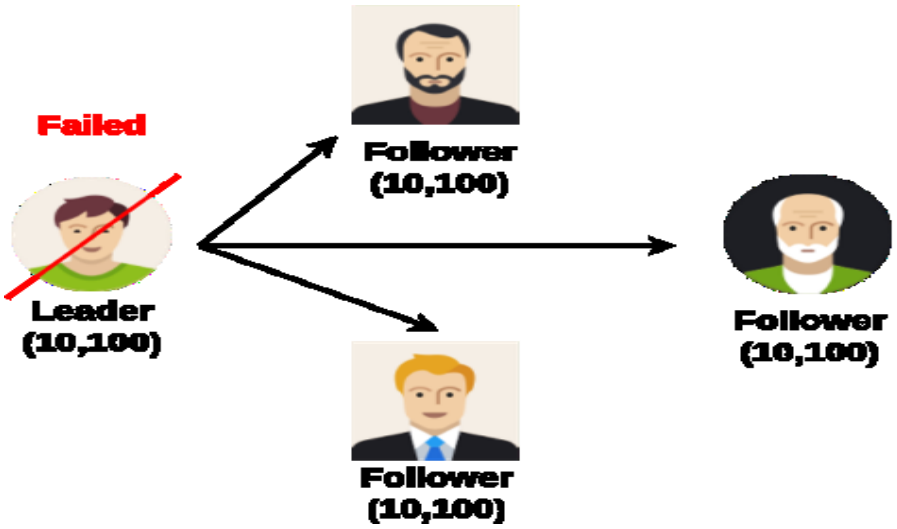
- Use of Majority voting
    - leader selection
    - commit the log entry
    - Like Paxos, the followers vote for one of the candidates and based on the majority of votes, a leader is selected.
    - Then in the next part, the elected leader will propose values which the followers will choose so that the system can reach consensus.

Multiple Leader Candidates: Current Leader Failure



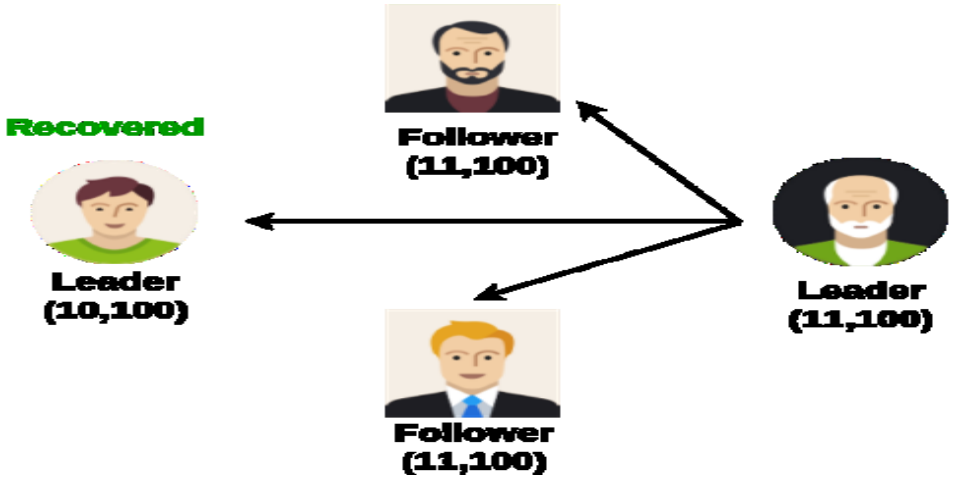
- A leader with three followers
- term: 10

  commit index: 100



The leader node failed

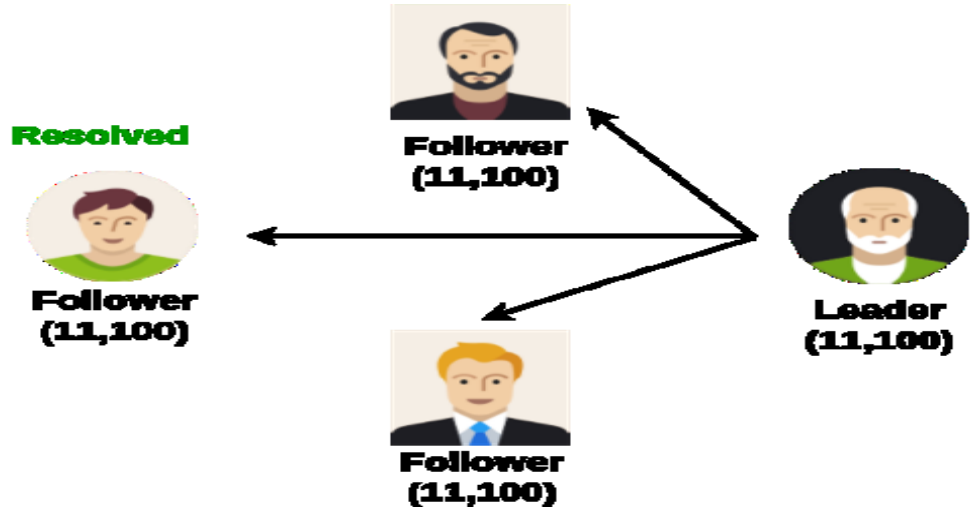
Multiple Leader Candidates: Current Leader Failure



- New leader elected with term 11

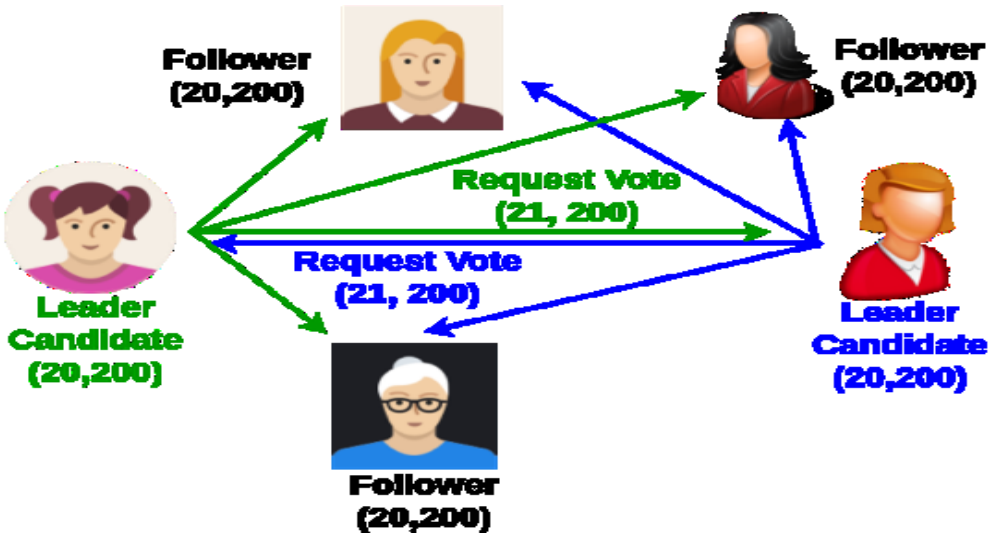
  Old leader recovered

Multiple Leader Candidates: Current Leader Failure

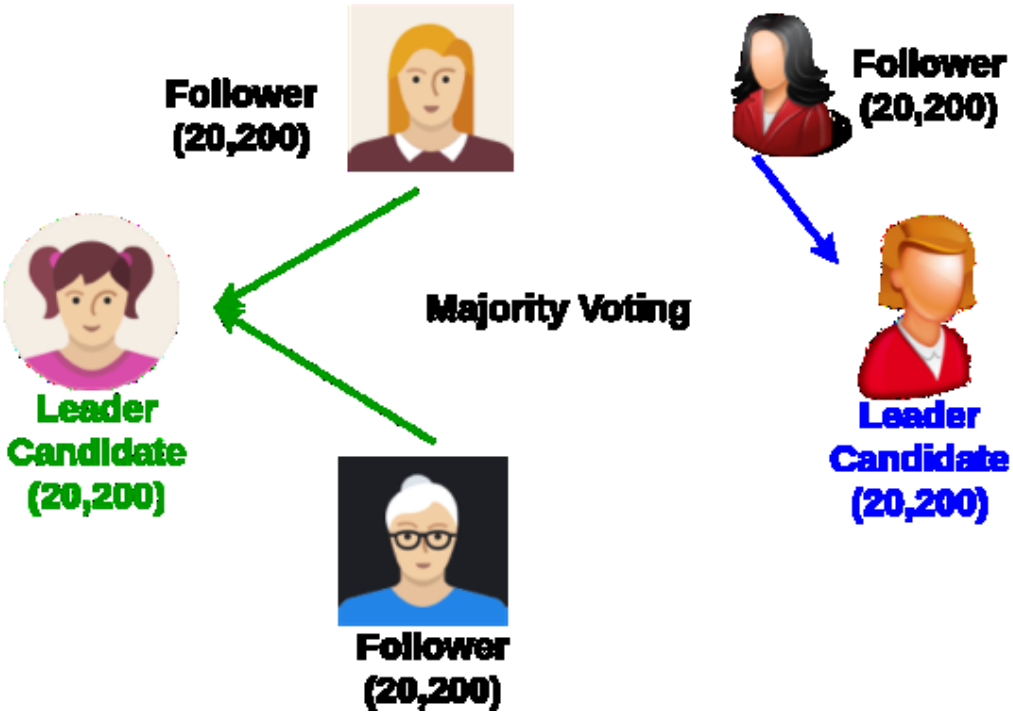


- Old leader receive heartbeat message from new leader with greater term
- Old leader drops to follower state
- The leader sends out heartbeats (signals) to all followers at regular intervals in order to maintain its authority.

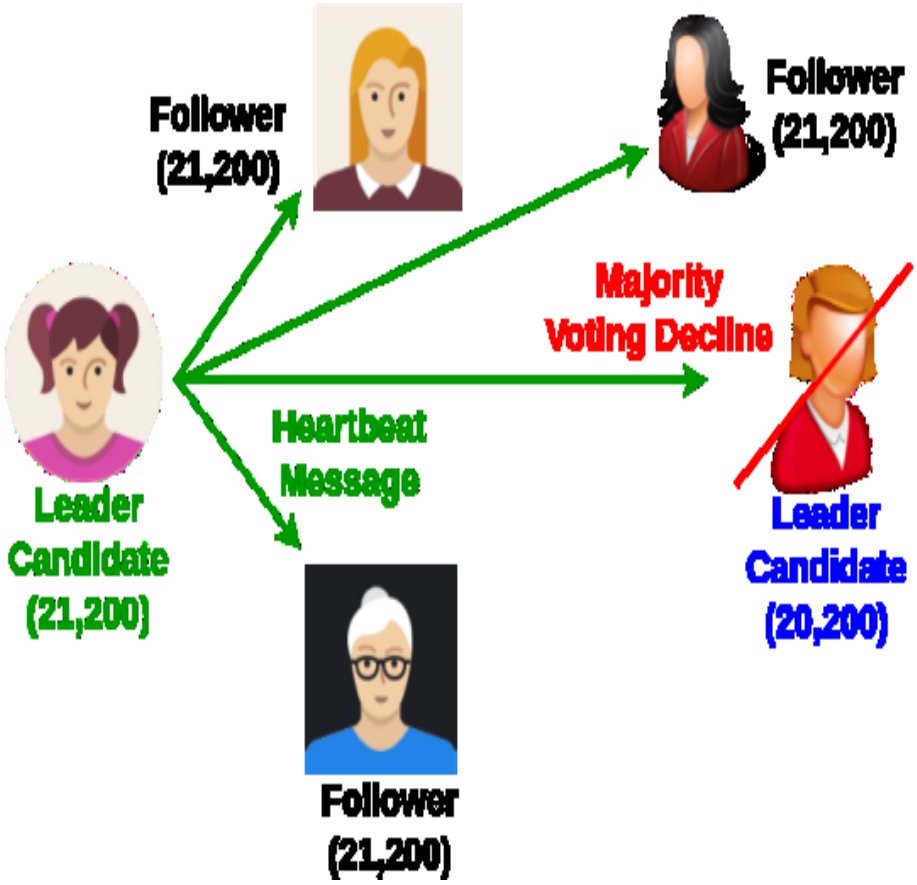
Multiple Leader Candidates: Simultaneous Request Vote



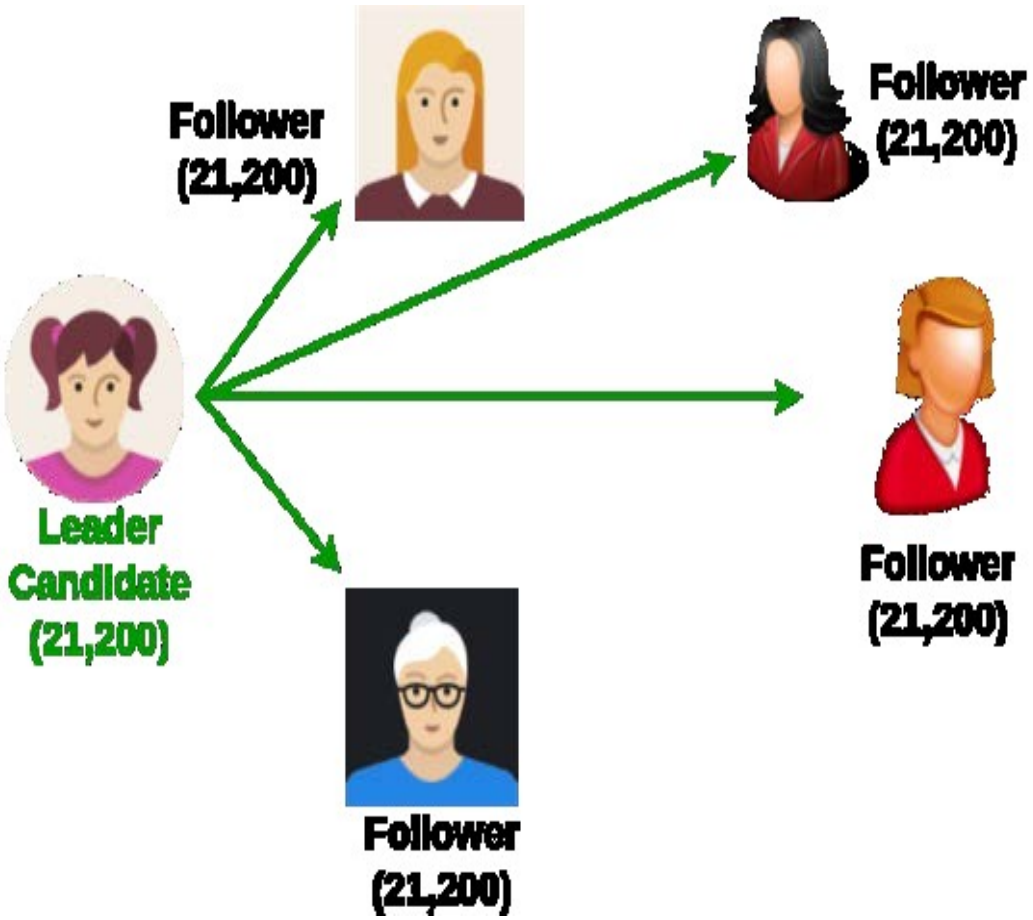
- Two nodes send Request vote message with term 21 at the same time



- One of them gets majority voting

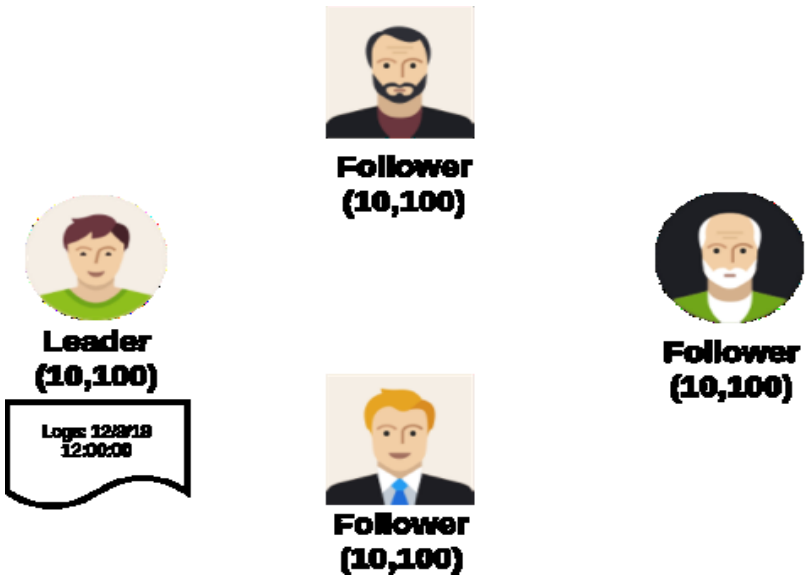


- Winner sends heartbeat message

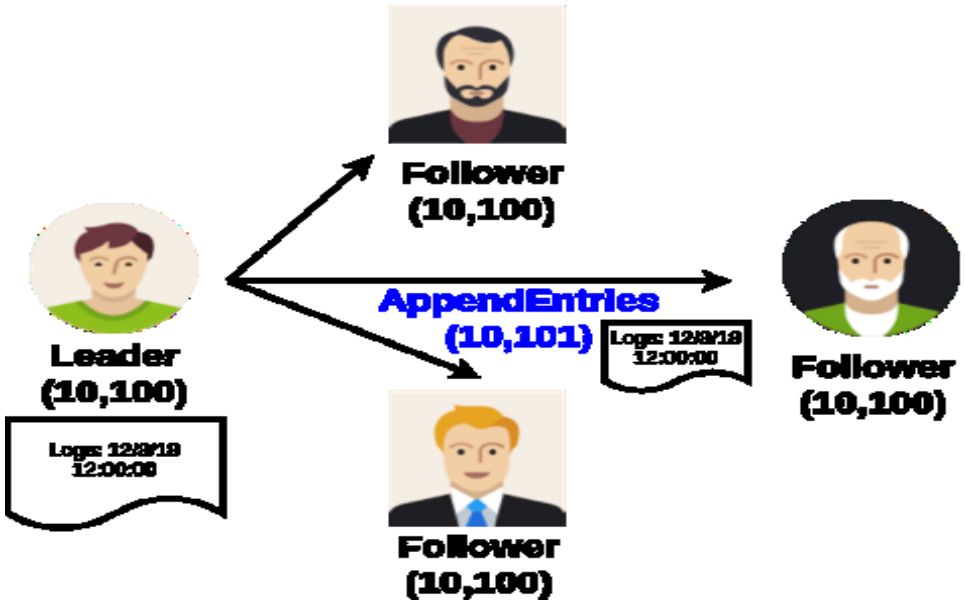


- Other leader candidate switches to follower state

Commiting Entry Log



-
- Leader adds entry to log with term 10 and index 101



- Leader sends *AppendEntries* message to followers with index 101

- Successfully accept entry log

  – All leader and followers update committed index to 101

Handling Failure

- Failure of up to N/2 - 1 nodes does not affect the system due to majority voting.

Uses

- Raft algorithm realizes the same safety performance as Paxos and is better suited in real life implementation and comprehension.
- Raft algorithm cannot support byzantine nodes and can stand up to failure of 50 % of nodes .
- As in case of Permission-ed Blockchain, nodes are verified members.
- Hence, it is more essential to resolve crash faults rather than Byzantine faults for private Blockchain.