BLOCKCHAIN

Unit3: Class 9

Consensus Algorithm – Practical Byzantine Fault Tolerant

Byzantine Falut Tolerance

We believe that Byzantine fault-tolerant algorithms will be increasingly important in the future because malicious attacks and software errors are increasingly common and can cause faulty nodes to exhibit arbitrary behavior. The number of software errors is increasing due to the growth in size and complexity of software. Whereas BFT assumed a synchronous system or were too slow to be used in practical. Practical BFT works in asynchronous environments like the Internet and incorporates several important optimizations that improve the response time of previous algorithms by more than an order of magnitude.

Leslie Lamport proved that if we have 3m+1 correctly working processors, a consensus(agreement on same state) can be reached if atmost m processors are faulty which means that strictly more than two-thirds of the total number of processors should be honest.

How PBFT works ?

PBFT tries to provide a practical Byzantine state machine replication that can work even when malicious nodes are operating in the system.

The algorithm is designed to work in asynchronous systems and is optimized to be high-performance with an impressive overhead runtime and only a slight increase in latency.

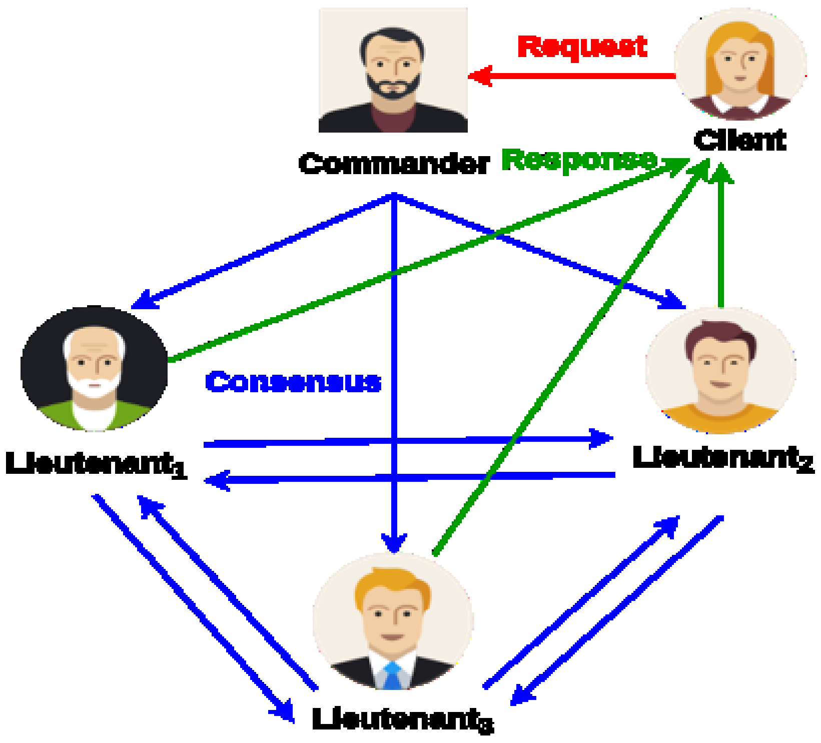
- Why Practical?
  - Ensures safety over an asynchronous network (not liveness!)- safety properties informally require that "something bad will never happen" in a distributed system. Safety is concerned with a program not reaching a bad state and that liveness is concerned with a program eventually reaching a good state. This means that clients eventually receive replies to their requests and those replies are correct according to linearizability

- Byzantine Failure
- Low overhead

- Real Applications
  - Tendermint
  - IBM's Openchain
  - ErisDB
  - Hyperledger

The number of nodes increase, the system becomes more secure.

Practical Byzantine Fault Tolerant Model



- Asynchronous distributed system
  - delay, out of order message
- Byzantine failure handling
  - arbitrary node behavior

- Supports Privacy
  - Ensures messages are tamper-proof message, applies hashing techniques to provide authentication
- A state machine is replicated across different nodes. The service is modeled as a state machine that is replicated across different nodes in a distributed system. Each state machine replica maintains the service state and implements the service operations.
- $3f + 1$ replicas are there where f is the number of faulty replicas. Although there could be more than 3f+ 1 replicas, the additional replicas degrade performance.
- The replicas move through a successions of configurations, known as

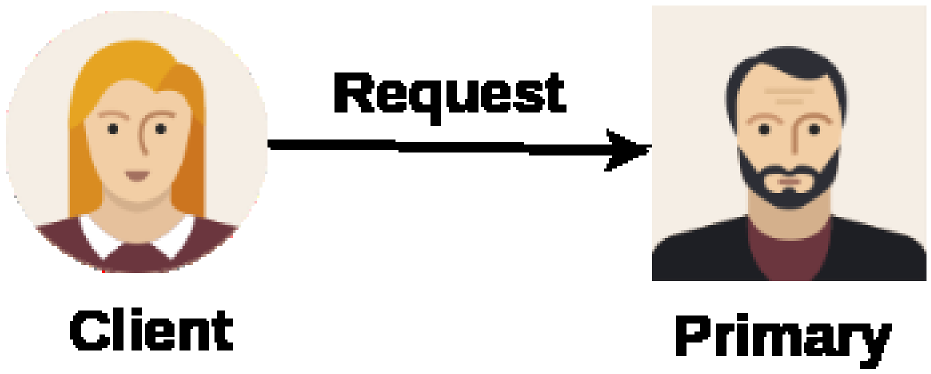
  views
- One replica in a *view* is *primary* and others are *backups*. Setup of primary and a backup we consider as single view. Views are numbered consecutively. Nodes in a PBFT enabled distributed system are sequentially ordered with one node being the primary(or the leader node) and others referred to as secondary(or the backup nodes).

  Any eligible node in the system can become the primary by transitioning from secondary to primary(typically, in the case of a primary node failure). The goal is that all honest nodes help in reaching a consensus regarding the state of the system using the majority rule.
- A practical Byzantine Fault Tolerant system can function on the condition that the maximum number of malicious nodes must not be greater than or equal to one-third of all the nodes in the system.
- As Views are changed when a *primary* is detected as faulty
- Every view is identified by a unique integer number v
- Only the messages from the current views are accepted

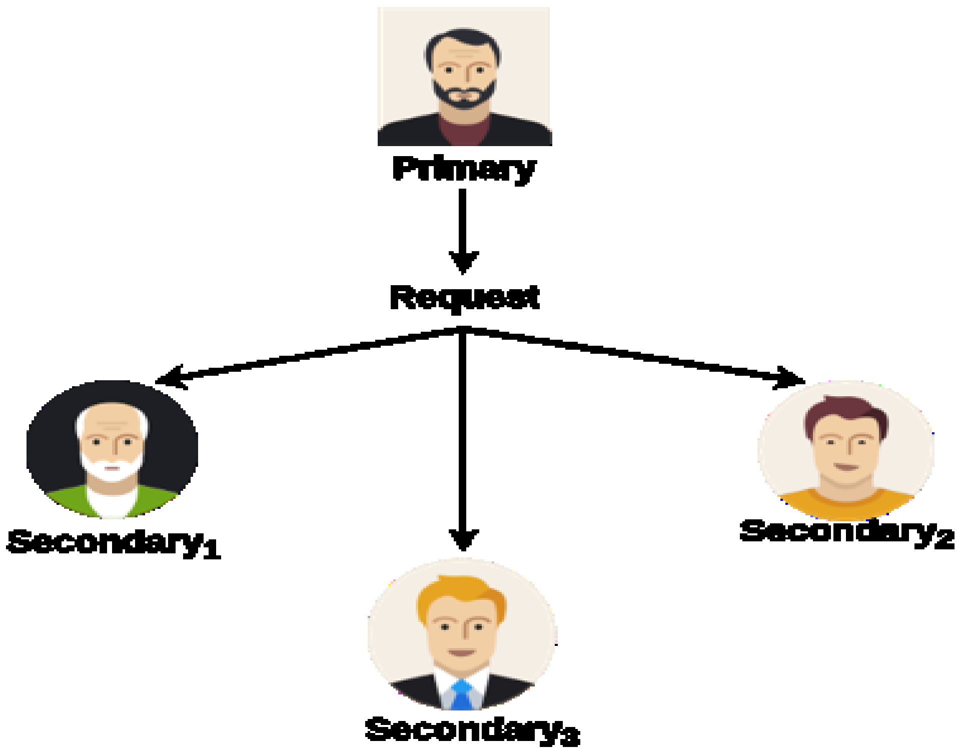
PBFT consensus rounds are broken into 4 phases(refer with the image below):

- The client sends a request to the primary(leader) node.
- The primary(leader) node broadcasts the request to the all the secondary(backup) nodes.

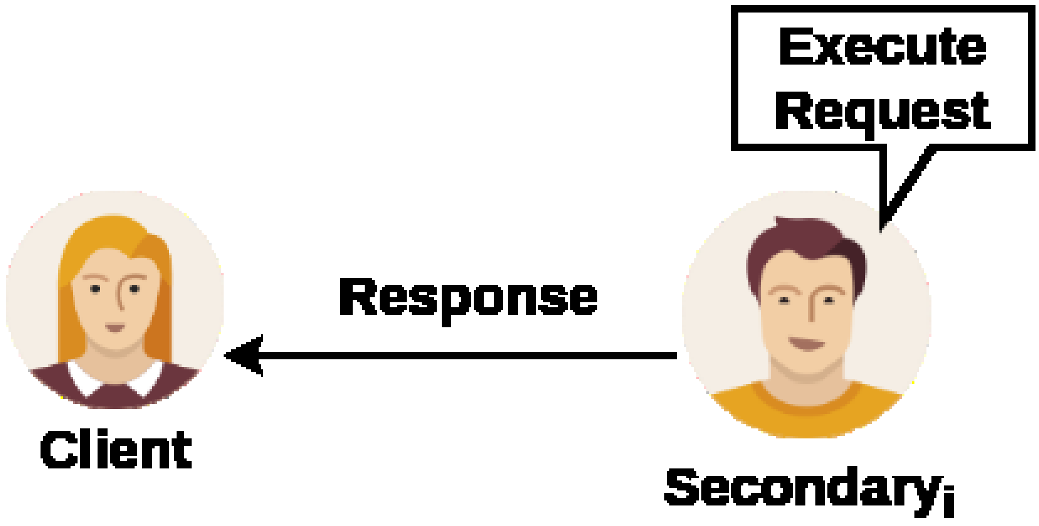
- The nodes(primary and secondaries) perform the service requested and then send back a reply to the client.
- The request is served successfully when the client receives 'm+1' replies from different nodes in the network with the same result, where m is the maximum number of faulty nodes allowed.
- The primary(leader) node is changed during every view(pBFT consensus rounds) and can be substituted by a view change protocol if a predefined quantity of time has passed without the leading node broadcasting a request to the backups(secondary). If needed, a majority of the honest nodes can vote on the legitimacy of the current leading node and replace it with the next leading node in line.



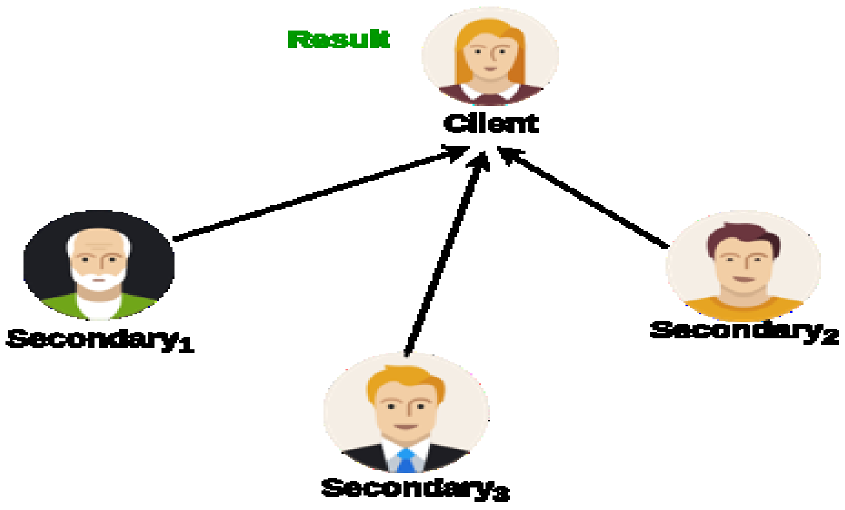
- A client sends a request to invoke a service operation to the primary



- The primary multicasts the request to the backups.



- Backups execute the request and send a reply to the client



- The client waits for $f + 1$ replies from different backups with the same result.
  - f is the maximum number of faulty replicas that can be tolerated.

It is a three phase protocol

- Pre-prepare: primary proposes an order
- Prepare: Backup copies agree on #
- Commit: agree to commit

Three phase commit protocol- Pre-Prepare

Primary assigns a sequence number n to the request and multicast a message

$\langle\langle \text{PRE-PREPARE}, v, n, d\rangle_{\sigma_p}, m\rangle$ to all the backups

- v is the current view number
- n is the message sequence number
- d is the message digest
- σ_p is the private key of primary- works as a digital signature
- m is the message to transmit

Three phase commit protocol- Pre-prepare

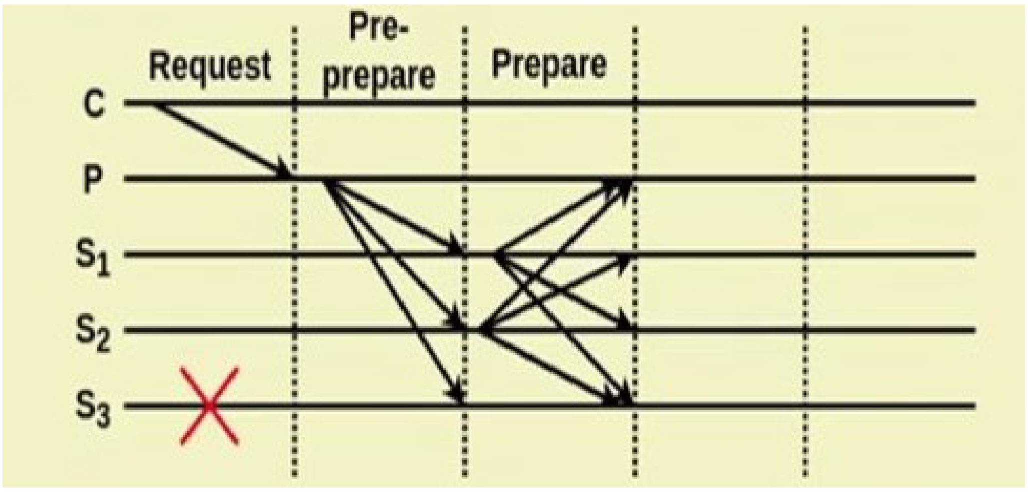
- Pre-Prepare messages are used as a proof that request was assigned with sequence number n in the view v

A Back up accepts a pre-prepare message if

- The signature is correct and d is the digest for m
- The backup is in view v
- It has not received a different PRE-PREPARE message for view v and sequence number n containing a different digest.
- The sequence number is within a threshold.

Three phase commit protocol- Prepare

- If the backup accepts the PRE-PREPARE message, it enters prepare phase by multicasting a message $< PREPARE, v, n, d, i >_{\sigma\, i}$ to all other replicas.
- A replica (both primary and backups) accepts prepare messages if
  - signature are correct
  - View number equals to the current view
  - Sequence number is within a threshold



Prepare

- Replicas agree on assigned sequence number

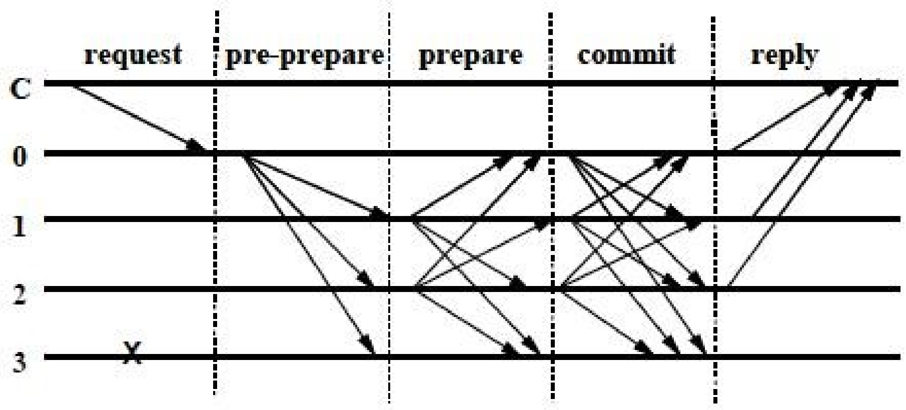


Figure 1: Normal Case Operation

- Pre-Prepare and prepare ensure that non-faulty replicas guarantee on a total order for requests within a view.

Three phase commit protocol-Commit

- Multicast $< COMMIT, v, n, d, i >_{\sigma\_i}$ message to all the replicas including primary.
- Replicas accept commit messages and insert them in their log provided
  - They are properly signed.
  - The view number in the message is equal to the replica’s current view.
  - The sequence number is within the threshold.
- Commit a message if
  - 2f prepare from different backups matches with the corresponding pre-prepare
  - You have total 2f+1 votes (one from primary) from non-faulty replicas.
- Commit a message when a replica
  - Has sent a commit message itself
  - Has received 2f+1 commits

Advantages

- Ability to provide transaction finality without the need for confirmations like in Proof-of-Work models such as the one Bitcoin employs.
- If a proposed block is agreed upon by the nodes in a pBFT system, then that block is final. This is enabled by the fact that all honest nodes are agreeing on the state of the system at that specific time as a result of their communication with each other.
- Another important advantage of the pBFT model compared to PoW systems is its significant reduction in energy usage.

Limitations of pBFT:

The pBFT consensus model works efficiently only when the number of nodes in the distributed network is small due to the high communication overhead that increases exponentially with every extra node in the network.

- Sybil attacks : The pBFT mechanisms are susceptible to Sybil attacks, where one entity(party) controls many identities. As the number of nodes in the network increase, sybil attacks become increasingly difficult to carry out. But as pBFT mechanisms have scalability issues too, the pBFT mechanism is used in combination with other mechanism(s).
- Scaling : pBFT does not scale well because of its communication(with all the other nodes at every step) overhead. As the number of nodes in the network increase(increases as O(n^k), where n is the messages and k is the number of nodes), so does the time taken to respond to the request.