Blockchain DLOC Sem VII

CSDC7022 : Block Chain

Module - 5 : Private Blockchain (8 Hours)

Instructors : Tarun Shetty, Sujal Patil, Om Borate, Divesh Mangtani

Department of Computer Engineering, VESIT, Mumbai

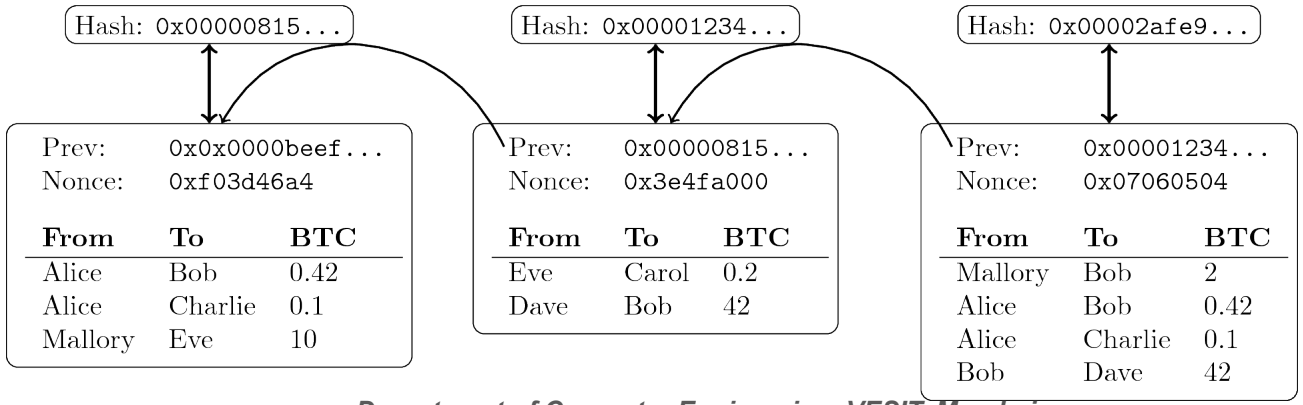
Topics to be covered

- Private blockchain
  - Introduction
  - Key Concepts
  - Need of Private blockchain
  - Smart Contracts in Private Environment
  - State Machine Replication
  - Consensus Algorithm : PAXOS and RAFT
  - Byzantine Fault
- Introduction to Hyperledger
  - Tools and Frameworks
  - Hyperledger Fabric
  - Comparison between hyperledger fabric and other technologies

Private blockchain/ Permissioned blockchain / Closed blockchain

Looking Back at Public Blockchains

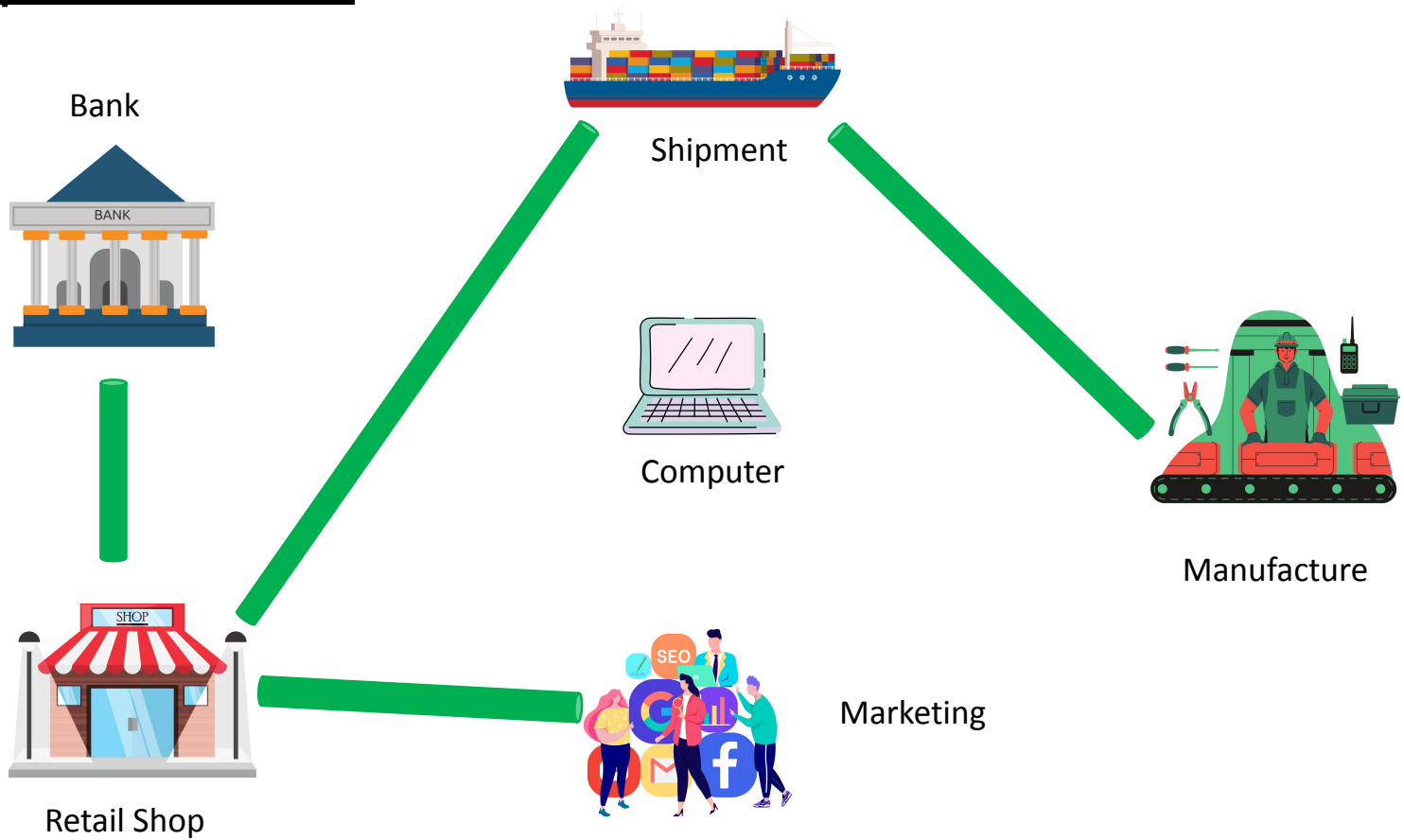
A block is a fundamental component of a blockchain, which is a distributed and decentralized digital ledger used to record transactions and other data in a secure and immutable manner. Each block contains a collection of transactions, along with metadata and a reference to the previous block, forming a chronological chain of blocks, hence the name "blockchain."



Public Blockchain vs Private Blockchain

Properties	Public Blockchain	Private Blockchain
Decentralize Ledger	✓	✓
Cryptocurrency / tokens	✓	✗
Immutable	✓	✓
Anonymous	✓	✗
All node verification	✓	✗
Smart Contract	✓	✓
Transparent	✓	✗

Example usecase:



Private blockchain over public :

Controlled Access: Private blockchains are typically restricted to a defined group of known participants, ensuring that only authorized entities can join and participate in the network. This level of control is beneficial in situations where data privacy and regulatory compliance are paramount.

Enhanced Privacy: Private blockchains provide greater privacy for sensitive transactions and data. Participants can keep certain details of transactions confidential, while still benefiting from the transparency and immutability features of blockchain technology.

Efficiency and Performance: Private blockchains often exhibit higher transaction throughput and lower latency compared to public blockchains. This is because the network's limited size allows for faster consensus mechanisms, and the reduced number of participants simplifies communication and verification.

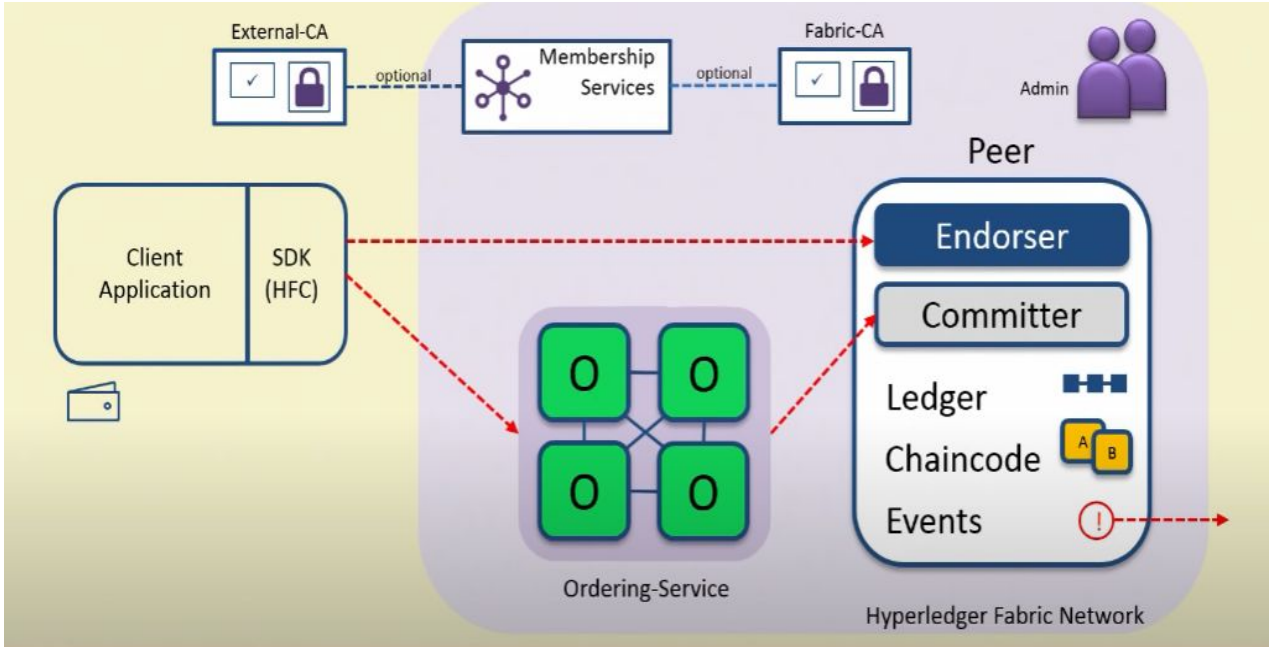
Tailored Governance: Private blockchains enable participants to define and implement governance rules that suit their specific needs. Decisions regarding consensus mechanisms, upgrades, and changes can be made more easily and with consensus among a smaller group.

Private blockchain over public:

Scalability: Private blockchains can be designed with scalability in mind for specific use cases, allowing organizations to adapt the blockchain to their current and anticipated transaction volumes without being constrained by the limitations of a public network.

Cost-Efficiency: Operating a private blockchain can be more cost-effective since it doesn't require the extensive computational resources needed for public blockchain consensus mechanisms like Proof of Work. Private blockchains also have lower energy consumption.

Private Blockchain Architecture in context to hyperledger

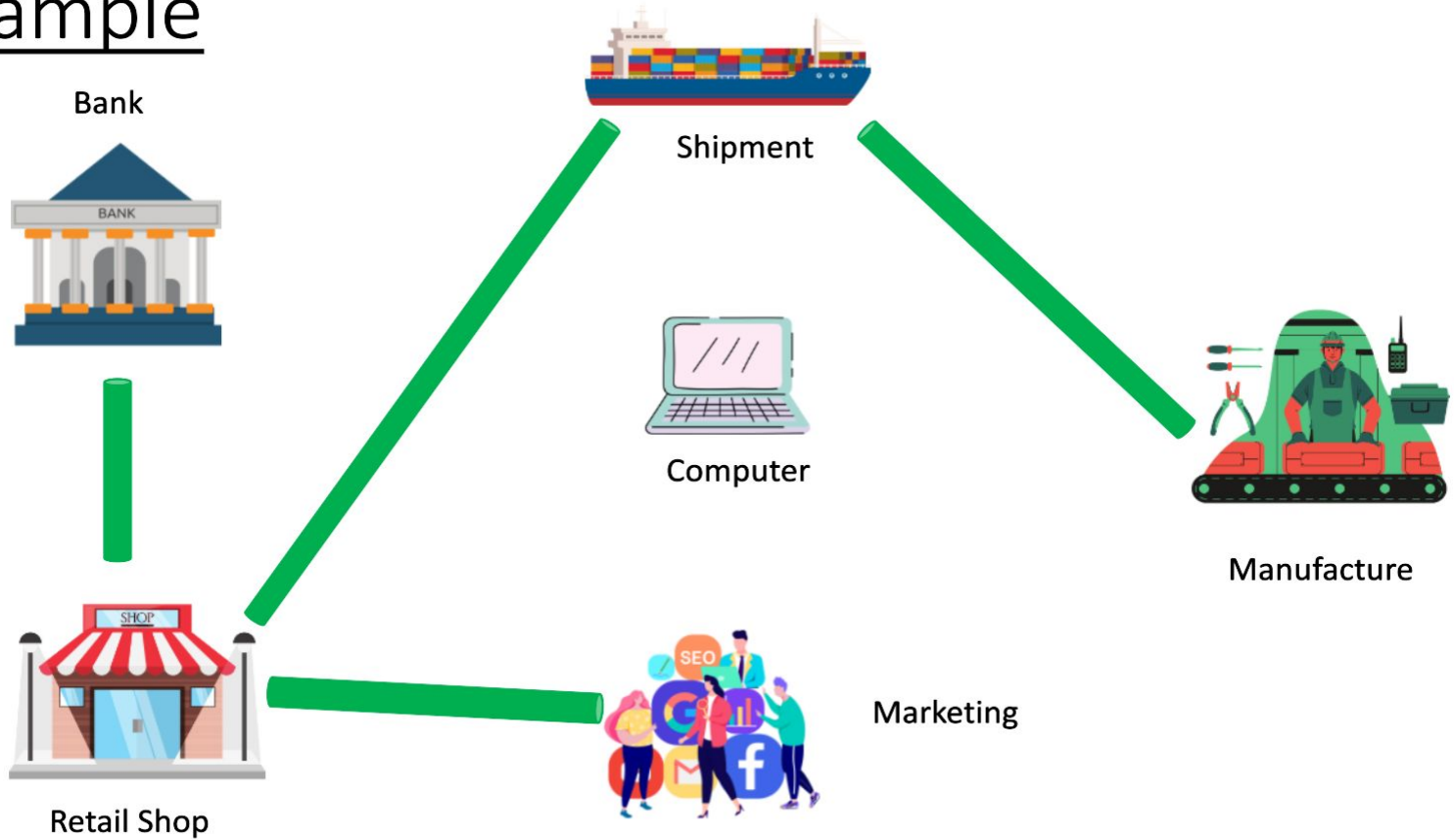


Characteristics of Permissioned Model

- A blockchain architecture where users are authenticated a priori
  - A membership service provider(msp) helps to obtain the access to the chain

- Users know each other
  - However, users may not trust each other
  - Security and consensus are still required.

Example


Bank
BANK
Shipment
Computer
Manufacture
SHOP
SEO
Marketing
Retail Shop

Smart Contracts over closed network

Smart Contract : “A self executing contract in which the terms of the agreement are written in the code which justify what actions are to be performed when a certain condition is met.”

Agreement on a Smart Contract Execution

- Store the contract on a blockchain.
- Once an event is triggered, execute the codes **locally on each peer.**
- Generate transactions as the output of the contract execution.
- The peers of the blockchain network validates the transaction, and the output is committed in the blockchain - may trigger the next event to execute the code further.

“Do we need to execute the codes locally on each peer?”

Why?

The Solution:

- Execute contract at a subset of nodes, and ensure that the same state is propagated to all the nodes
  - Majority of the peers should agree on the state
  - Validation: Generate a **“proof”** that a peer has agreed on the **state of execution**
- **Proof :** Proof of Authority(PoA), Proof of Stake(PoS), etc.

Smart Contract as State Machine replication

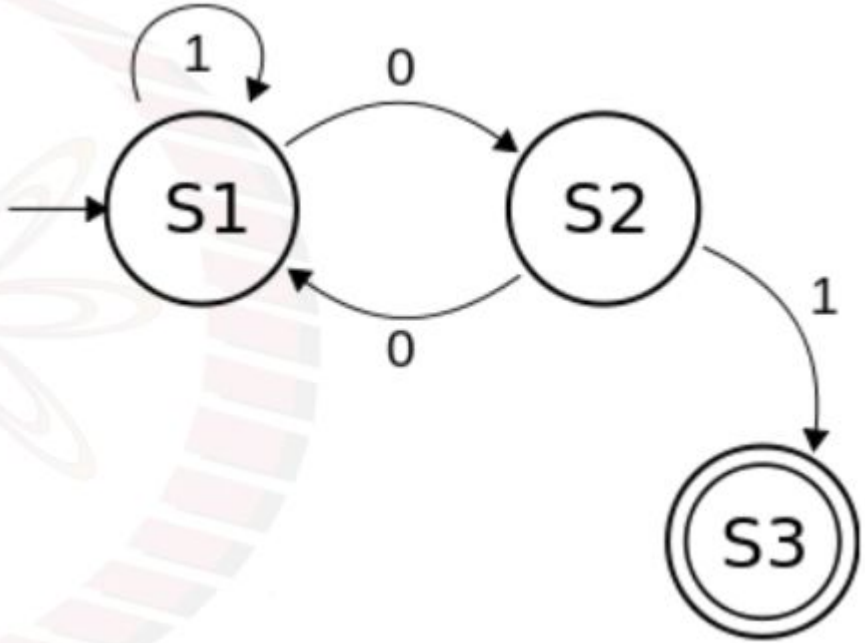
STATE MACHINE REPLICATION

Represent the smart contract as a state machine

- Remember, any deterministically executable code can be represented as a state machine.

Example of smart contract as state machine

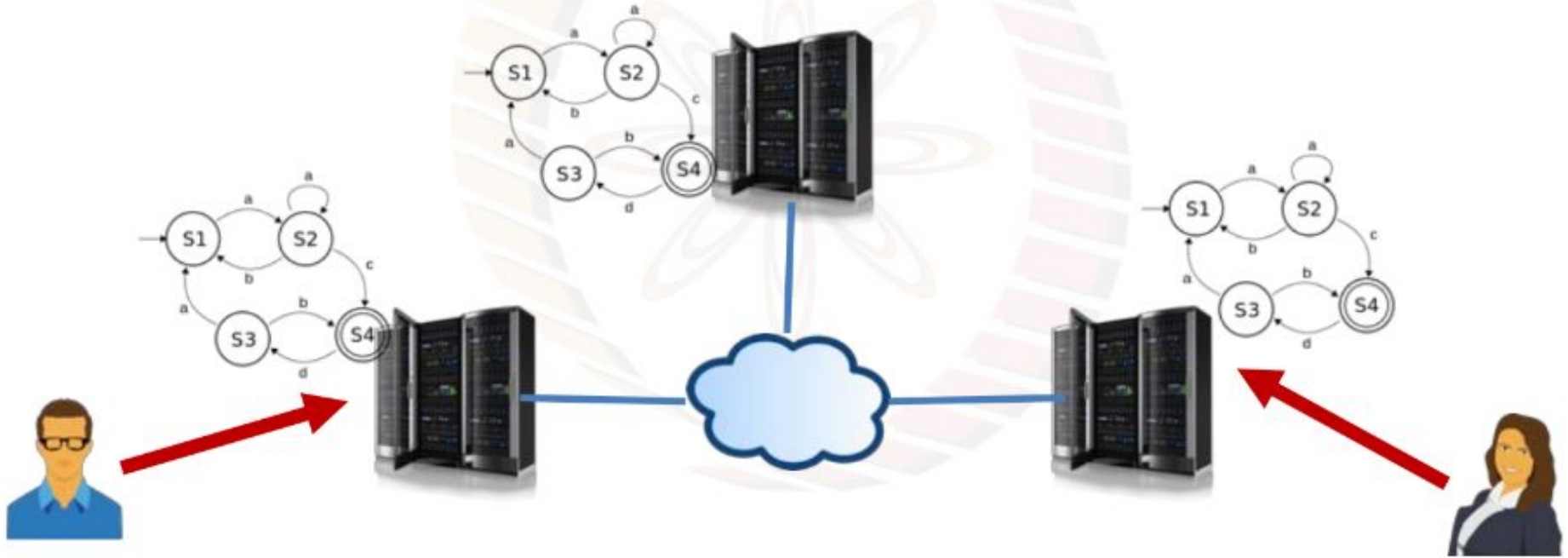
```
S1:
While (moreGoods == 1)
        DeliverGoods();
S2:
If (allOrderComplete == 0) goto S1;
Else {
        S3:
        printf("Goods transfer complete"):
}
```



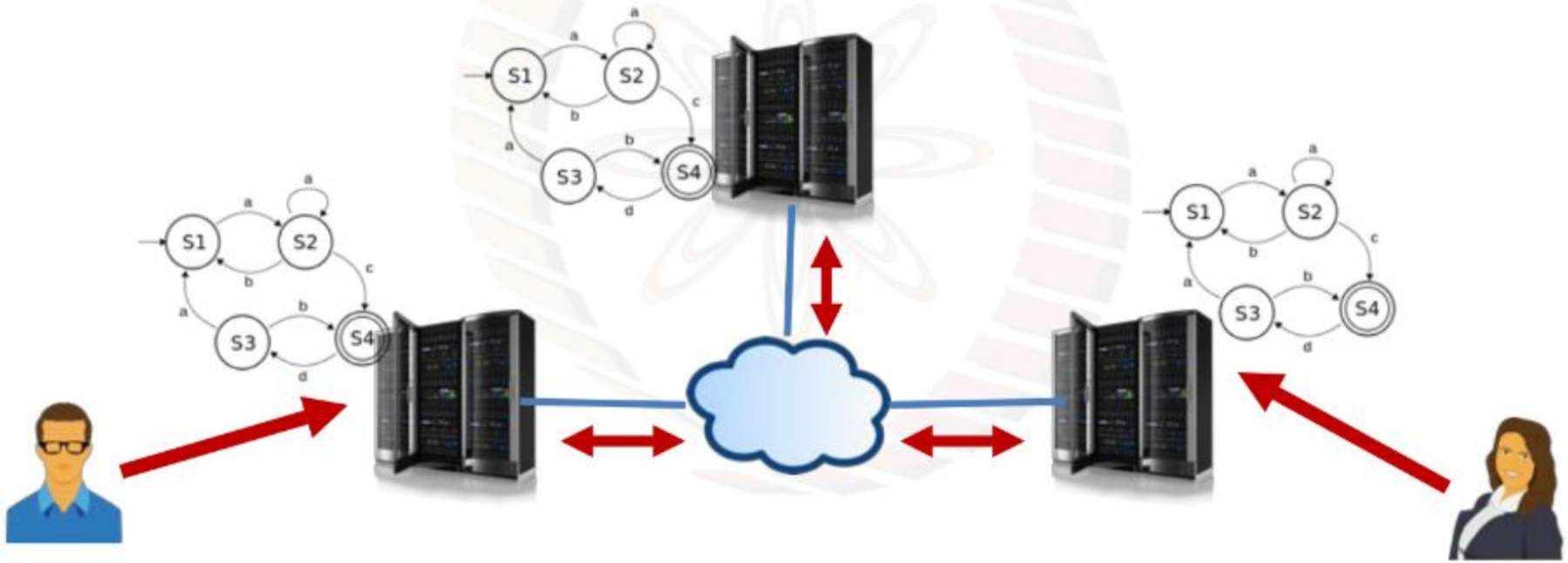
State Machine replication

1. Place copies of the state machine on multiple independent servers

2. Receive client requests, as an input to the state machine



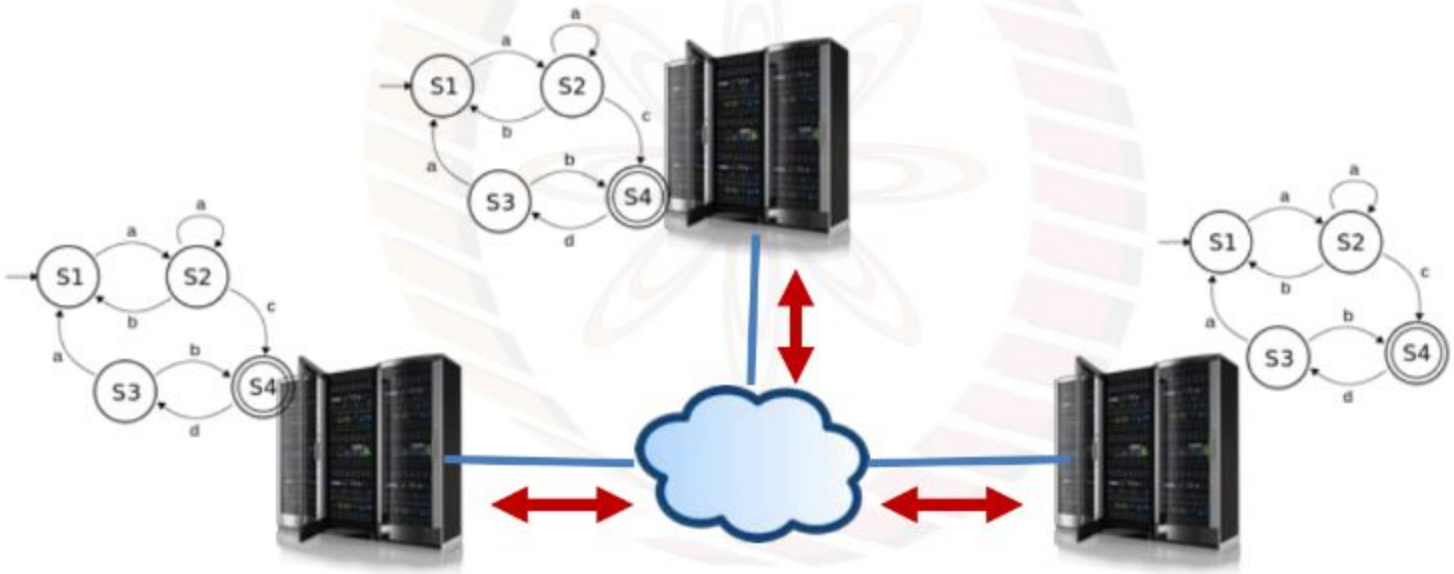
3. Propagate the inputs to all the servers



4. Order the inputs based on some ordering algorithm

4. Execute the inputs based on the order decided, individually at each server

5. Sync the state machines across the servers, to avoid any failure.



6. If output state is produced, inform the clients about the output

Consensus

- A procedure to reach in a common agreement in a distributed or decentralized multi-agent platform

Why do we need it?

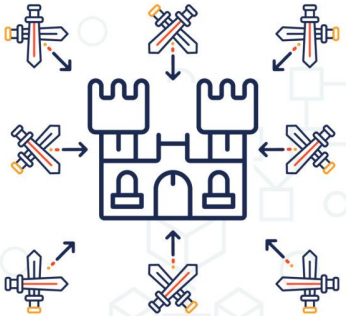
- To create Reliability and Fault tolerance in a distributed environment.
- To ensure correct operation in the presence of the faulty individuals.

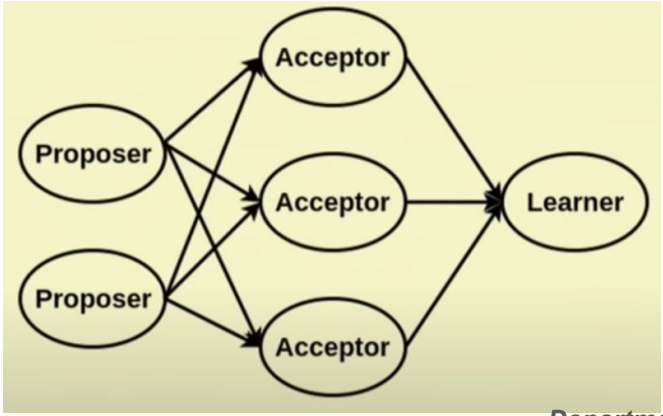
Some common faults in the distributed systems

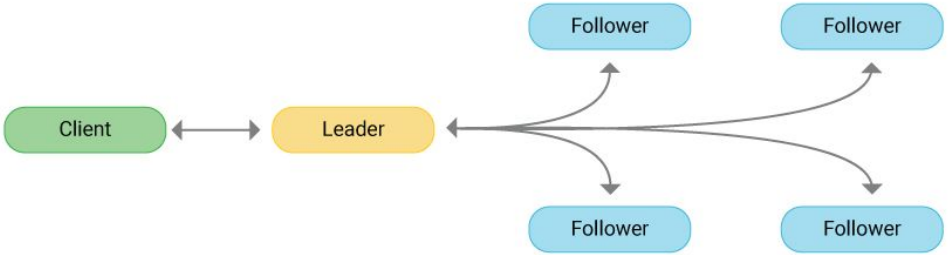
- **Crash fault:** A node suddenly crashes or becomes unavailable in the middle of a communication
- **Network or Partitioned Faults**: A network fault occurs (say the link failure) and the network gets partitioned
- **Byzantine Faults**: A node starts behaving maliciously

Consensus Algorithms for Private Blockchain

- Byzantine fault tolerance (Malicious activity)
- Paxos (Crash or Network fault)
- Raft (Crash or Network fault)







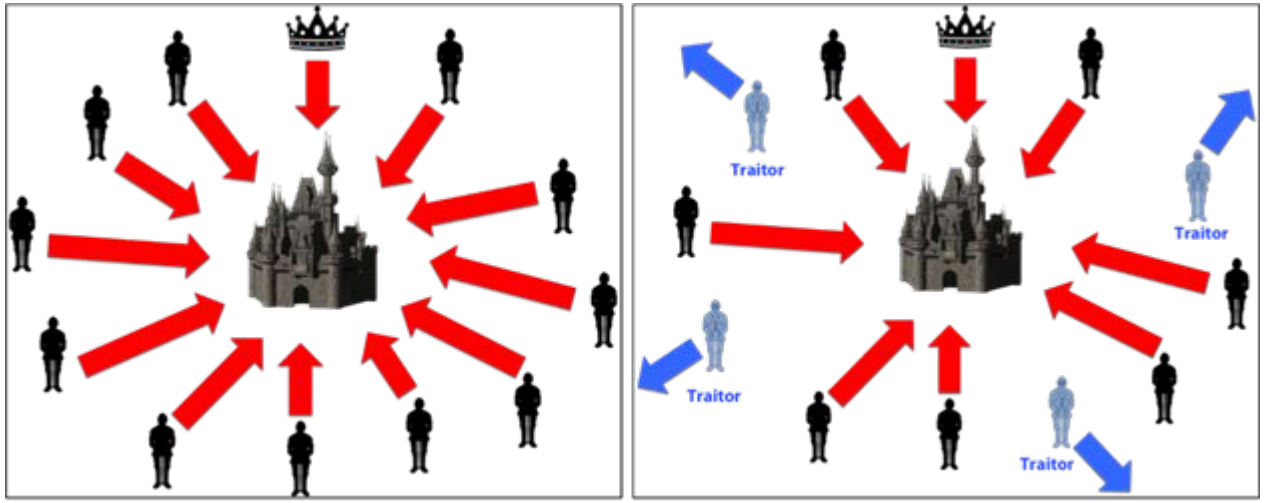
Byzantine Fault Tolerance (BFT)

Byzantine Fault Tolerance (BFT) is a consensus approach that resists a system to get into the Byzantine Generals' problem

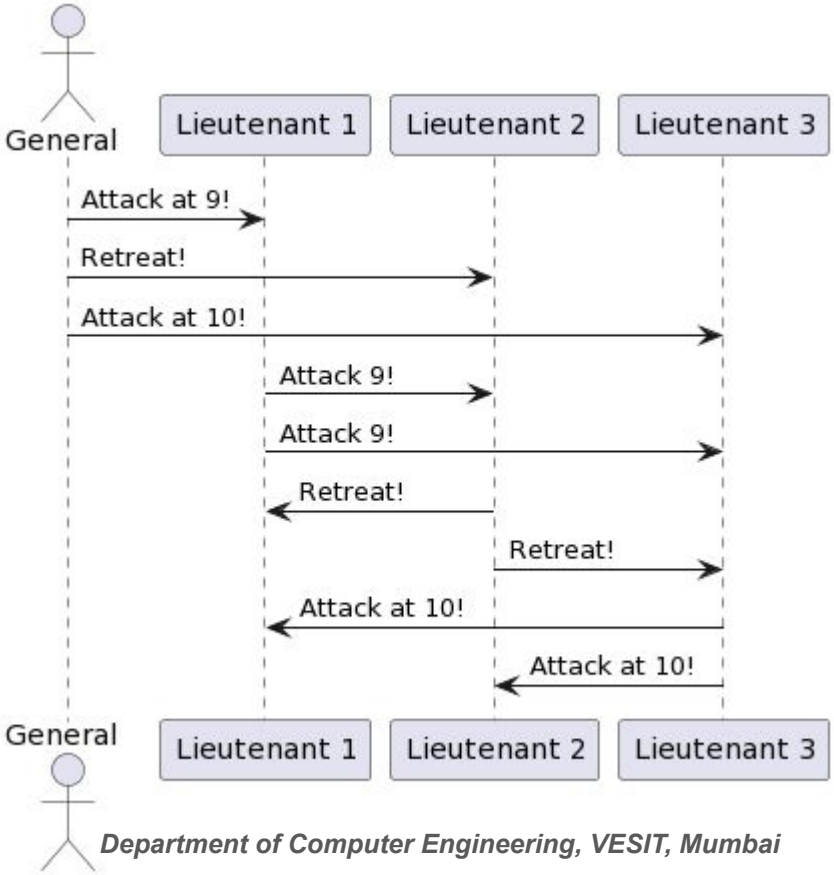
The consensus is maintained as long as 66% of the nodes are not corrupt

Byzantine Generals' problem

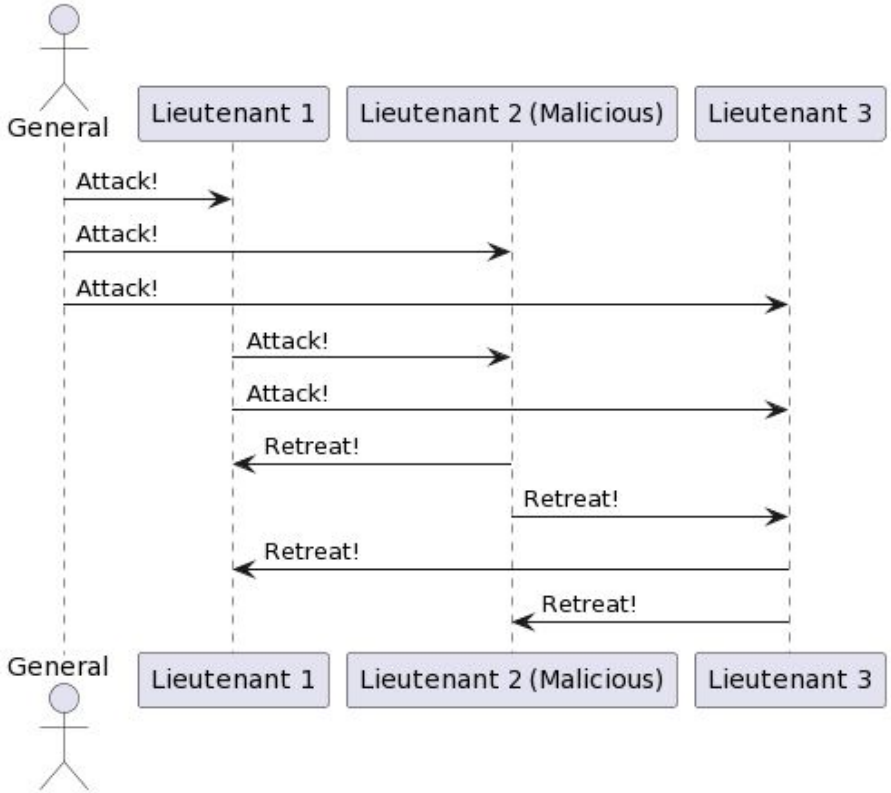
The Byzantine Generals' problem refers to the game theory problem. A group of generals attacks a fortress; every general has an army and surrounds a fort from one side. Every general has a preference about whether to attack or retreat. It has to be a coordinated attack or retreat to incur minimum losses.



Case 1: General is Traitor



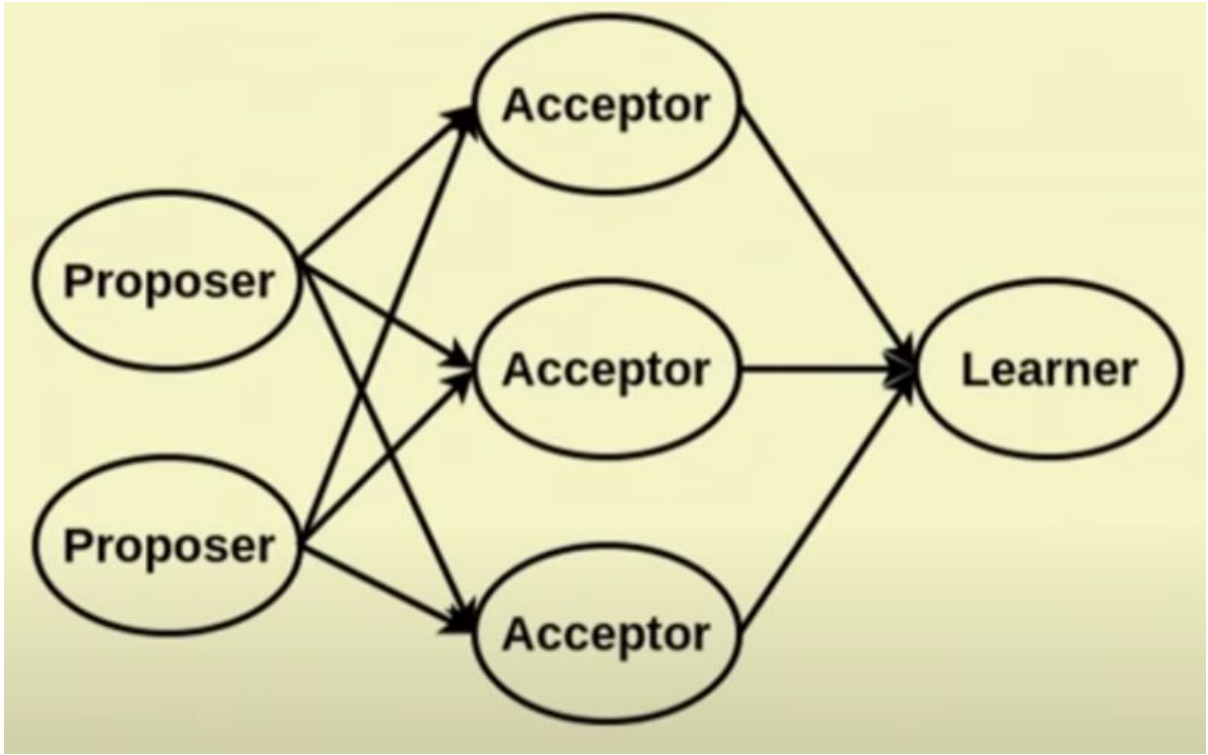
Case 2: Lieutenant is Traitor



Paxos

As one of the main algorithms in distributed consensus, Paxos requires at least two phases to reach an agreement:
The preparation phase (the request to prepare)
The acceptance phase (the request to accept).

It is a majority wins algorithm



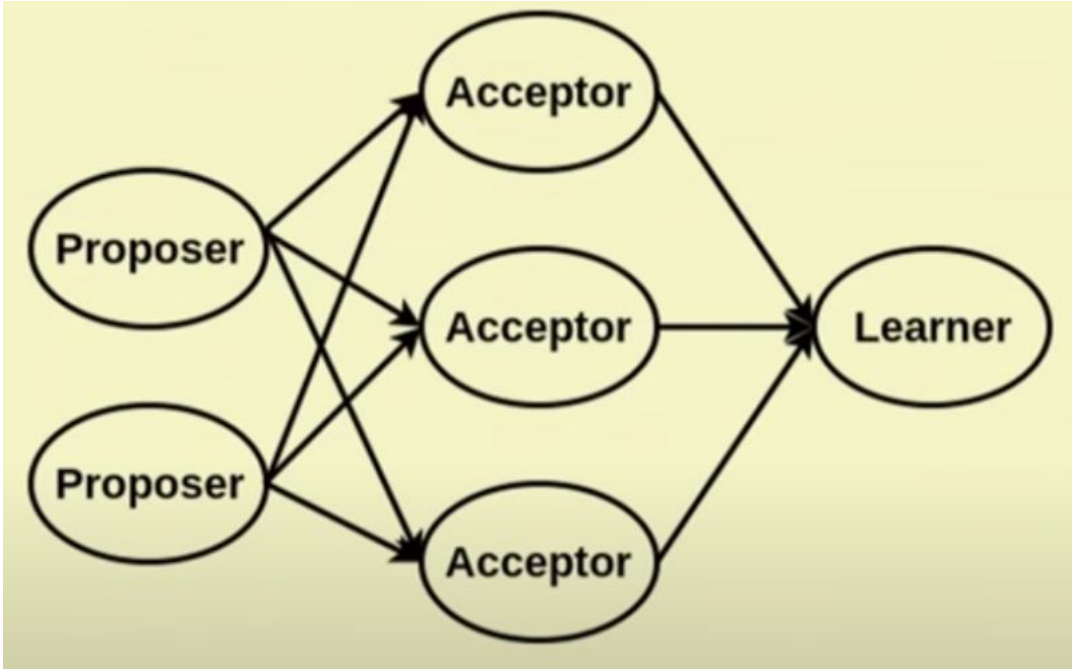
A simple analogy to understand Paxos

- Director is absent
- People easily distracted
- No fun if group splits up
- Hungry: must come to decision fast
- Yeiling fails : one to one communication only.

Approach to solve the issue

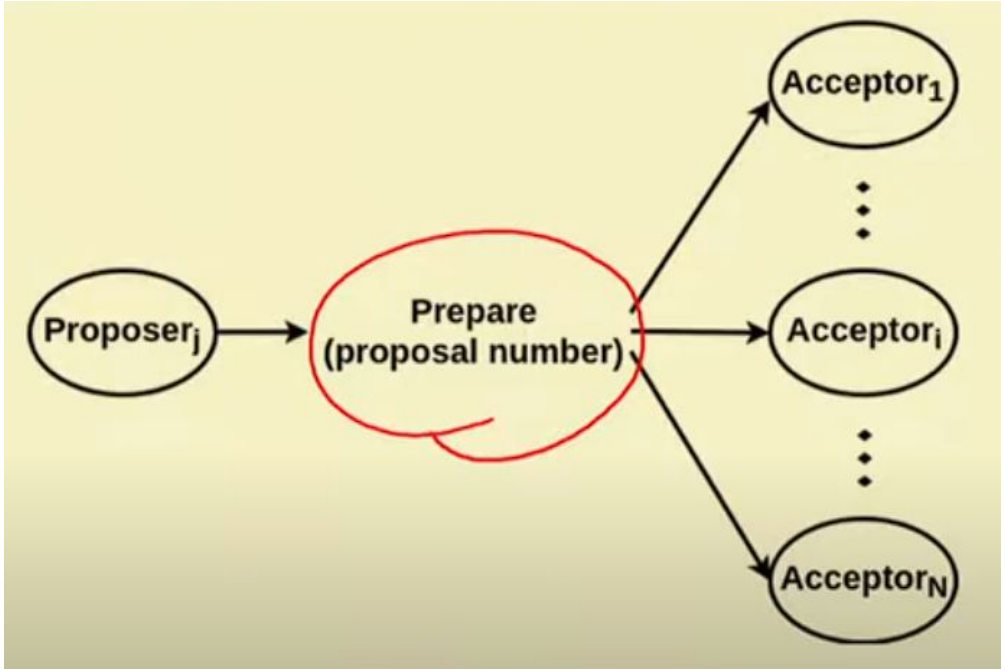
1. You ask your friends - What's in there mind?
2. If there is nothing then you propose something to them
3. If they have something in mind then they propose you
4. In this way slowly all the folks have made up their minds mind to eat something
5. At the end we just decide who has the majority.

Types of Node



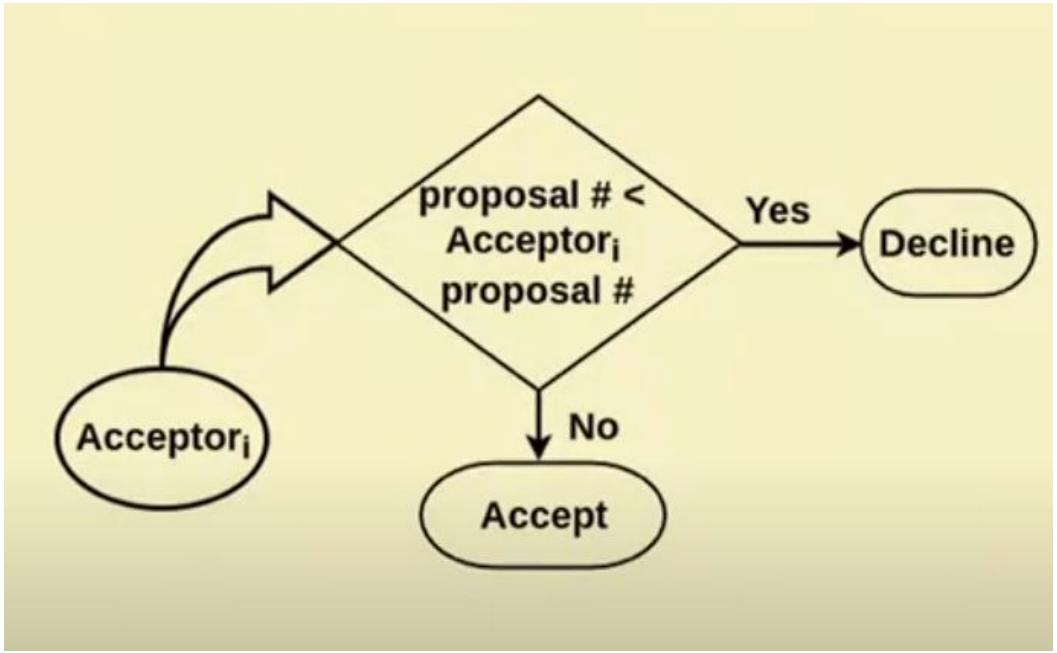
- **Proposer**: propose values that should be chosen by the consensus
- **Acceptor**: form the consensus and accept values
- **Learner**: learn which value was chosen by each acceptor

Making Proposal: Proposer Process



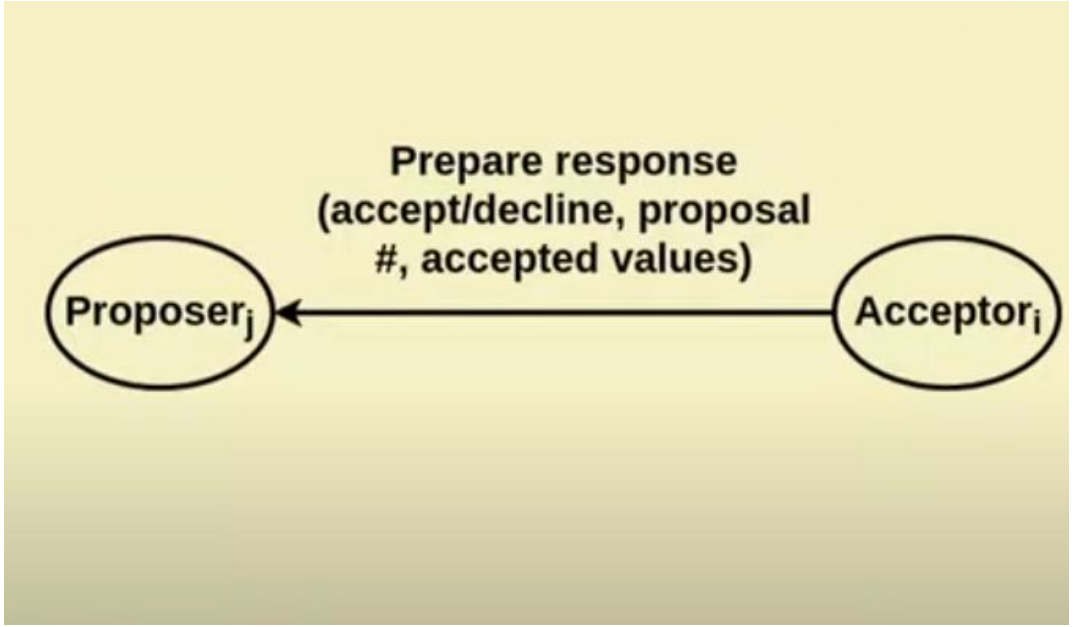
- **Proposal number:** form a timeline, biggest number considered up-to-date

Making Proposal: Acceptor's Decision Making



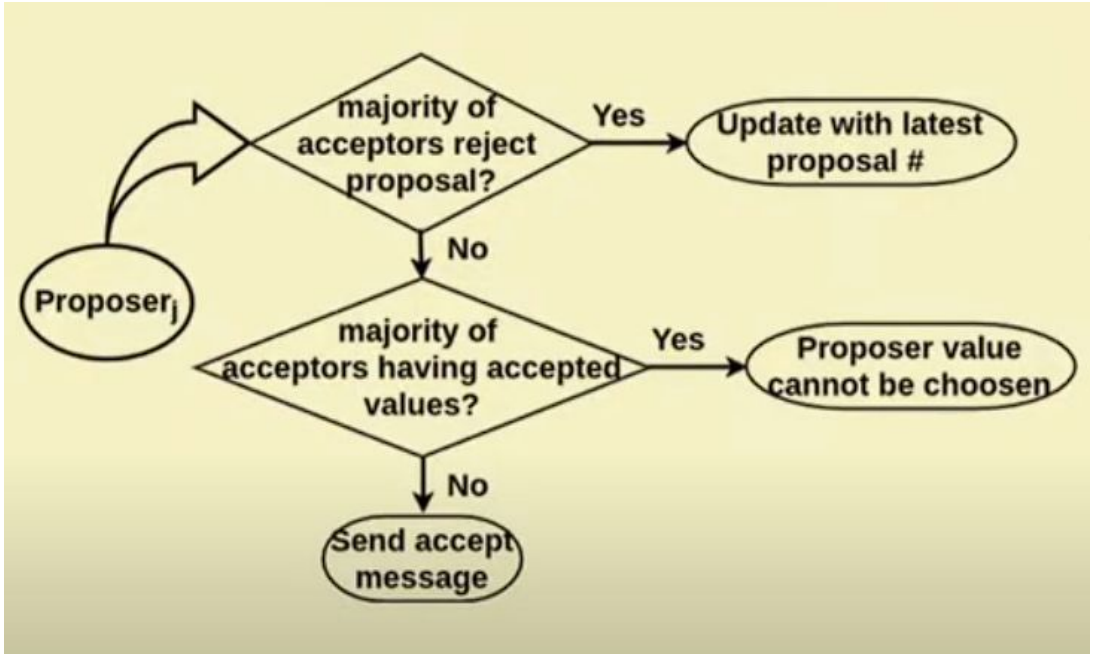
- Each acceptor compares received proposal number with the current known values for all proposer's prepare message

Making Proposal: Acceptor's Message



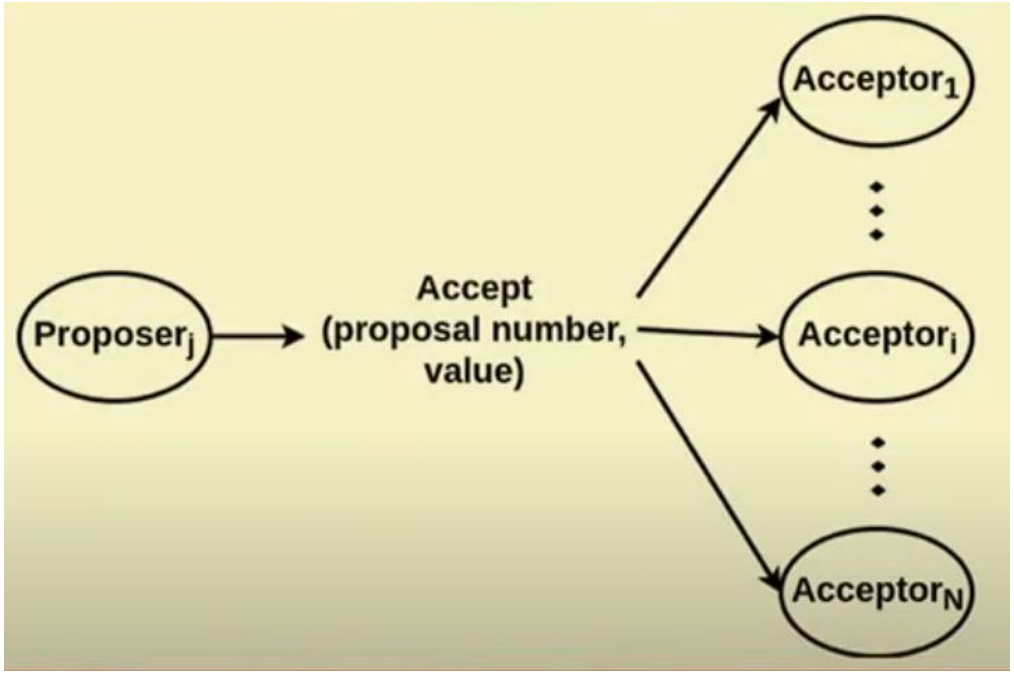
- **accept/decline**: whether prepare accepted or not
- **proposal number**: biggest number the acceptor has seen
- **accepted values:** already accepted values from other proposer

Making Proposal: Proposer's Decision Making



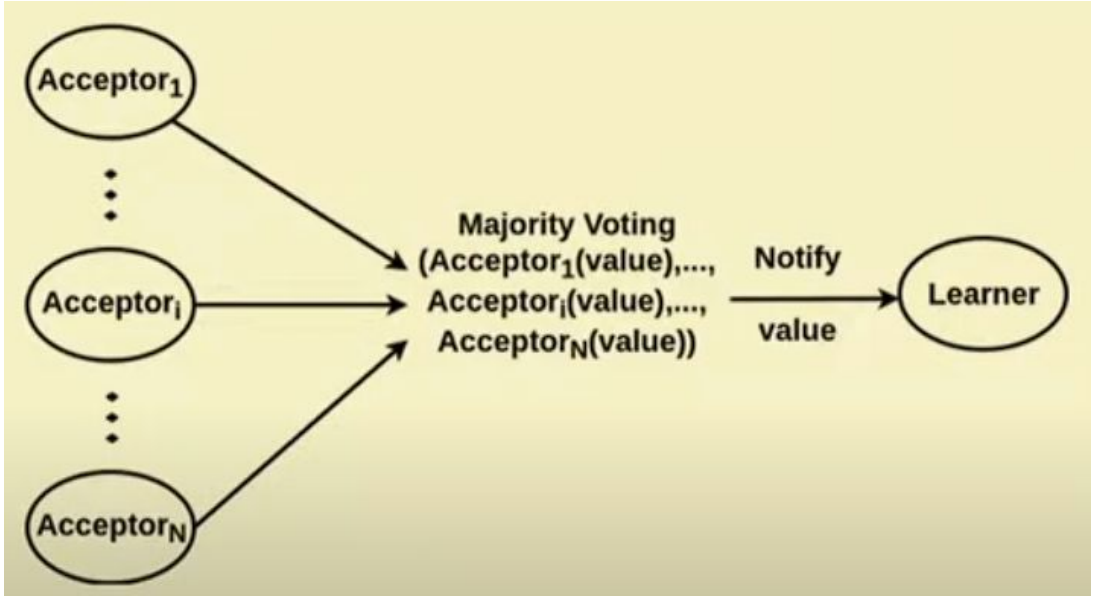
Proposer receive a response from **majority** of acceptors before proceeding

Making Proposal: Accept Message



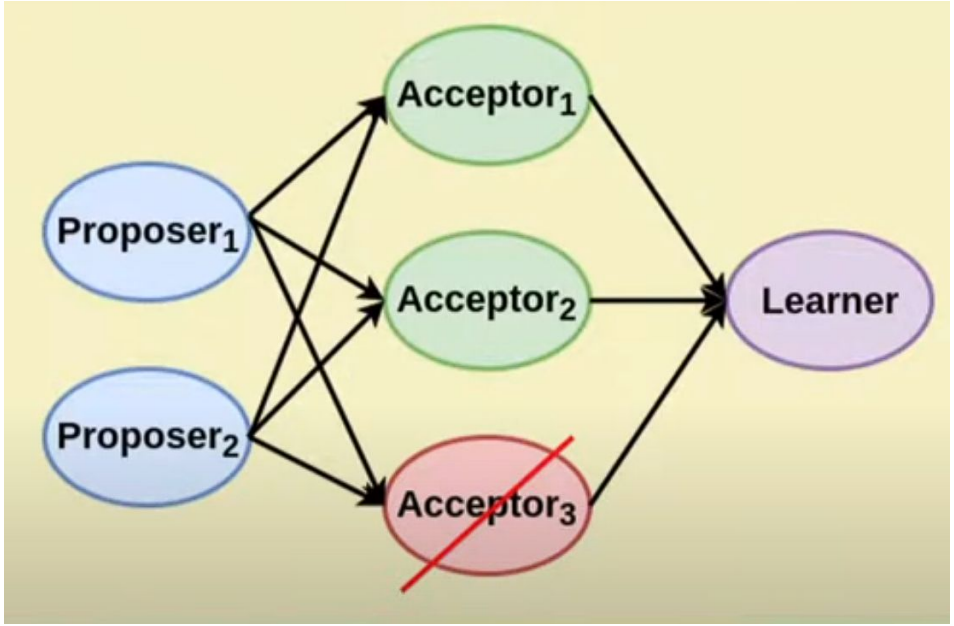
- **proposal number**: same as prepare phase value
- **value**: single value proposed by proposer

Making Proposal: Accept Message



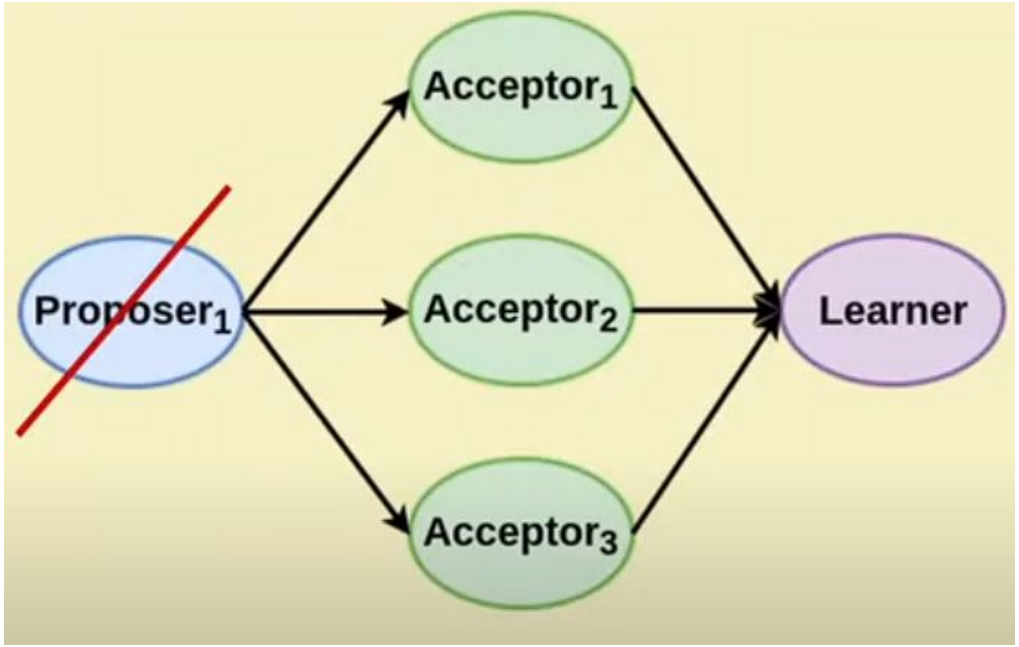
- Each acceptor accept value from any of the proposer
- Notify learner the majority voted value

Failure Handling: Acceptor Failure



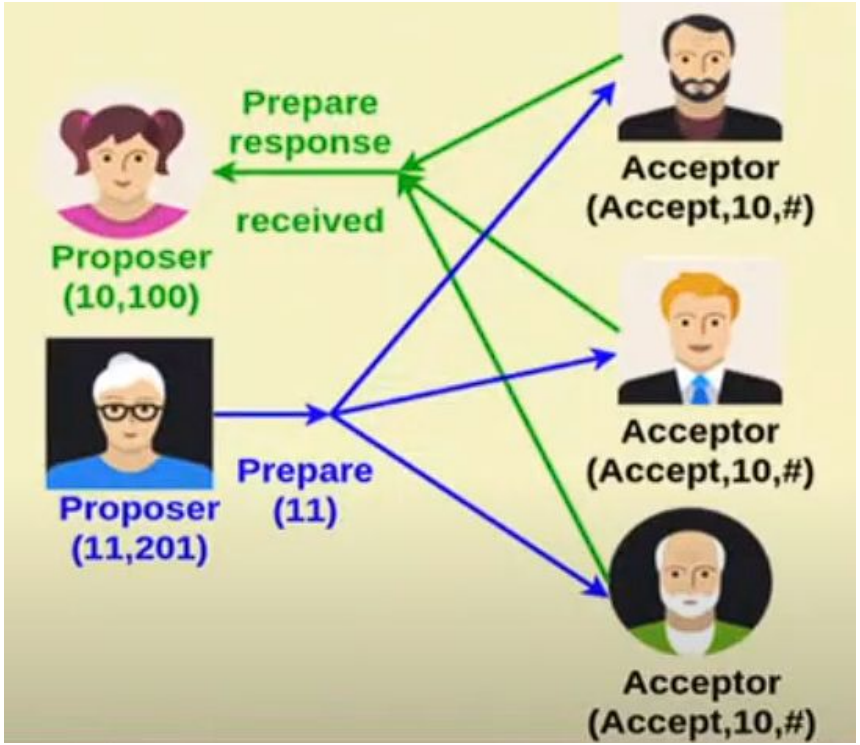
- No issues if the acceptor fails in the prepare phase or accept phase as other acceptor's present
- N/2 acceptors should not fail at an instance

Failure Handling: Proposer Failure



- **Prepare phase:** the acceptor wait for a duration and then if still no response is seen, a new Proposer is selected
- **Accept phase:** the proposal has already been received and they just have to come to one conclusion

Failure Handling: Dueling Proposers



- The first proposer is blocked as soon as a prepare message of higher value is received
- Leader is elected and leader decides who to block

Raft

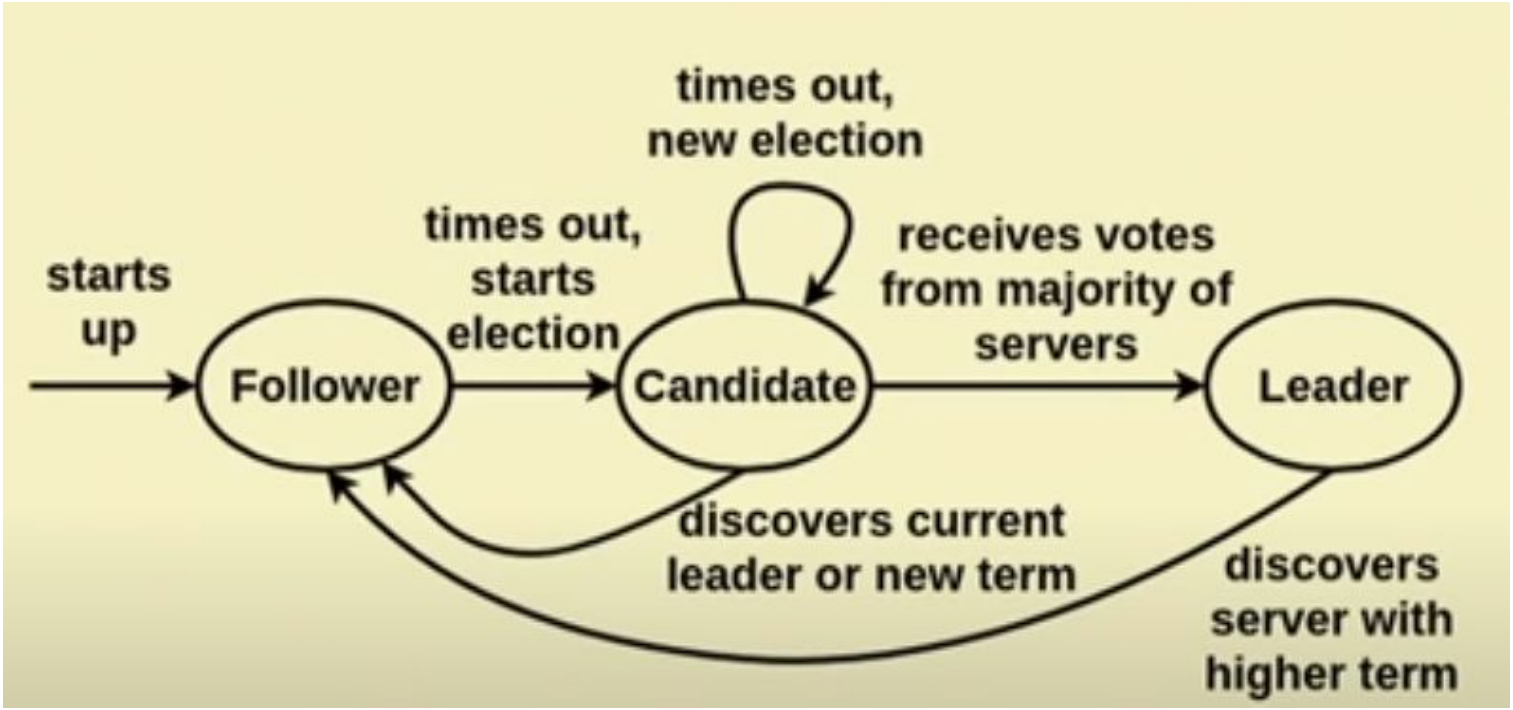
Designed as an alternative to Paxos

A generic way to distribute a state machine among a set of servers

Basic idea:

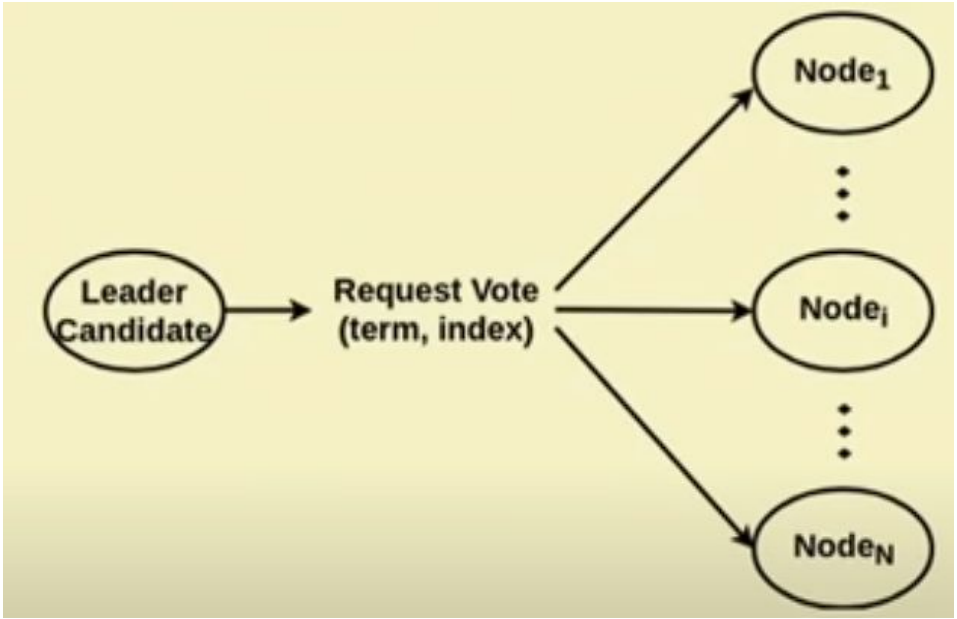
- The nodes collectively selects a leader; others become followers
- The leader is responsible for state transition log replication across the followers

RAFT algorithm



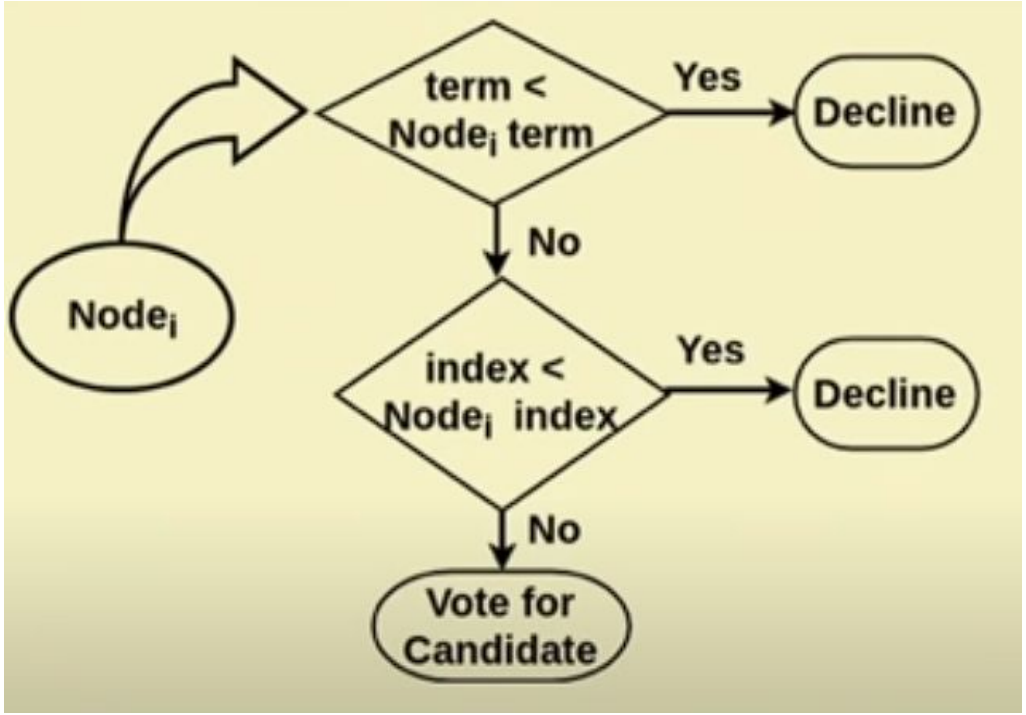
Leader Election mechanism

- **term :** last term + 1
- **Index :** committed transaction available to the candidate



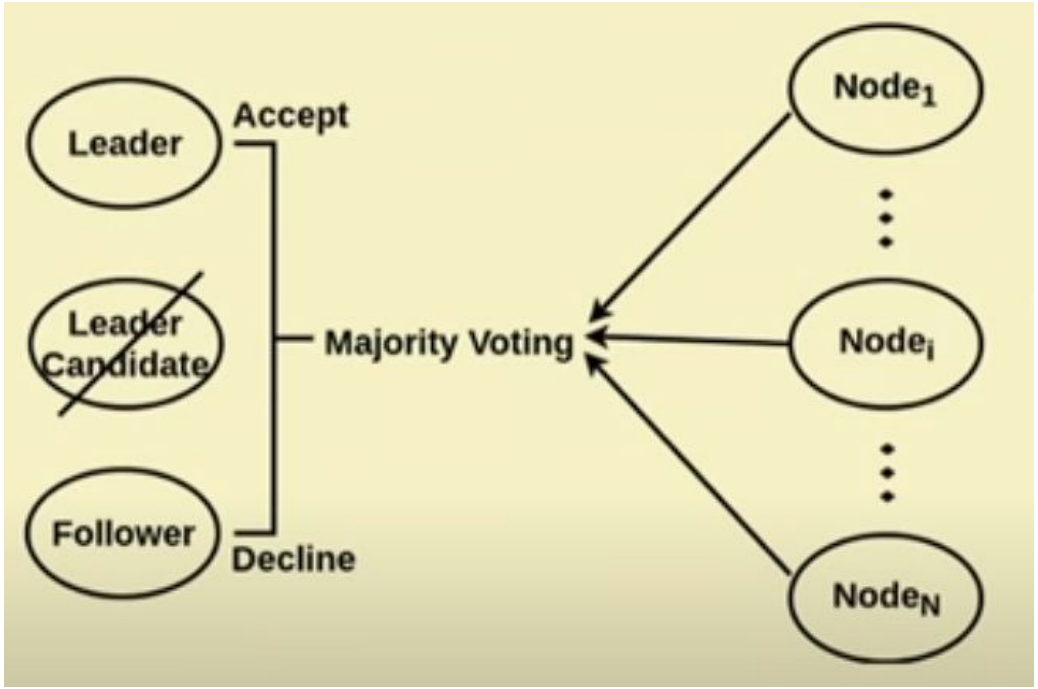
Leader Election mechanism

- Each node checks the term and index value with their current values

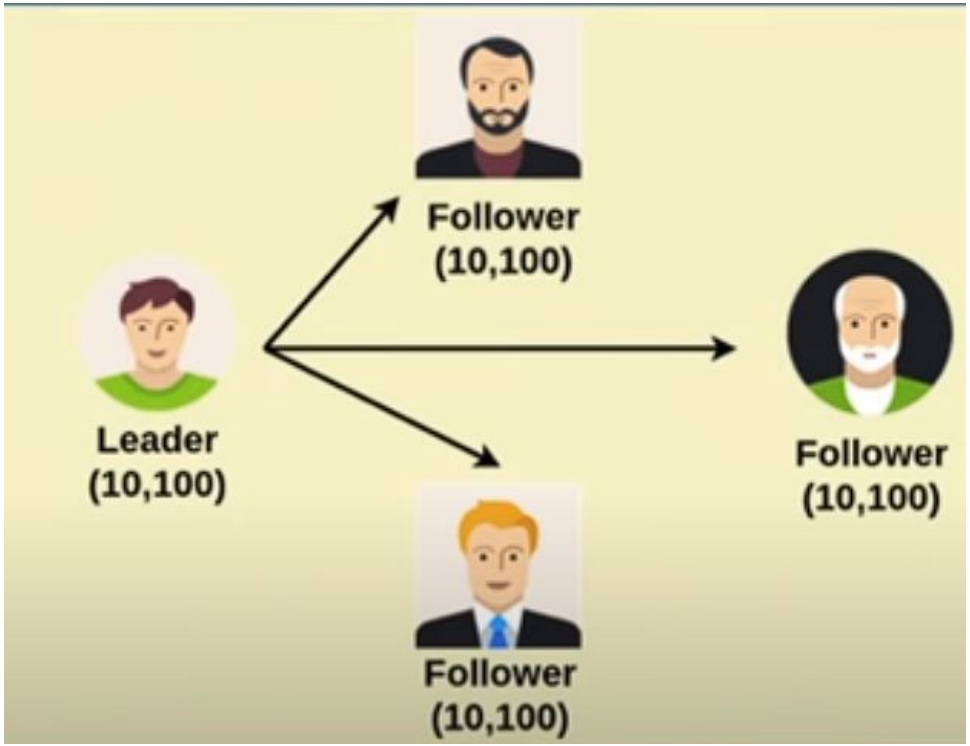


Leader Election mechanism

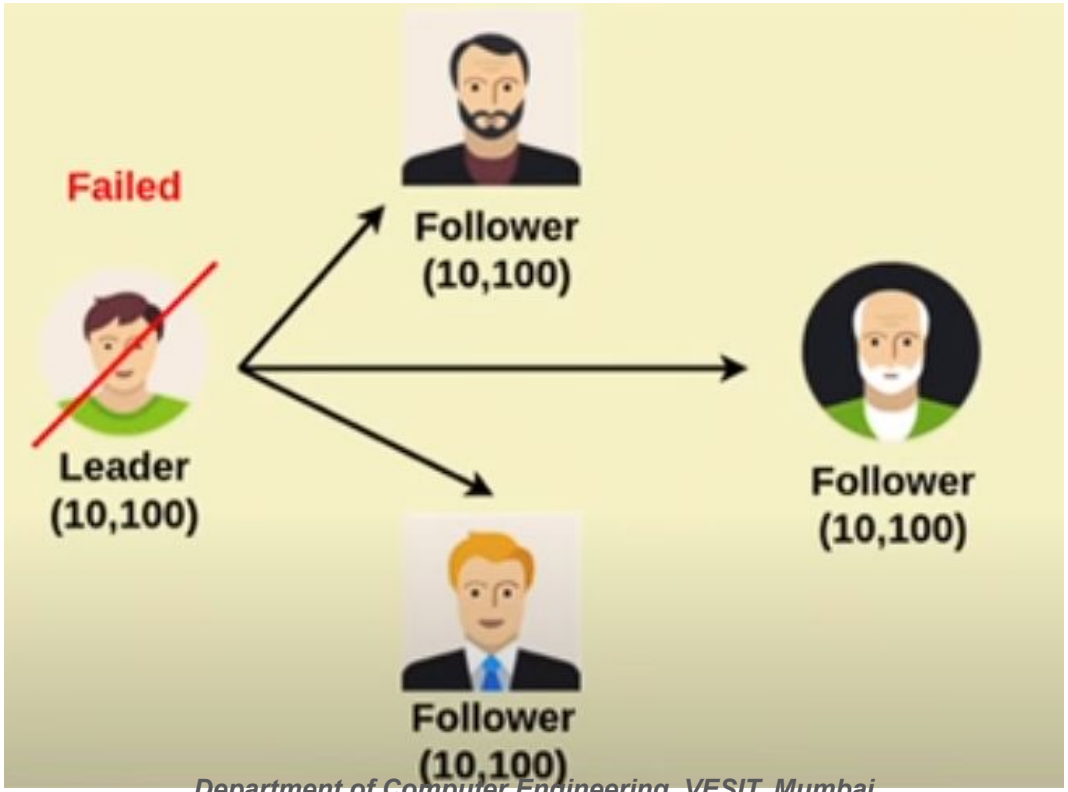
Using the majority voting leader are selected



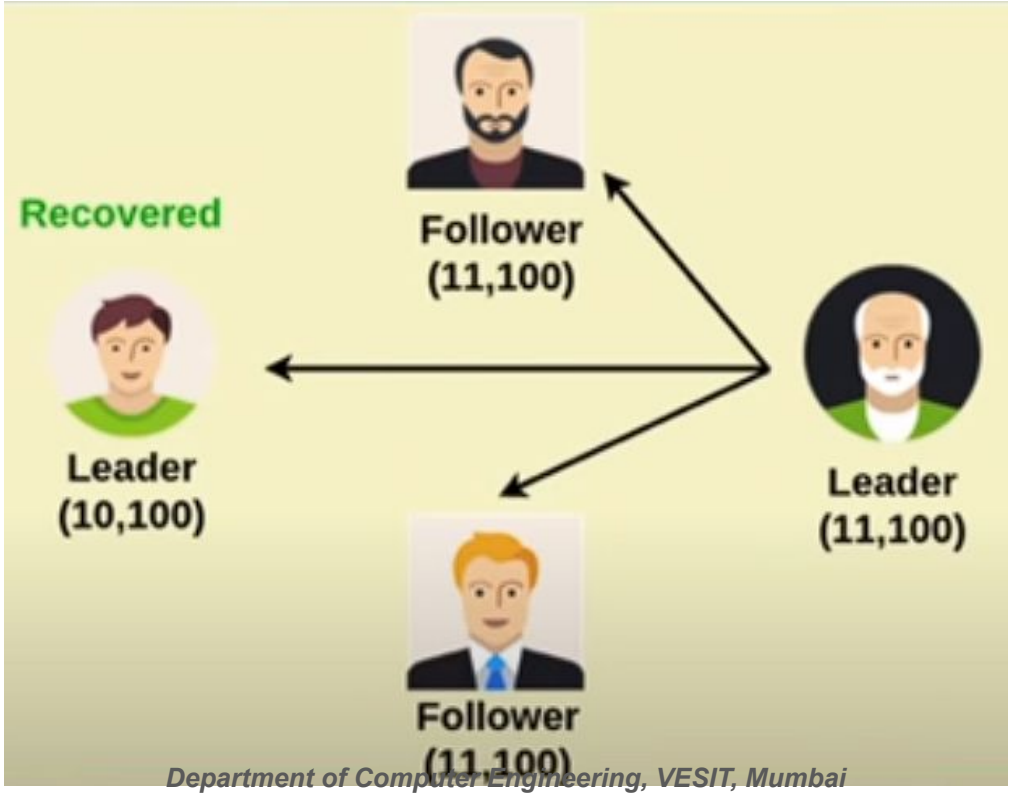
Current Leader failure



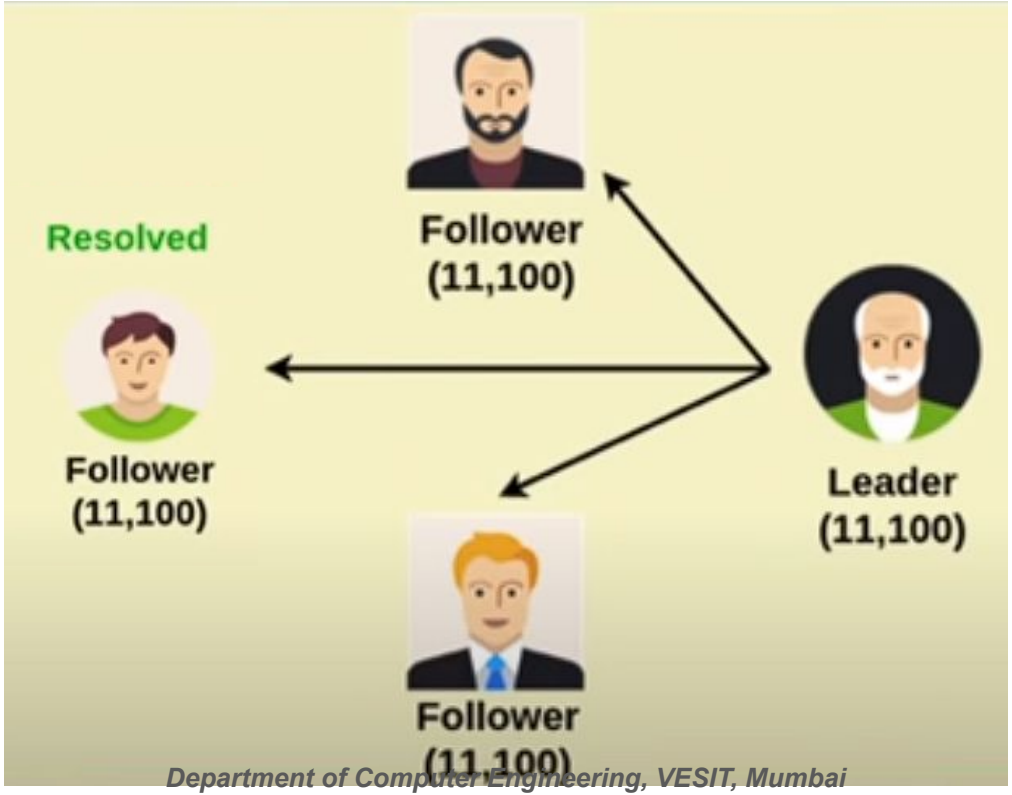
Current Leader failure



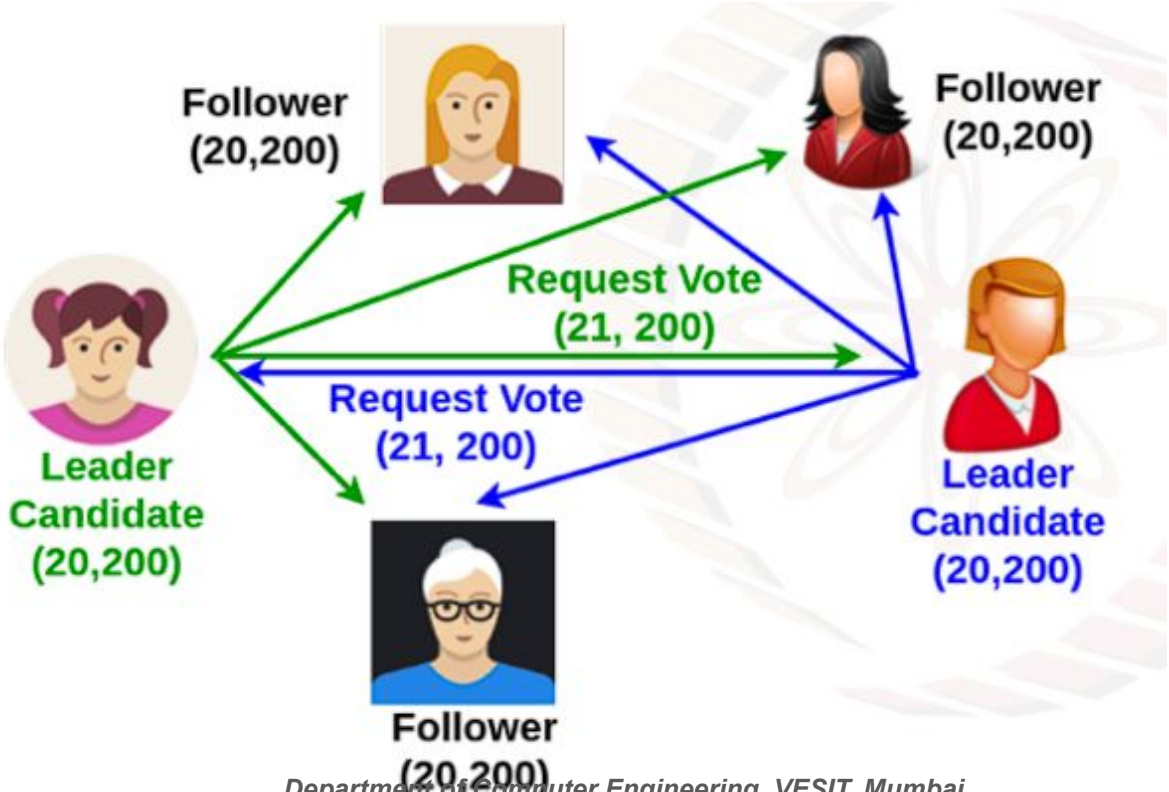
Current Leader failure



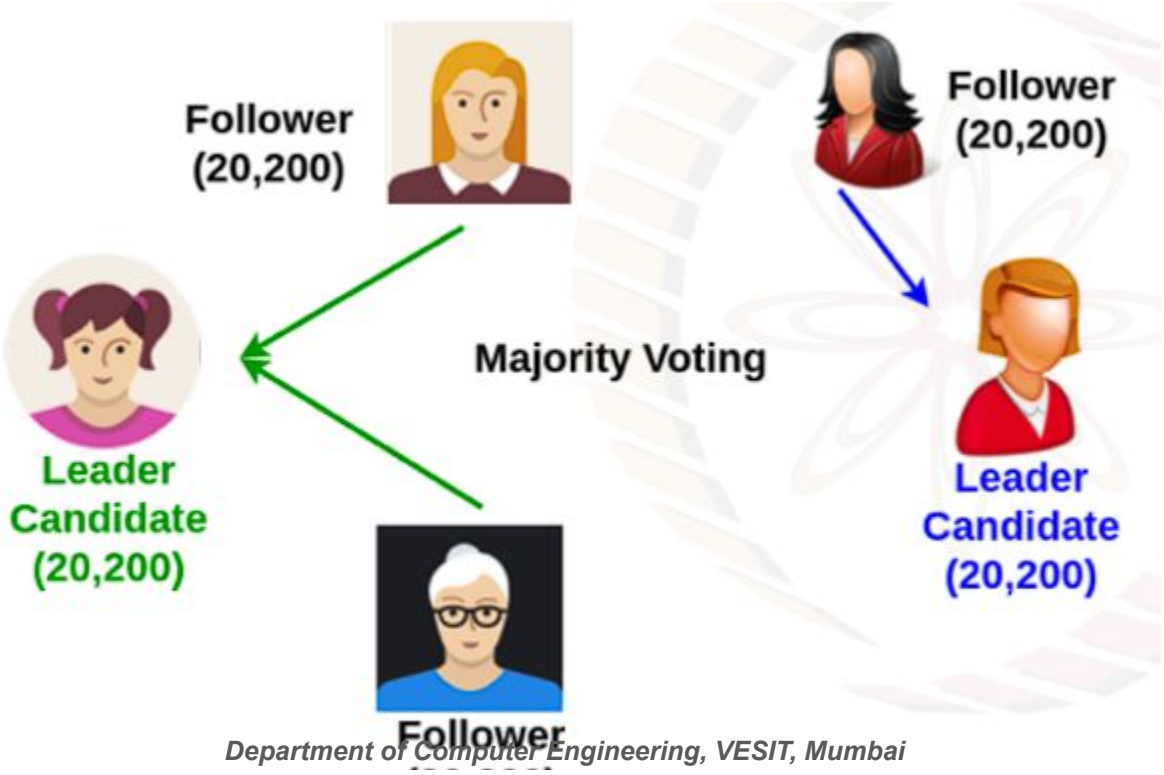
Current Leader failure



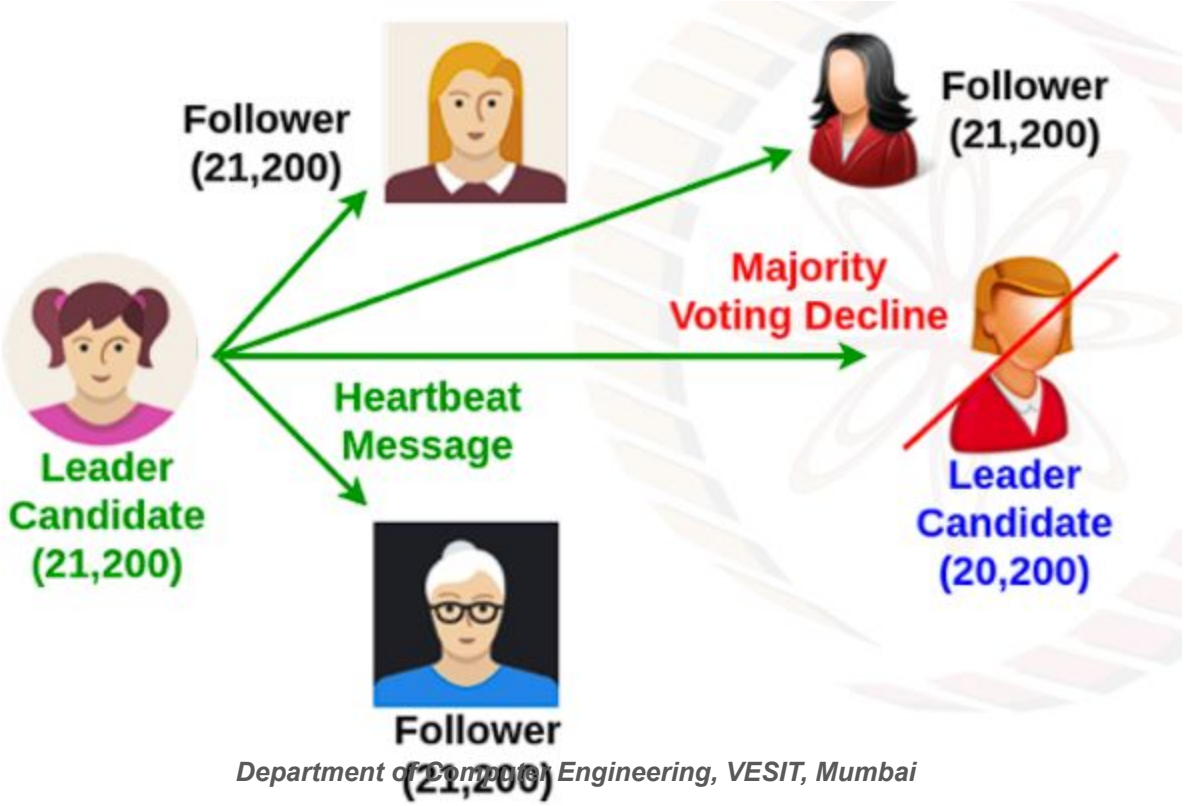
Two vote request with same term and index



Two vote request with same term and index



Two vote request with same term and index

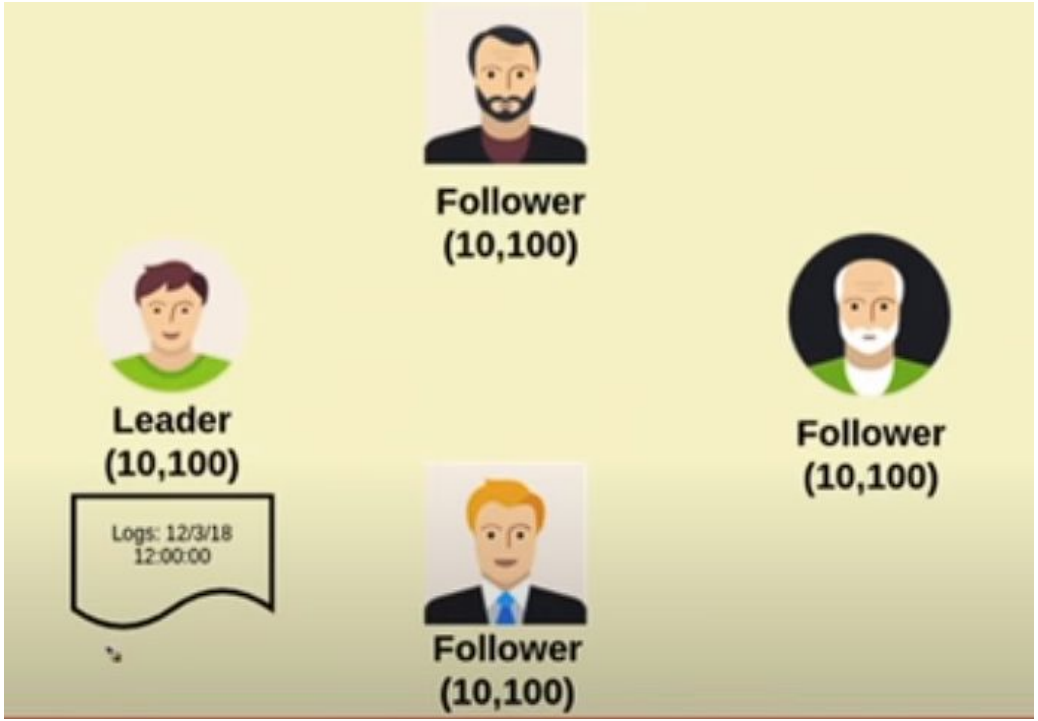


Committing Entry Log

When a new log is received by the leader he adds it

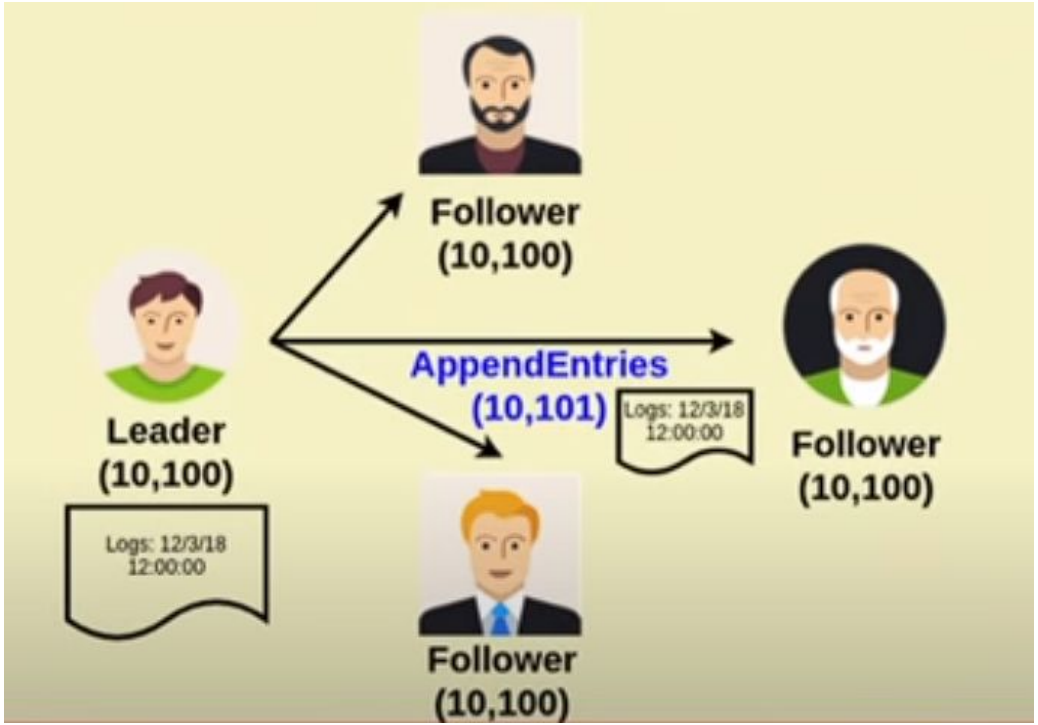
term : current term

Index : previous index + 1



Committing Entry Log

Leader send AppendEntries message to all its followers

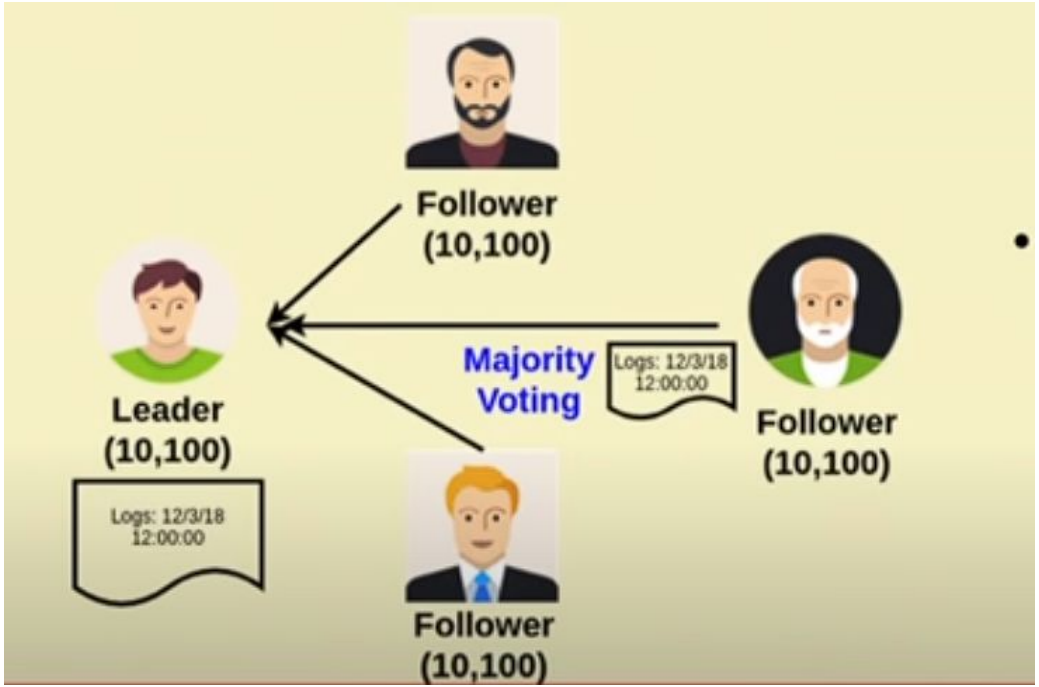


Committing Entry Log

Followers will send an accept or reject response

Finally majority is seen and the transaction / log gets added or not

All update to the latest state



Failure handling

Failure of up to N/2 - 1 nodes can be handled by RAFT consensus
Same as Paxos : N/2 nodes are required for majority

Hyperledger

- Hyperledger is open source collaborative effort created to advance cross industry blockchain technology.It is a global collaboration hosted by linux foundation.
- Not company
- Not Blockchain
- Not a cryptocurrency

Hyperledger Fabric

- Hyperledger Fabric platform is an open source blockchain framework hosted by The Linux Foundation

Features of Hyperledger Fabric

- Highly modular and permissioned architecture.
- Multi-language smart contract support for different languages such as JavaScript, Go, and Java.
- Flexible data privacy approaches with data isolation through channels and exclusive data sharing through private data collections.

Channel

A channel is a private communication pathway between two or more members of a Hyperledger Fabric network

Different types of Nodes

- Committing node - maintain ledger

  - Commit transaction

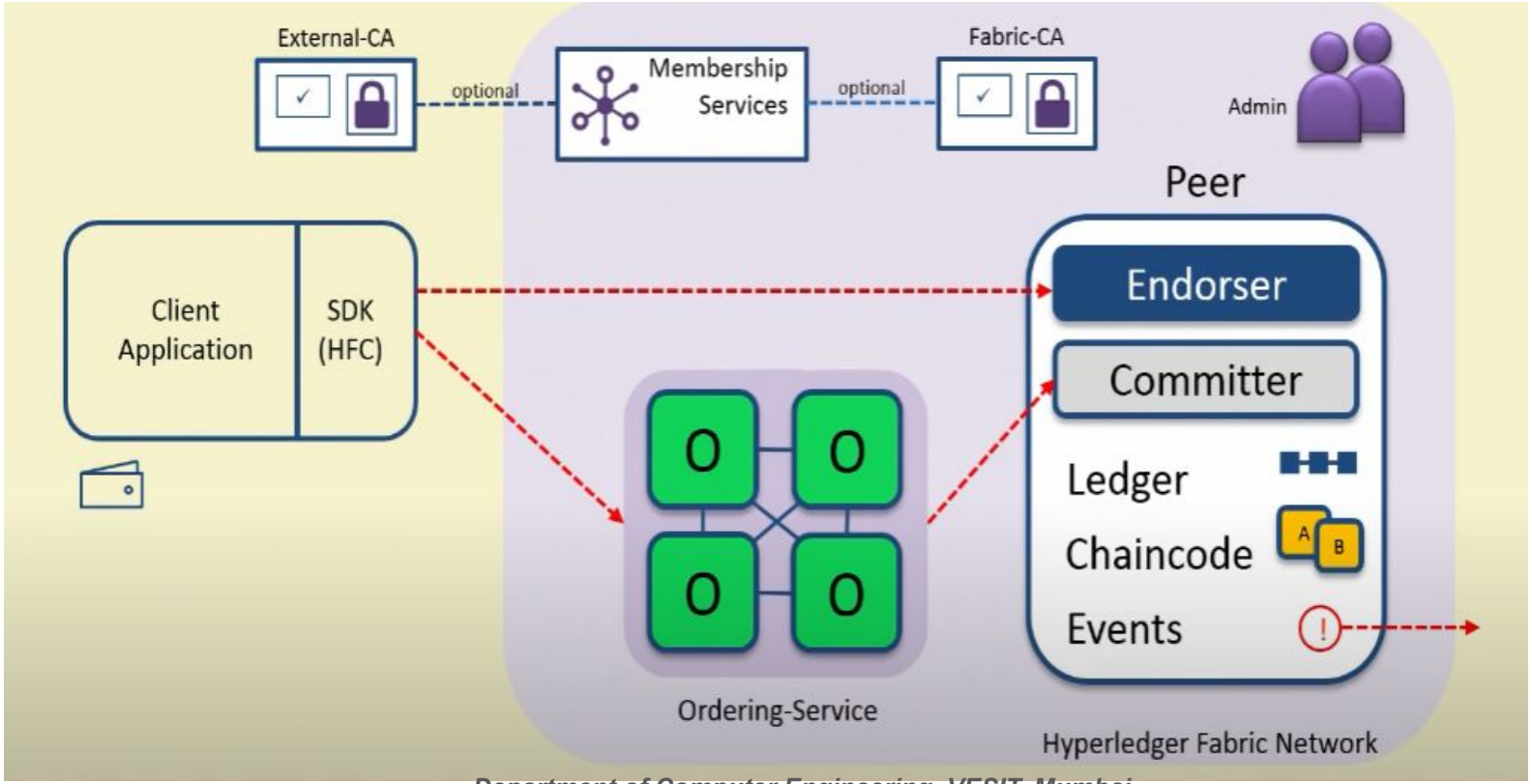
- Endorsing node - Contains smart contract

  - Responds the granting or denying transaction

- Ordering node - maintain the sequence of transactions

  - Does not hold ledger

Hyperledger Fabric Architecture

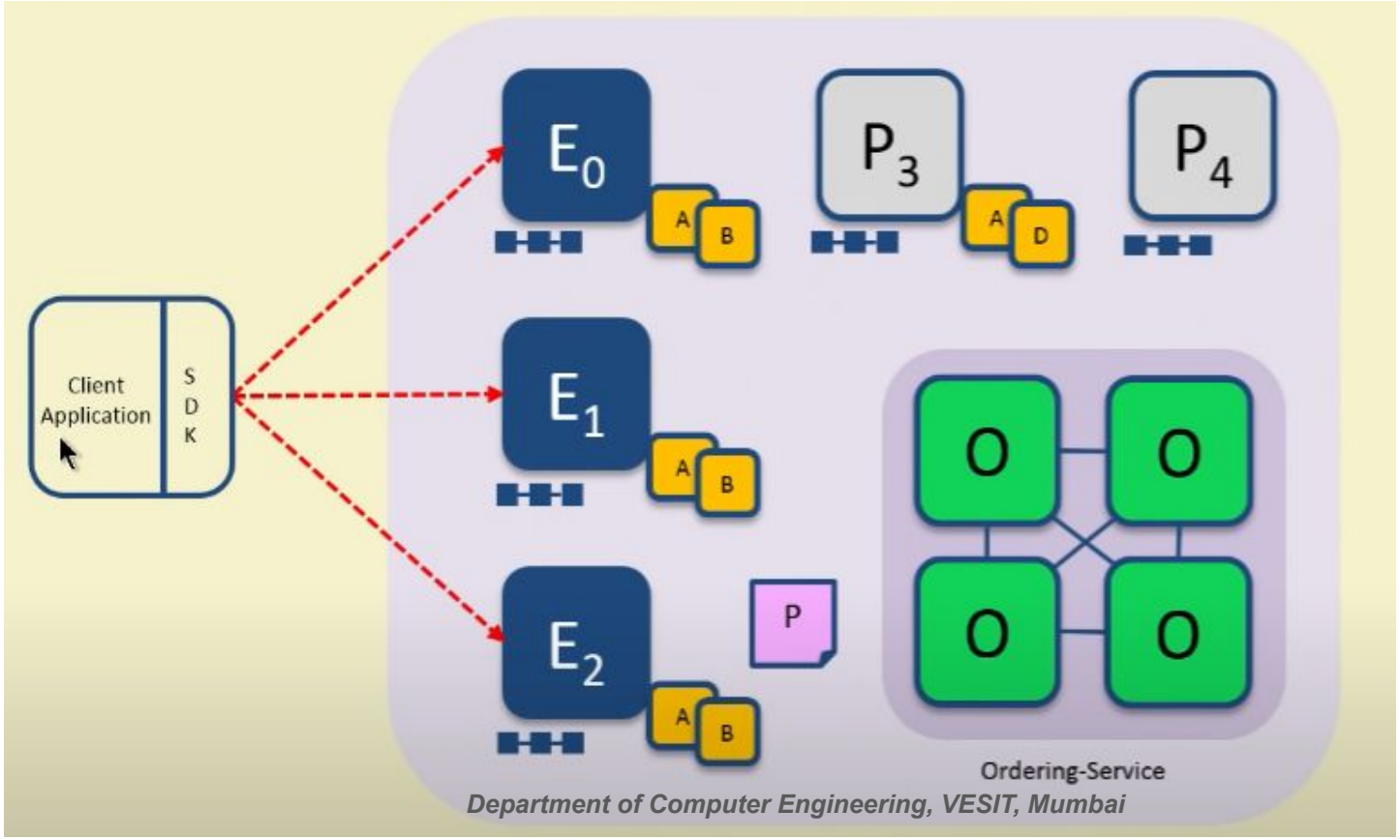


Transaction flow

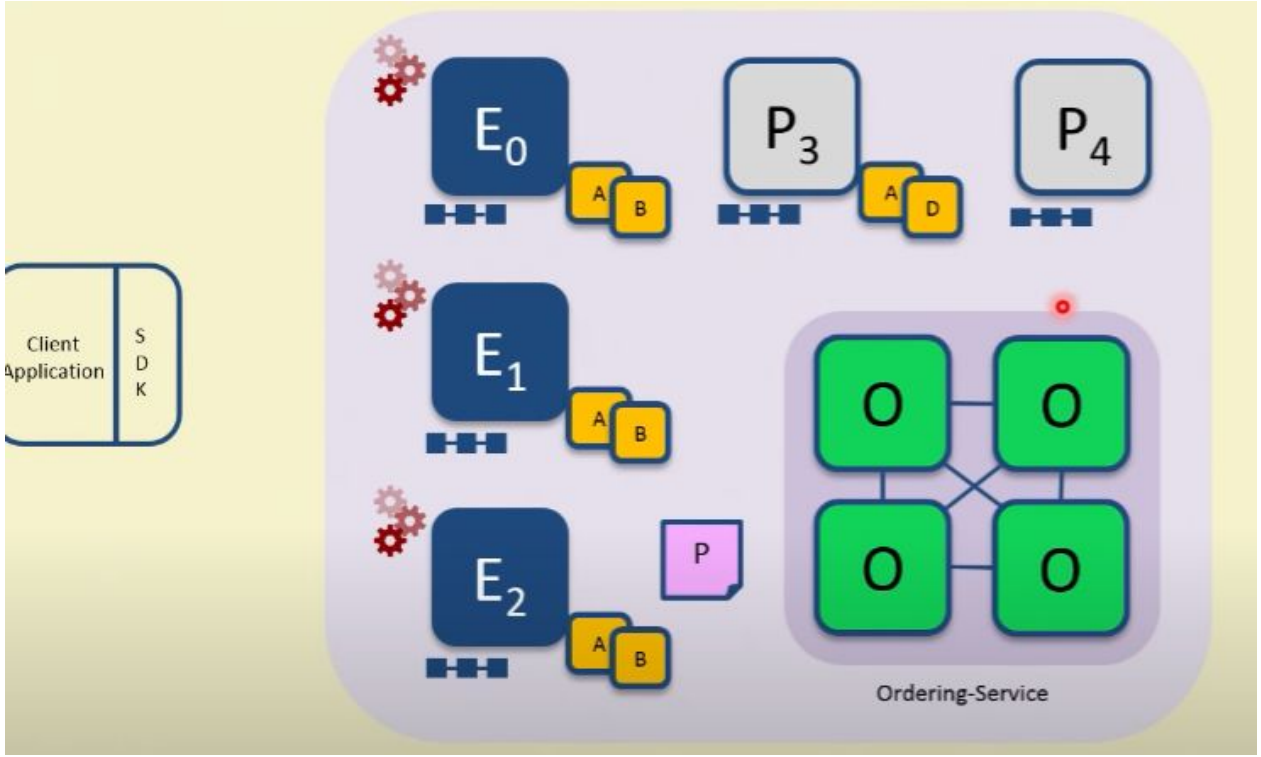
Transaction flow steps

1. Propose transaction
2. Execute proposed transaction
3. Propose response
4. Order transaction
5. Deliver transaction
6. Validate transaction
7. Notify transaction

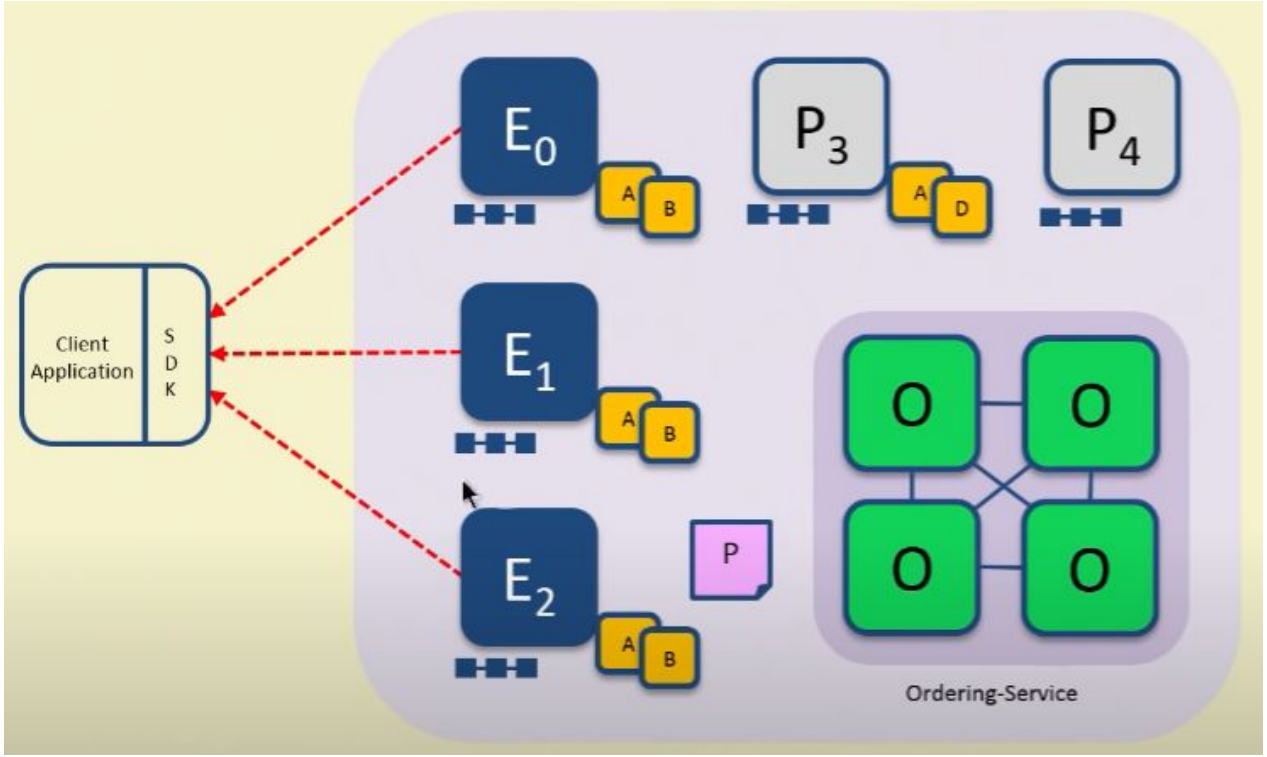
1. Proposed Transaction



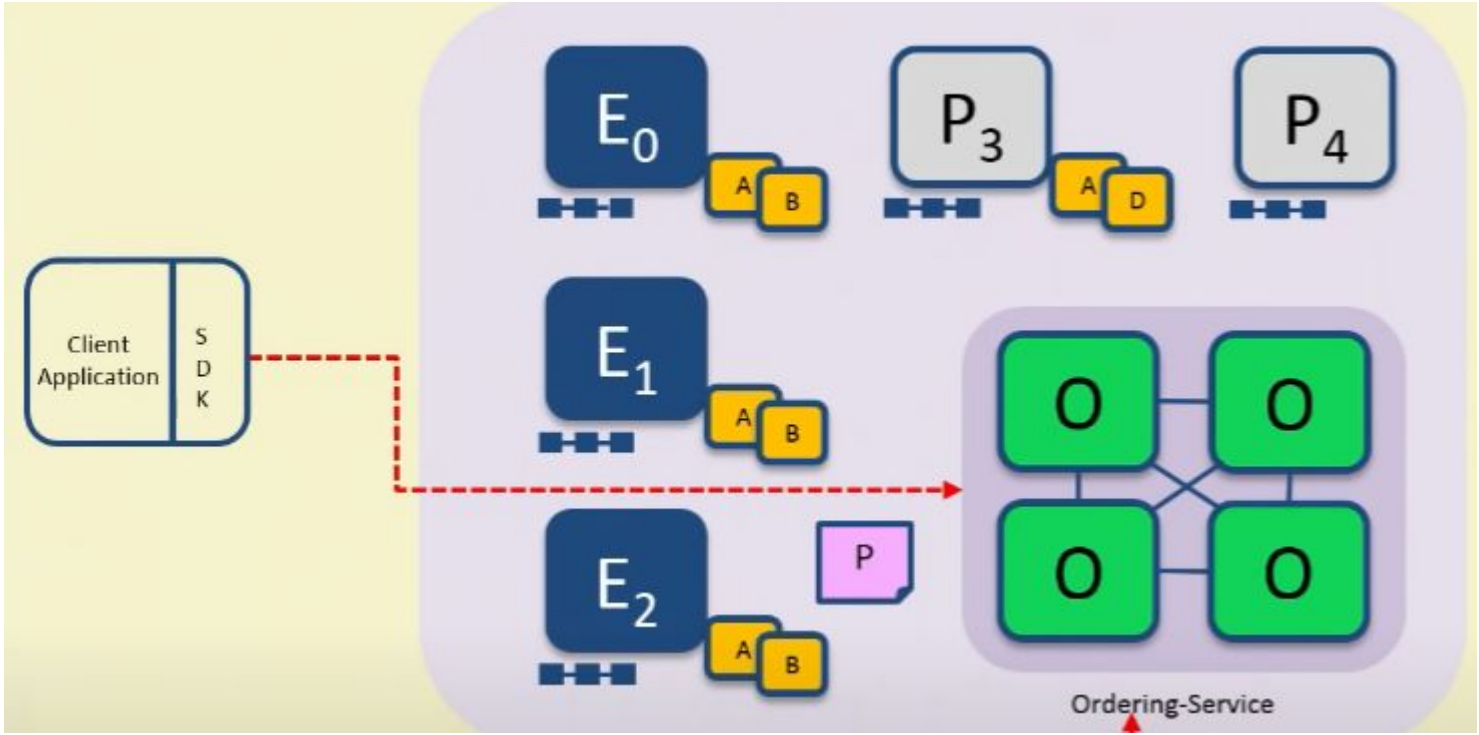
2. Execute proposed transaction



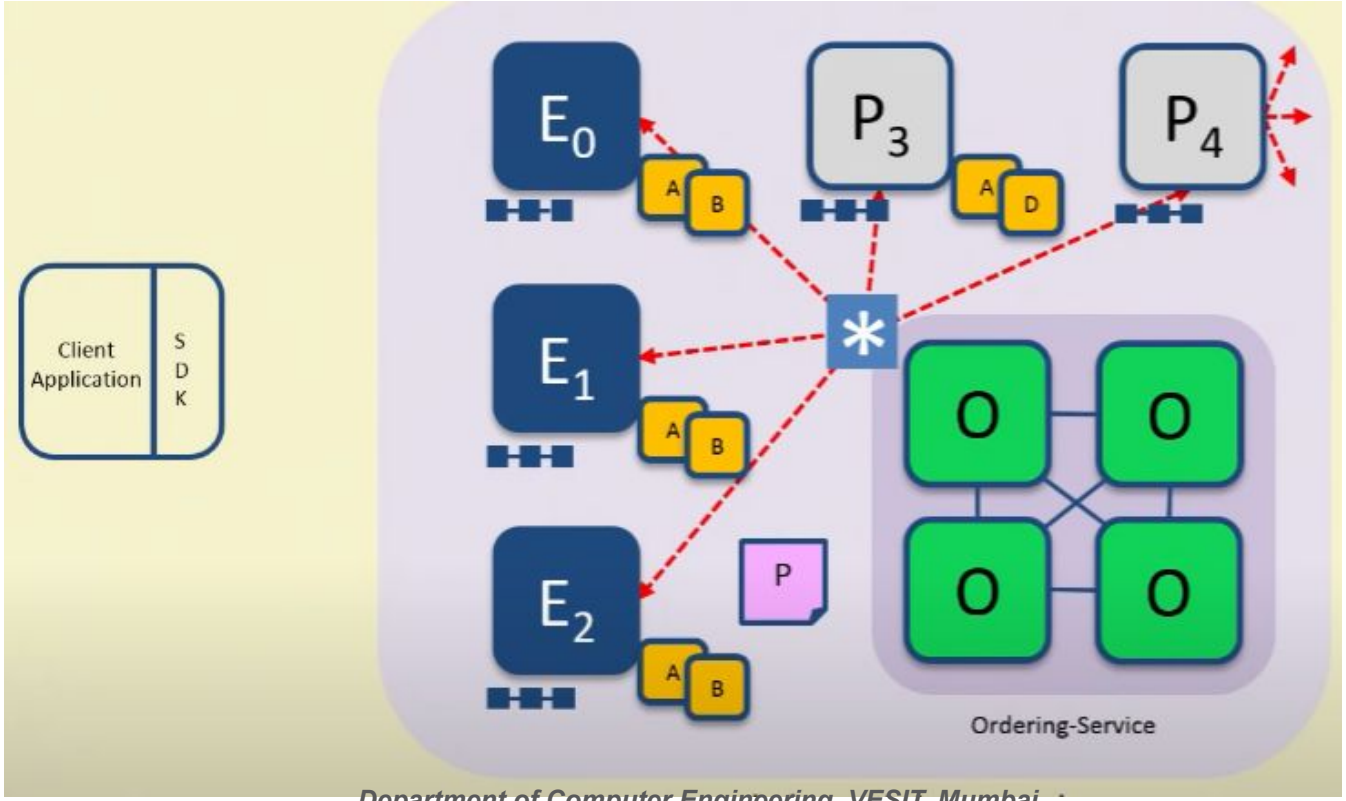
3. Proposal Response



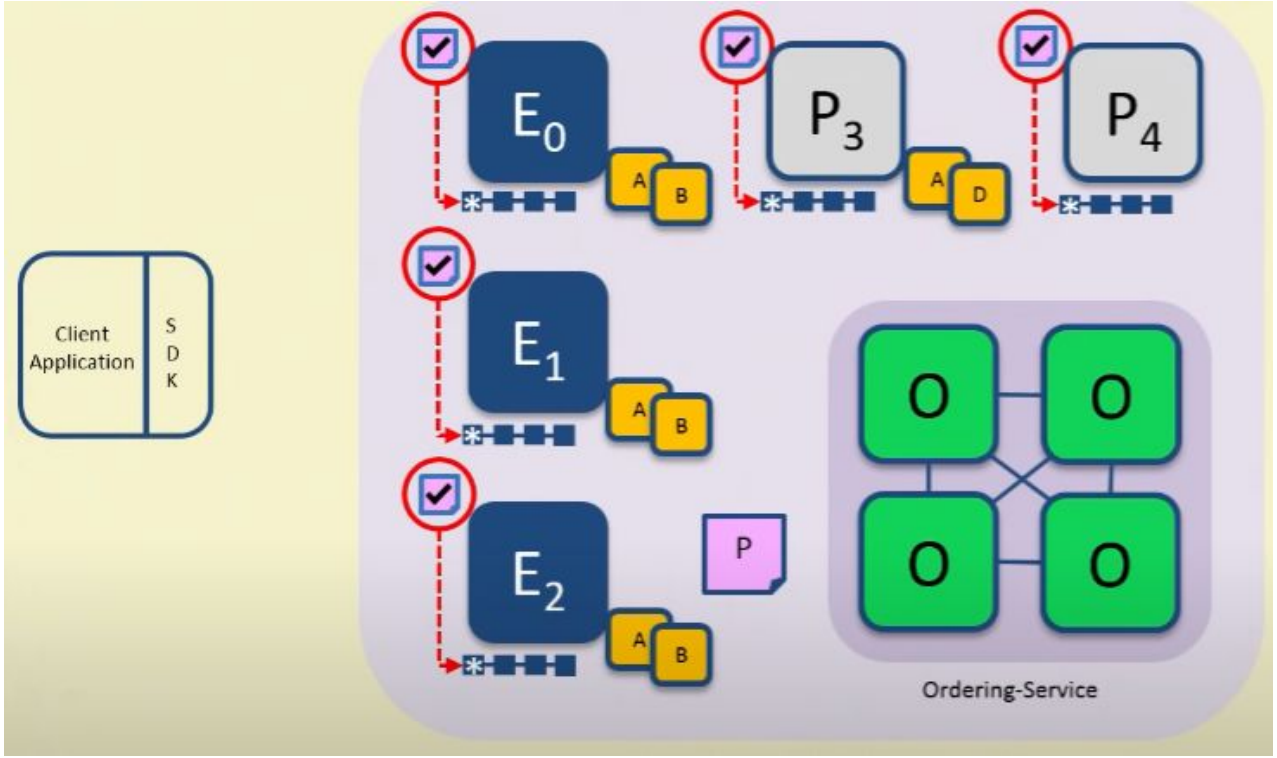
4. Order Transaction



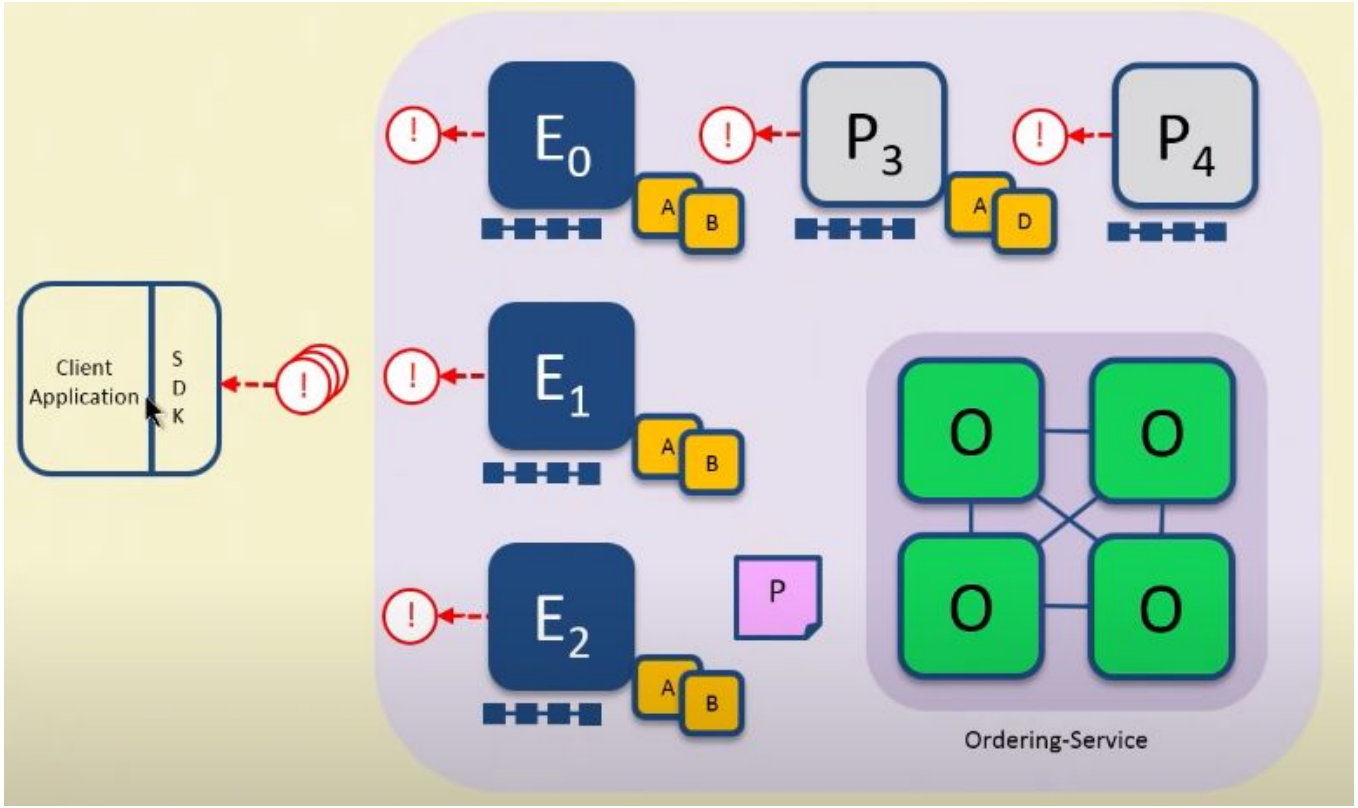
5. Deliver Transaction



6. Validate Transaction



7. Notify Transaction



HYPERLEDGER

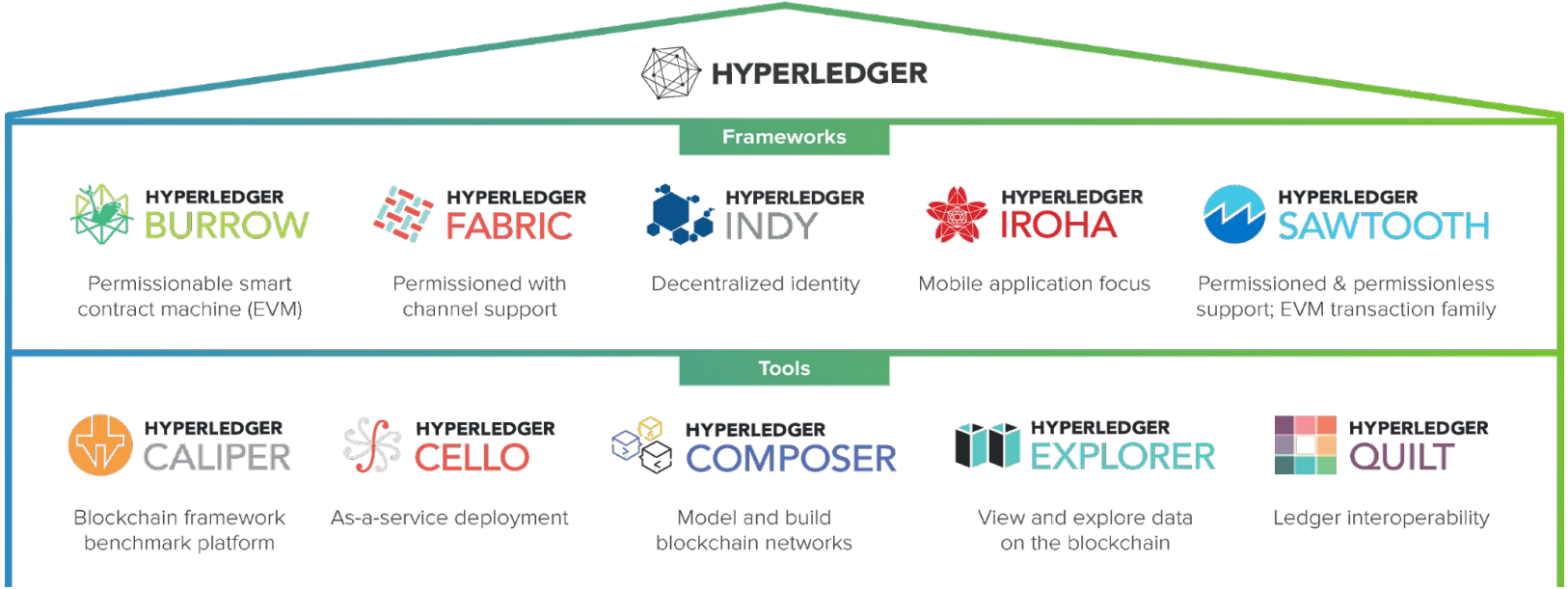
HYPERLEDGER

In 2015, the Linux Foundation unveiled an umbrella open-source blockchain project called Hyperledger. Under the leadership of Brian Behlendorf, Hyperledger is being used almost in all enterprise blockchains projects.

- It works by providing the necessary infrastructure and standards for developing various blockchain-based systems and applications for industrial use.
- **four major tools initiatives**, each having its distinct role in the development of the environment, applications, etc.

What is Hyperledger?

- Incubator of open source Blockchain projects created by The Linux Foundation
- Aims to bring the technology to companies

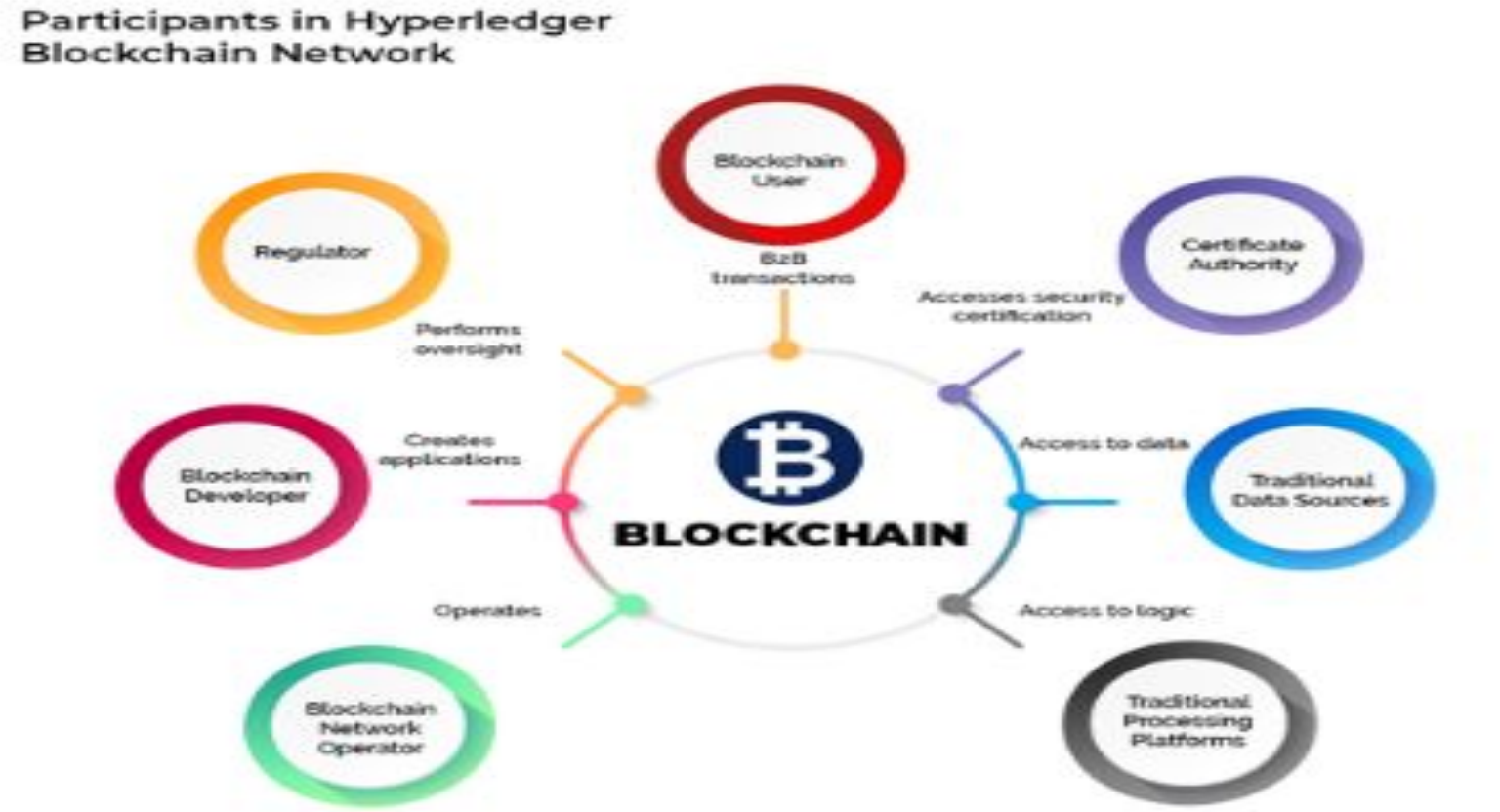


Hyperledger projects

- Hyperledger INDY
- Hyperledger SAWTOOTH
- Hyperledger IROHA
- Hyperledger FABRIC
- Hyperledger BESU

https://www.hyperledger.org/learn/blockchain-showcase

Fabric Architecture:



Features of Hyperledger Architecture

- Modular Design
- Extremely Secure Platform
- Interoperable
- Cryptocurrency-agnostic
- High end API Support

WHAT IS HYPERLEDGER?

Hyperledger is an open-source blockchain umbrella project with loads of free tools and frameworks for blockchain solutions.

Hyperledger has 15 projects, and only four are currently active.

FEATURES OF HYPERLEDGER ARCHITECTURE

- Modular Design
- Extremely Secure Platform
- Interoperable
- Cryptocurrency-agnostic
- High-end API Support

FOUR ACTIVE PROJECTS TO CHECK OUT

HYPERLEDGER FABRIC

Hyperledger Fabric is a modular distributed ledger network that offers developers with the highest quality of applications.

FEATURES

- Peer-To-Peer Gossip Services
- Membership Service Providers
- Ordering Services
- Smart Contracts
- Modular consensus
- Efficient processing

HYPERLEDGER INDY

The Hyperledger Indy is a distributed ledger for identity-based solutions that comes with reusable tools and libraries.

FEATURES

- Self-sovereignty
- User privacy
- Verifiable claims
- Zero-knowledge proofs
- No-hacks
- Full control over IDs

HYPERLEDGER IROHA

Hyperledger Iroha is a platform for easy integration into the enterprise environment.

FEATURES

- Easy to deploy and maintain
- Asset and identity management
- Modular and pluggable design
- Library for developers
- Role-based access
- Ledger queries

HYPERLEDGER SAWTOOTH

Sawtooth is actually a blockchain suit for developing, running, and creating new distributed ledgers.

FEATURES

- Smart contract applications
- Dynamic consensus
- Proof of Elapsed time consensus
- Transaction families
- EVM compatibility
- Parallel transactions

Hyperledger Fabric vs. Hyperledger Sawtooth: Hyperledger Sawtooth is another open source blockchain platform hosted by The Linux Found under the Hyperledger Project. Hyperledger Fabric and Hyperledger Sawtooth networks have differing governance capabilities and consensus algorithm

CHARACTERISTICS	HYPERLEDGER FABRIC	HYPERLEDGER SAWTOOTH
Permissions	Created specifically for permissioned networks.	Supports permissioned and permissionless networks.
Privacy and Network Governance	Provide complete data isolation between a set of participants. Strict network governance enabled by the Hyperledger Fabric certificate authority (CA), and channels.	All network peers have access to all transaction data. Does not provide channels or certificate authority (CA) capabilities.
Transaction Flow	Unique Execute-Order-Commit endorsement model where transactions are initially executed on a set of peers while ordering service handles packaging and delivery. Flexibility in defining set of required endorsers at the data level or contract level. This approach makes the framework more scalable and prevents nondeterminism in contract code.	Traditional Order-Execute-Commit flow. Sawtooth Validator handles transaction processing, ordering, and delivery.

Consensus Algorithms	Pluggable consensus algorithm allowing the orderer to be switched based on needs of the environment.	Uses a default “Proof-of-Elapsed Time (PoET)” algorithm, which is a Byzantine Fault consensus mechanism that relies on a specialized hardware component.
Smart Contract Language	Go, Java, Node.js	Go, Java, Python, Rust

- Ethereum can be either public or private without any permissions whereas Hyperledger is a private and permissioned network.
- This means that in Ethereum, anybody can participate in the network at any time. But **Hyperledger has a predefined community of participants, and access to the network is restricted only to them**.

Any questions so far?