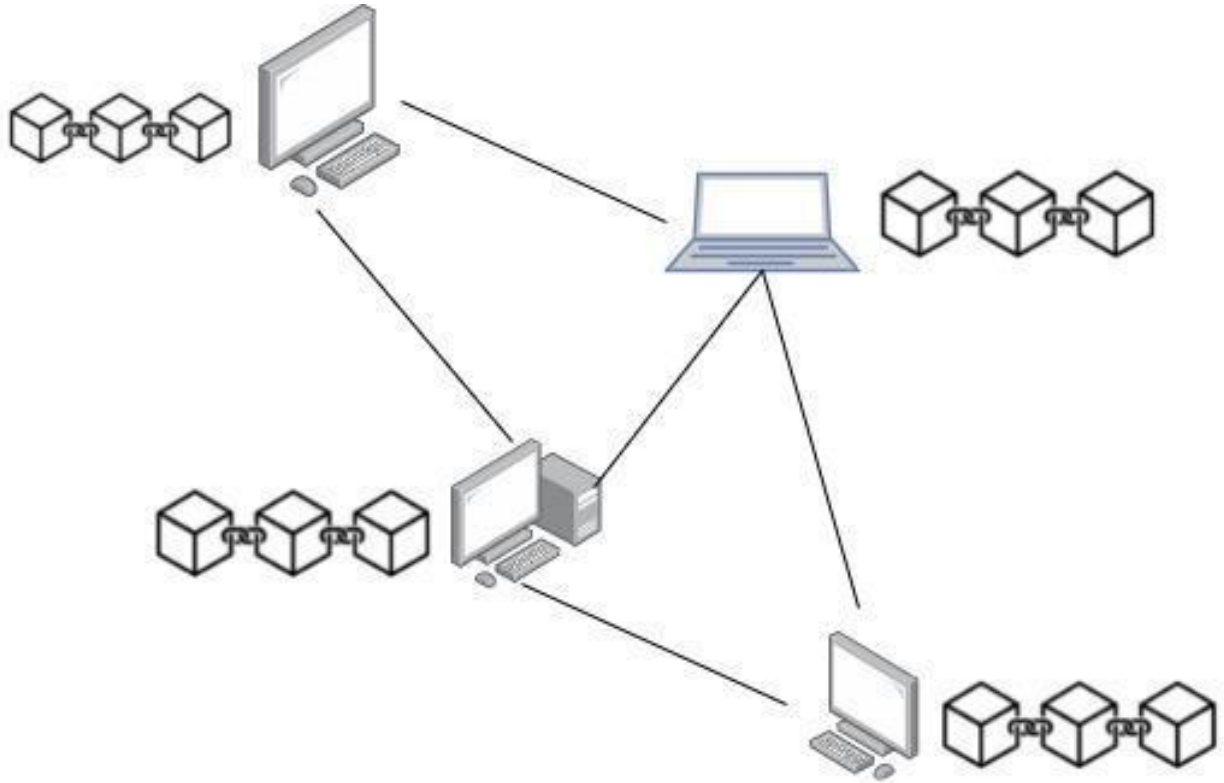
Hyperledger Fabric

Agenda

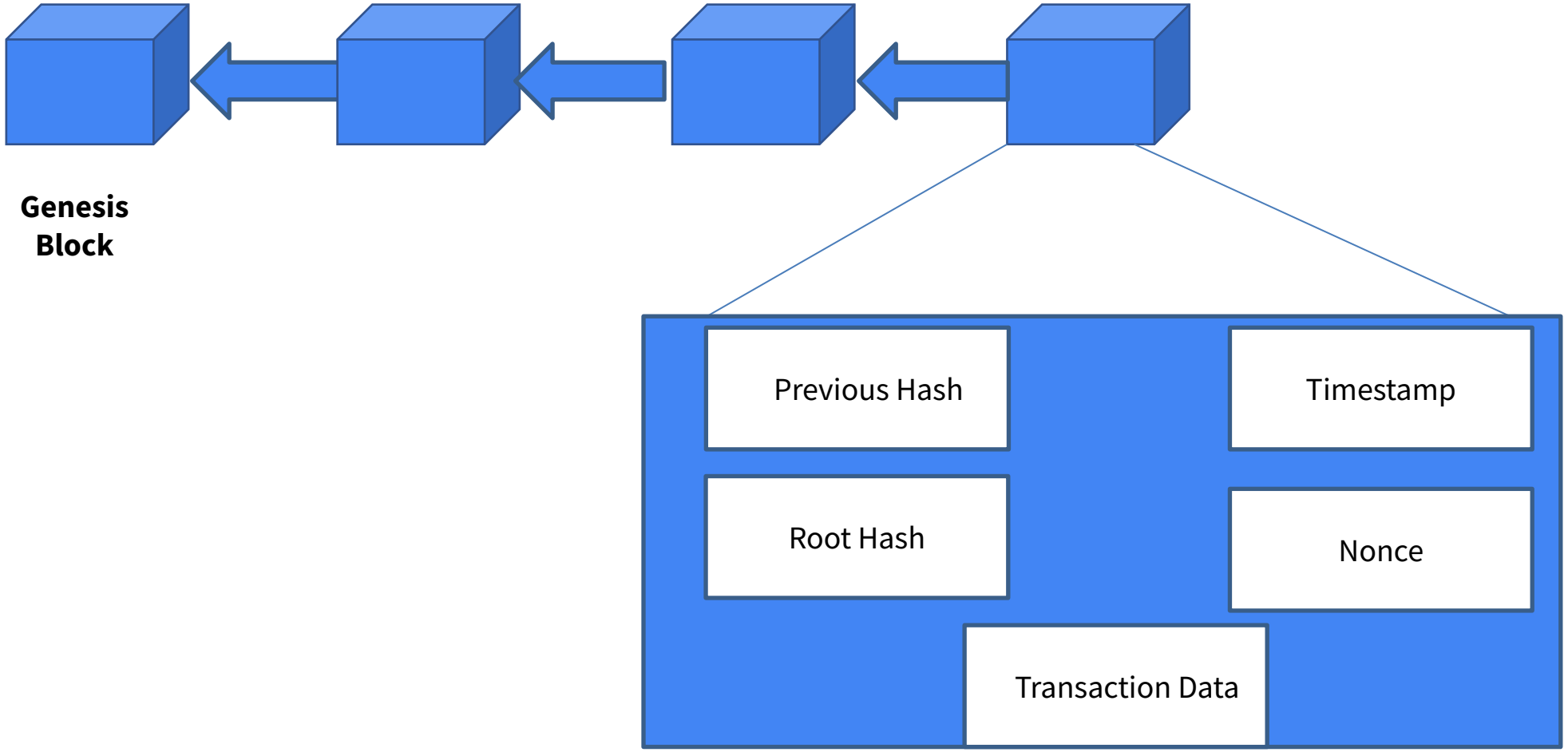
- Blockchain Overview
- Public Blockchain Vs Private Blockchain
- Features of Private Blockchain
- LF Decentralized Trust projects
- Hyperledger Fabric
  - Features of Hyperledger Fabric
  - Hyperledger Fabric Architecture
  - Transaction Flow

What is a Blockchain?

Blockchain technology is a **Distributed Ledger Technology**.



Blockchain


Genesis Block
Previous Hash
Timestamp
Root Hash
Nonce
Transaction Data

Public Blockchain	Private Blockchain
It is open and anyone will be able to access it.	Restrictions and permission mechanisms will be in place. Anyone who wants to join the network needs to get authorized
Each node will have equal privileges for a transaction and data	Only limited nodes or certain types of nodes can perform a transaction
Transaction-per-second (tps) is low	Will have very high Tps
Transaction cost is high	Transaction cost is comparatively very low
Uses consensus protocols like proof-of-work, proof-of-stake.	Uses consensus algorithms like Raft, smartBFT etc..
Requires no trust among members inside the network	Members inside the network need to trust each other.
Energy Consumption is very high	Energy consumption is too low.

LF Decentralized Trust

- The open source foundation for decentralized technologies and ecosystems
- Hyperledger Foundation is now a part of LF Decentralized Trust

LF Decentralized Trust projects

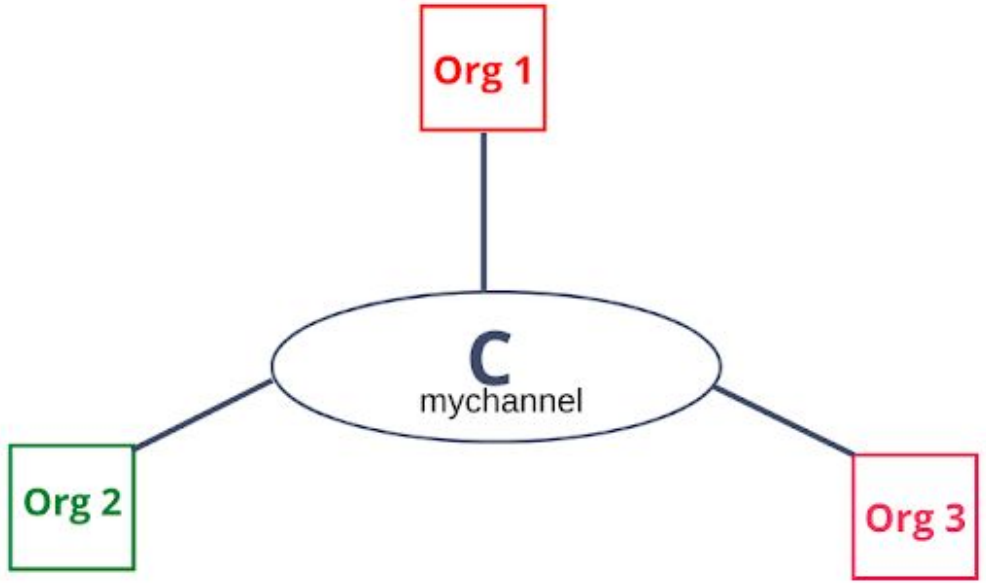

Hyperledger
FABRIC

Salient Features of Hyperledger Fabric

- Permissioned blockchain network
- Smart contract can be written in standard programming languages(Go, NodeJS, Java)
- No cryptocurrency and No mining
- Pluggable Architecture
- Privacy and confidentiality of transactions - Channels and PDC

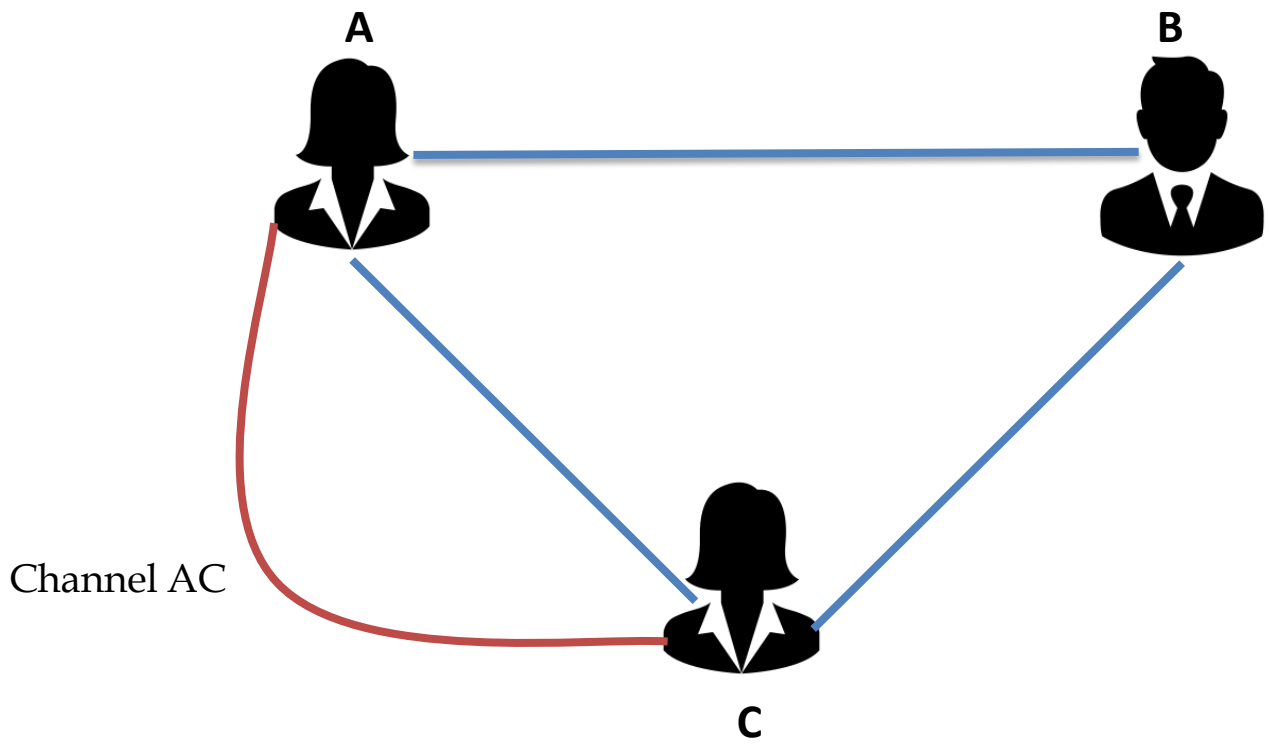
Channel

- A communication pathway between members of a Hyperledger Fabric network
- Transaction on a Hyperledger Fabric network occurs on a channel
- Multiple channels can be created in a network with the same or different participants.
- Each channel will have its own rules and help the participants to transact privately.



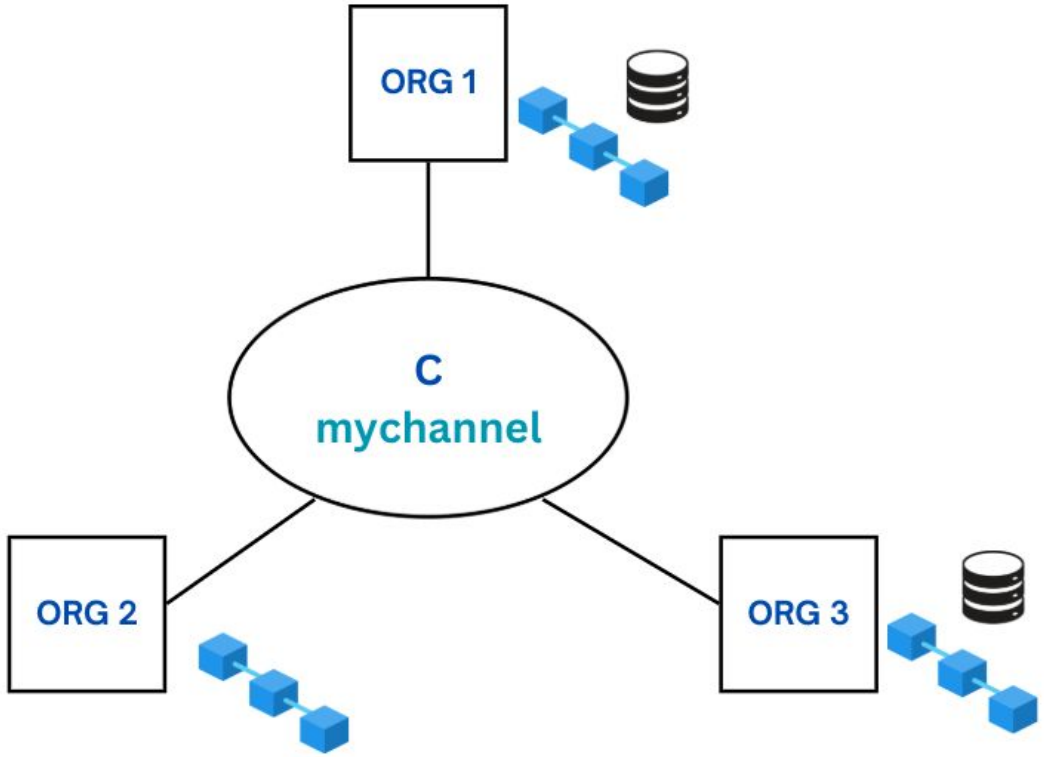
Multiple Channel

- Hyperledger Fabric can create multiple channels
- Let A, B, and C be participants in a DLT based business network application
- A and C can conduct a confidential transaction by creating a separate channel between them



Private Data Collection (PDC)

- Ensures privacy by sharing sensitive data only with specific organizations in a channel.
- Data stored off-chain; only hashes are recorded on the ledger.
- Ideal for sensitive use cases like finance and supply chains.



Components of Hyperledger Fabric

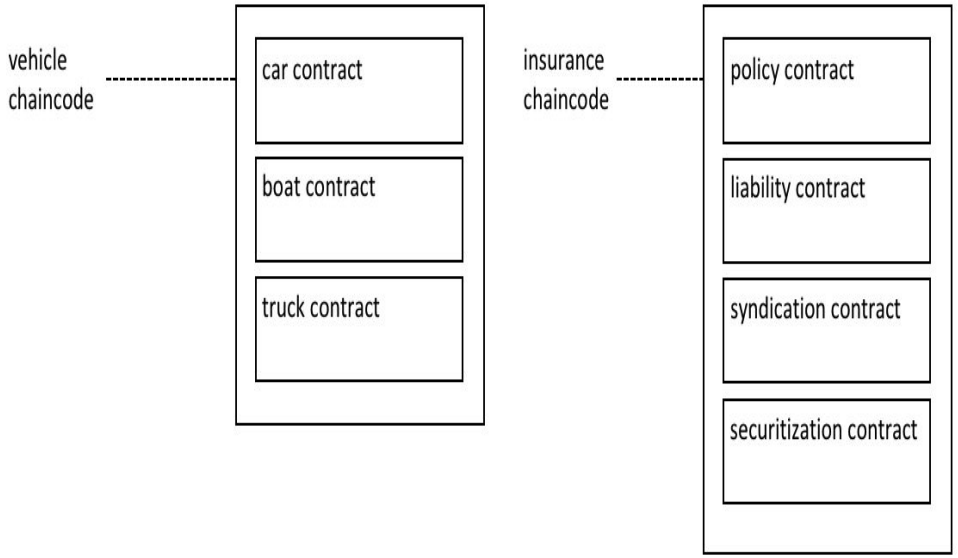
- Certificate Authority
- Peers
- Ledger
- Smart contract
- Orderer
- Channel

Certificate Authority (CA)

- Manages identities for organizations and users in the network
- Issues certificates for authentication and transaction signing
- The CA issues digital certificates following the **X.509** standard
- Handles certificate renewal and revocation.
- **Membership Service Provider (MSP)** - set of folders that are added to the configuration of the network and is used to define an organization
- Certificates can be generated using **Fabric CA** and **Cryptogen**

Smart Contract

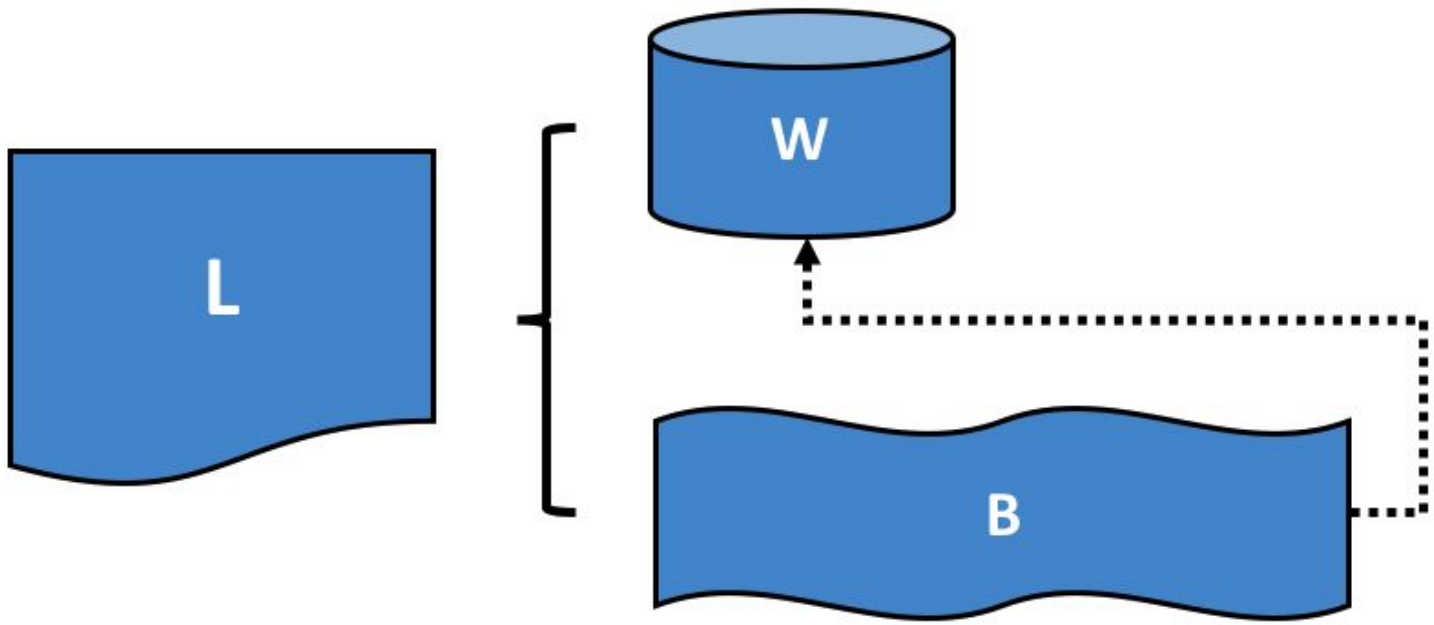
- A smart contract defines the **transaction logic.**
- A smart contract is defined within a **chaincode**.
- Multiple smart contracts can be defined within the same chaincode.
- When a chaincode is deployed, all smart contracts within it are made available to applications.



Fabric Ledger

- A ledger is a sequenced, tamper-resistant record of all state transitions
- A separate ledger is maintained for each channel
- The Fabric ledger consists of **blockchain** and **world state**
- **Blockchain** contains the **transaction log** which is immutable
- **World state** contains the **current state** of the asset
- Channel configurations, Transactions etc are written to the blockchain
- The worldstate can be either LevelDB (default) or CouchDB
  -**LevelDB** is a simple key/value store
  -**CouchDB** is a document store that allows complex queries

Ledger : World State and Blockchain



L	Ledger
W	World State
B	Blockchain
L: B, W	L comprises B and W
W ◄ B	B determines W

Peer

- A network entity(node) that maintains a ledger which consists of the Blockchain(Transaction Log) and World State
- Some peers run chaincode to perform read/write operations to the ledger
- Each organization in the network maintains one or more peers
- Peer has two roles Endorser and Committer

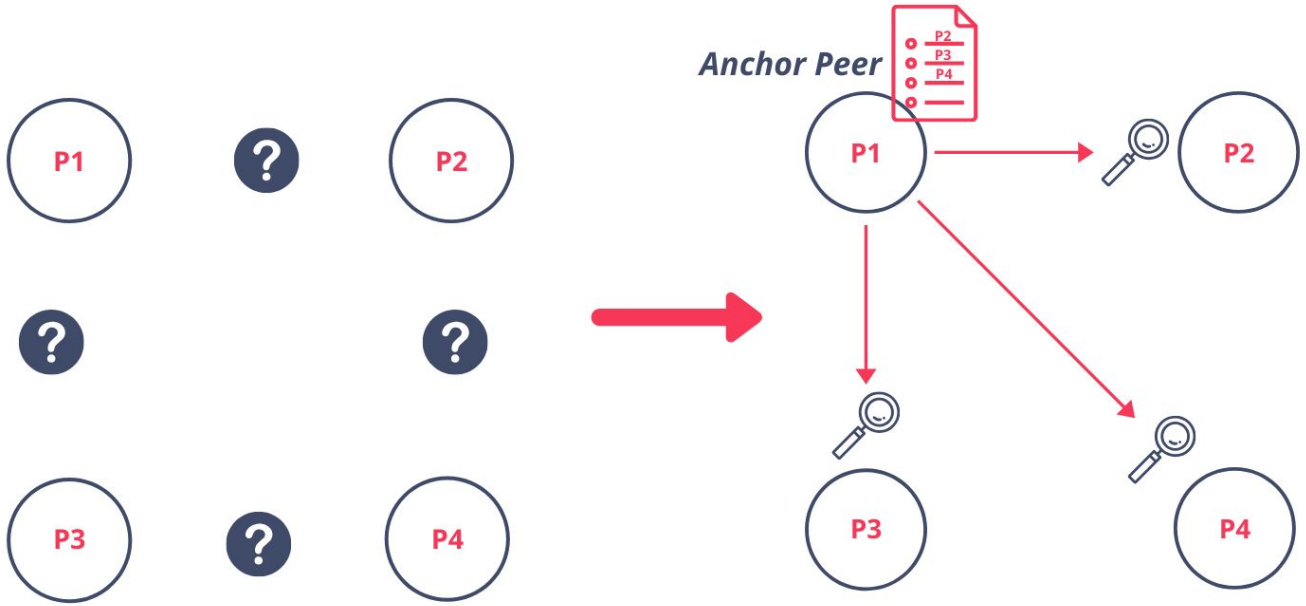
Types of Peers

Committing Peer: Maintains blockchain and world state. Commits transactions. May hold smart contract(Chaincode)

Endorsing Peer: Specialised peers that endorses transactions by receiving a transaction proposal and respond by granting or denying endorsement.
Must hold chaincode

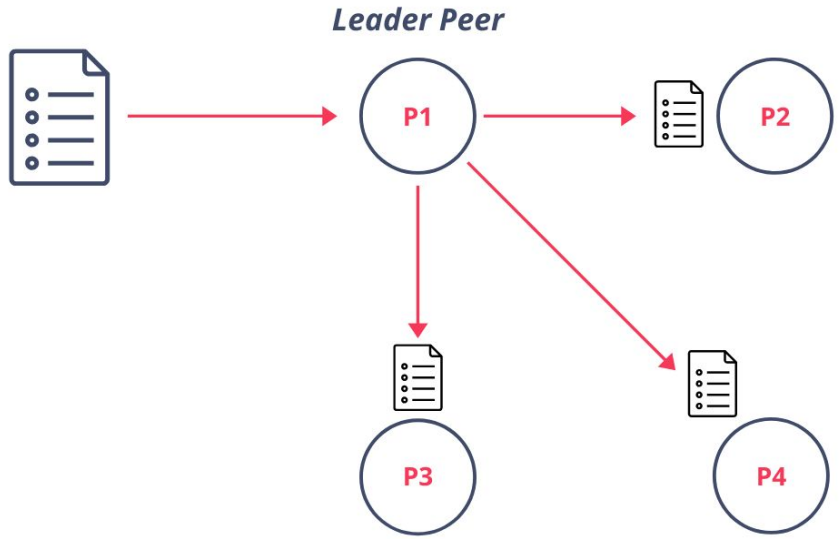
Anchor peer

- perform cross-organisation communication scenarios and it is defined in the channel configuration.



Leader Peer

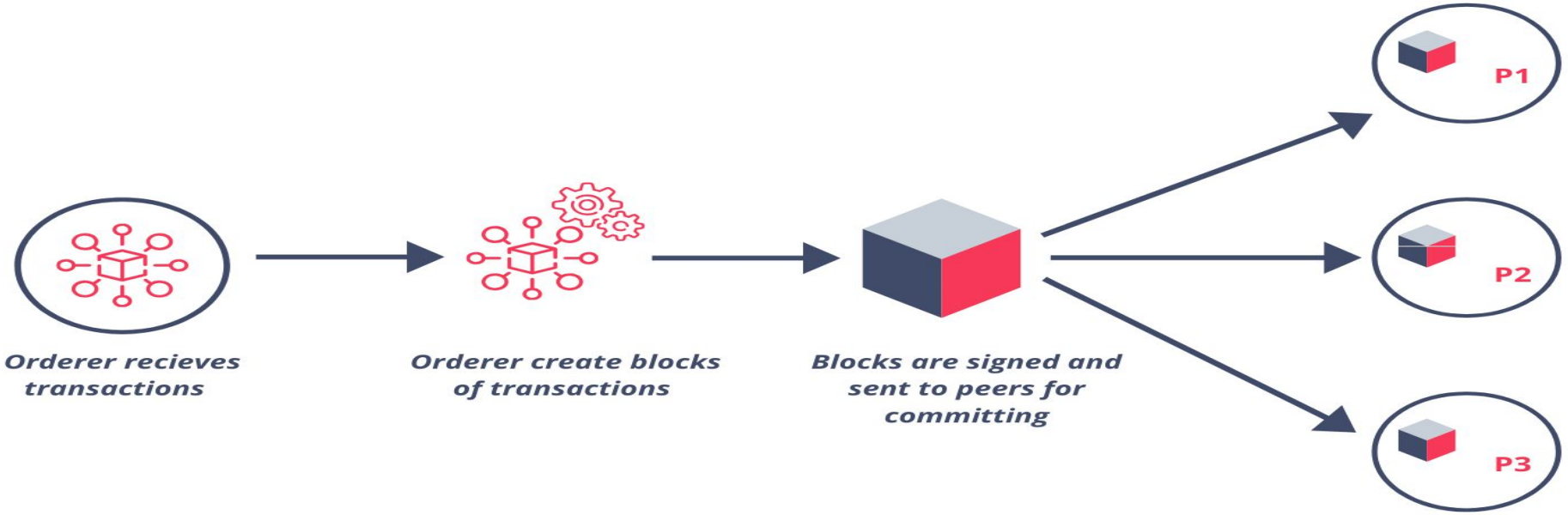
- sends the transactions to other **committing peers** in the organization.



Ordering Services

The ordering service packages transactions into blocks to be delivered to peers.

- Orderer Nodes receive transactions from many different application clients concurrently.
- Arrange batches of submitted transactions into a well-defined sequence and package them into ***blocks***.These blocks will become the *blocks* of the blockchain.
- Orderer Node also holds the ledger

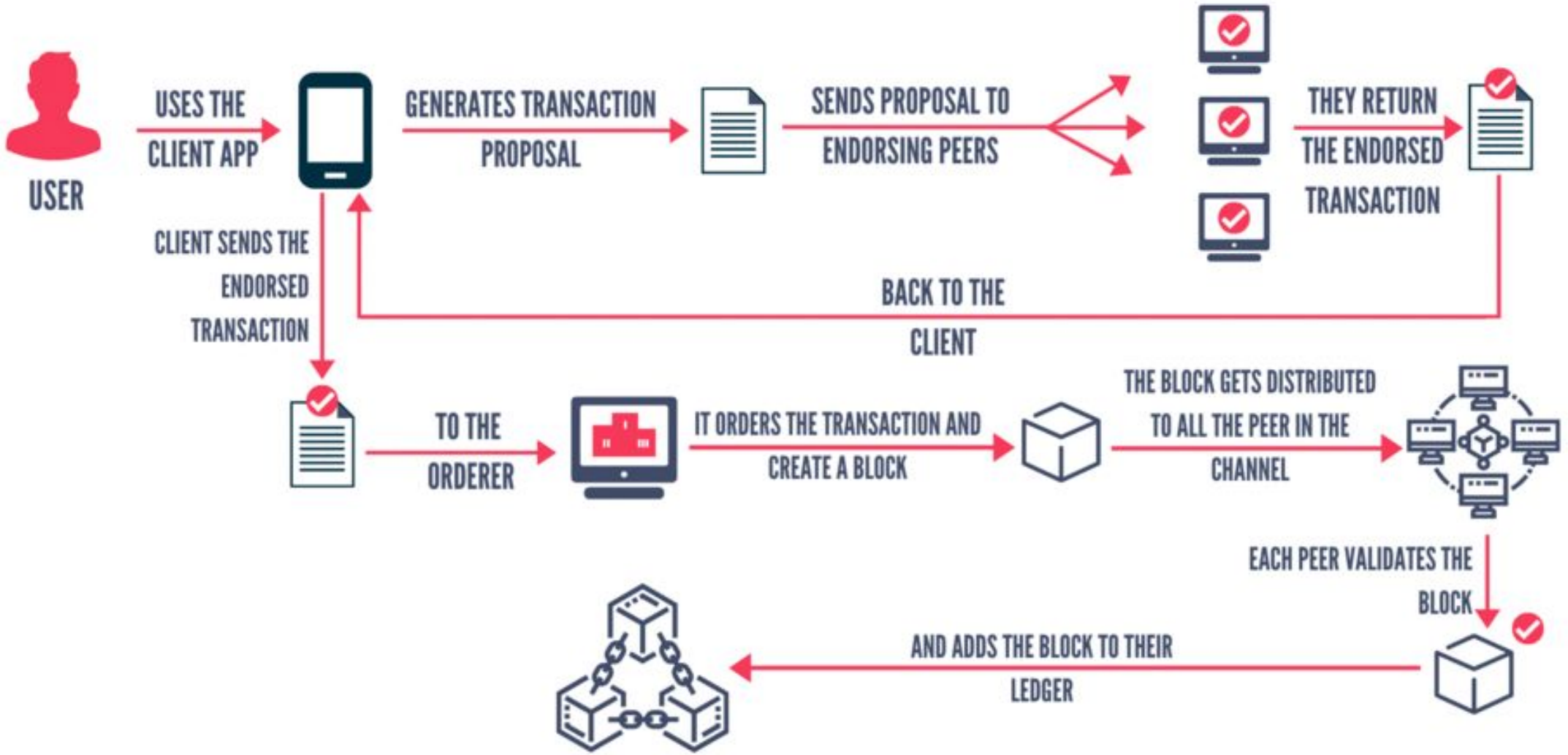


Endorsement Policy

- Every chaincode has an endorsement policy.
- Endorsement policies define the smallest set of organizations that are required to endorse(approve) a transaction in order for it to be valid.
- To endorse, an organization's endorsing peer needs to run the smart contract associated with the transaction and sign its outcome.
- When the ordering service sends the transaction to the committing peers, they will each individually check whether the endorsements in the transaction fulfill the endorsement policy.

Eg: AND('Org1.member', 'Org2.member', 'Org3.member')

Transaction Flow



Steps in Transaction Flow

1. Client generates Transaction Proposal.
2. Submit Transaction Proposal.
3. Endorsing.
4. Sending to Orderer.
5. Ordering and Block Distribution.
6. Updating the Ledger.
7. Notify the client.

Client generates Transaction Proposal

The client generates a transaction proposal message.

The proposal has the following attributes:

1. **ClientID** - Identifies the client
2. **ChaincodeID** - Identifies the Chaincode to be invoked
3. **Transaction Payload** - It contains the functions to be invoked and the arguments

The client then **signs** the proposal message with the private key.

Submits Transaction Proposal

Client Submits the Transaction Proposal

- The Client Node / Application Node submits the transaction proposal to the Endorsing peer for approval.
- The Endorsement is done based on how the Endorsing policy defined. Based on the policy, the proposal is submitted to Endorsing peers

Endorsing

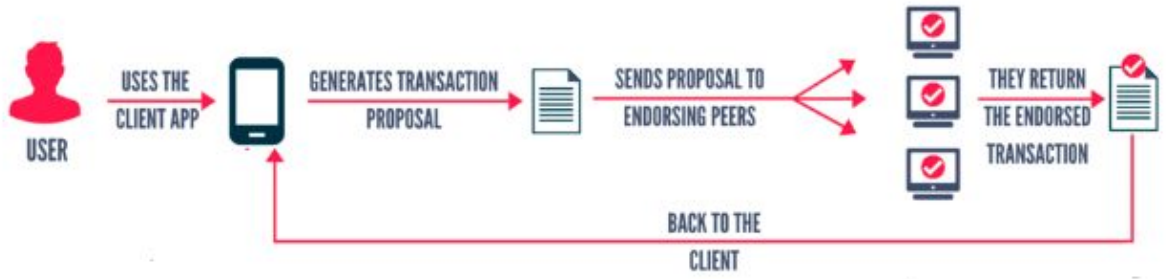
Once the **Endorsing peers(EP)** receive the proposal, they start performing the endorsing process.

If the signature is valid, then the EP will simulate the transaction execution. Simulation results in the creation of the **Read/Write set.**

The signing of the proposal results in the creation of **Transaction Proposal Response - Endorsed** message.

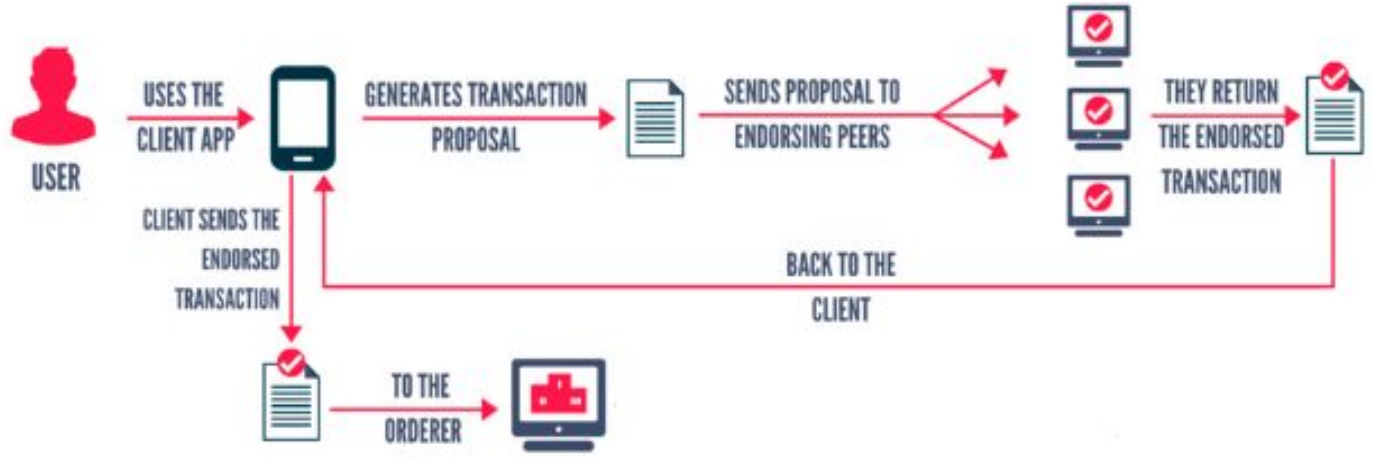
The response has the following attributes:

1. **Endorsing peer's signature**
2. **Read Set** - State value of the key used for the simulation.
3. **Write Set** - State value of the key after executing the logic



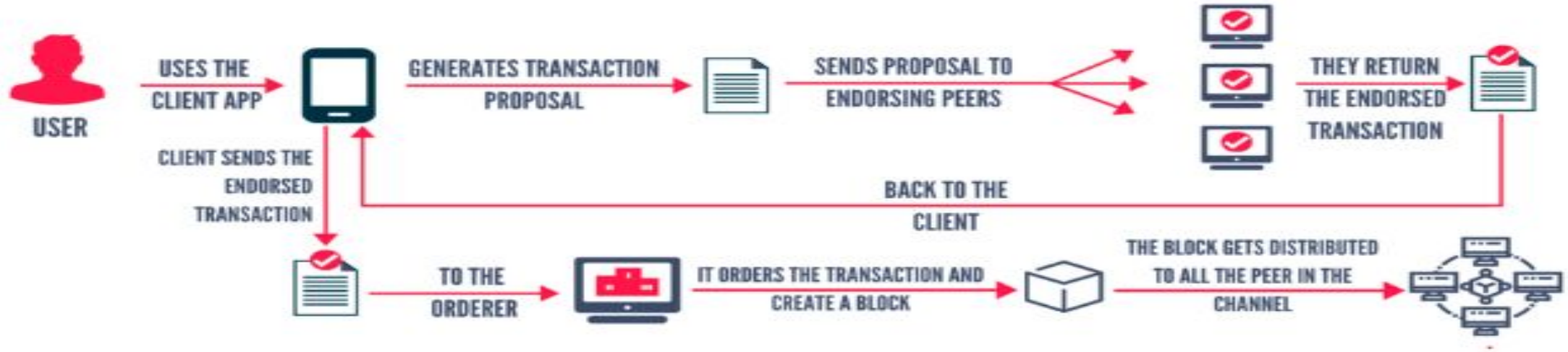
Sending to orderer

- Client verifies the signature of the endorsing peers by comparing the transaction proposal response with the signature of the endorsing peer.
- The transaction may get rejected if the EP signature is not valid or there is a mismatch in proposal responses.
- If everything is fine, then the client sends the transaction to the orderer.



Ordering and Block Distribution

- The Orderer receives the transactions in a group and orders them according to the sequence it received.
- It then creates a block of transactions based on parameters like block size or block timeout whichever is earlier.
- Created blocks are broadcasted to all the peers in the channel.



Updating Ledger

Peer performs its own validation on the blocks received from the orderer.

The process involves:

1. Validating the Endorser and client signature in the transaction done in endorsing step
2. Verifies that the transaction does not violate the current state

If the validation is successful, then the data in the block is added to the ledger marked as "**Valid**", and world state is updated. If unsuccessful, it is marked as "**Invalid**" and added to the transaction log.

Notify the client

- The client will be notified that the transaction is completed by using events.
- The client records it as a successful or unsuccessful transaction.

Consensus

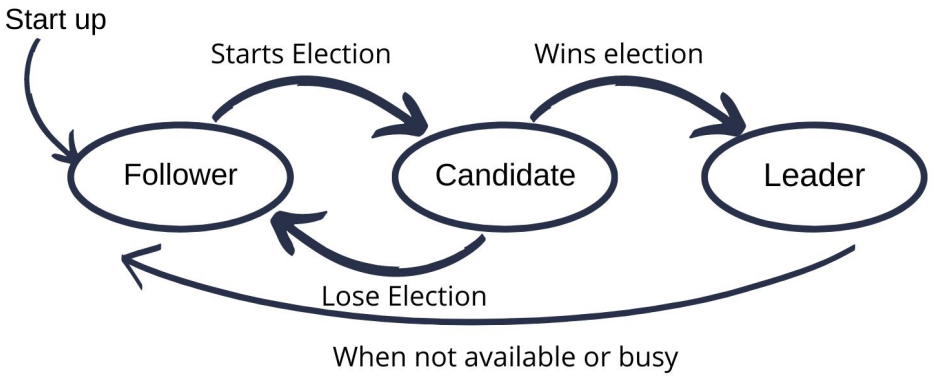
- A way of coming to an agreement
- Integral to a decentralized system
- Participants may or may not trust each other
- Agreement on common principles of functioning

- In Hyperledger Fabric:
  - Raft
  - SmartBFT
  - Solo and Kafka (deprecated)

Consensus - Raft

RAFT

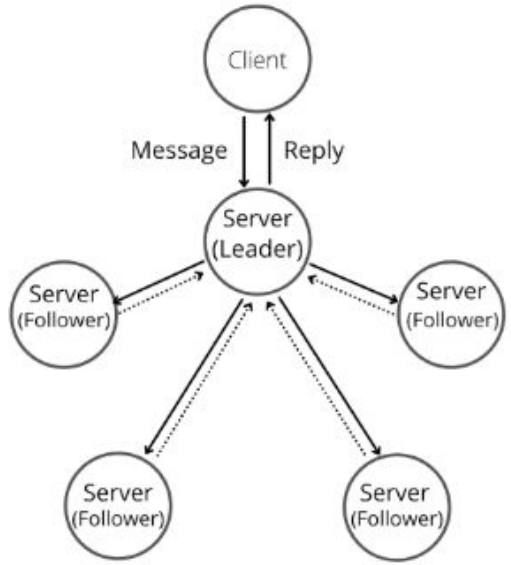
Raft is a distributed crash fault tolerance algorithm for consensus



[Visualise the leader election process]

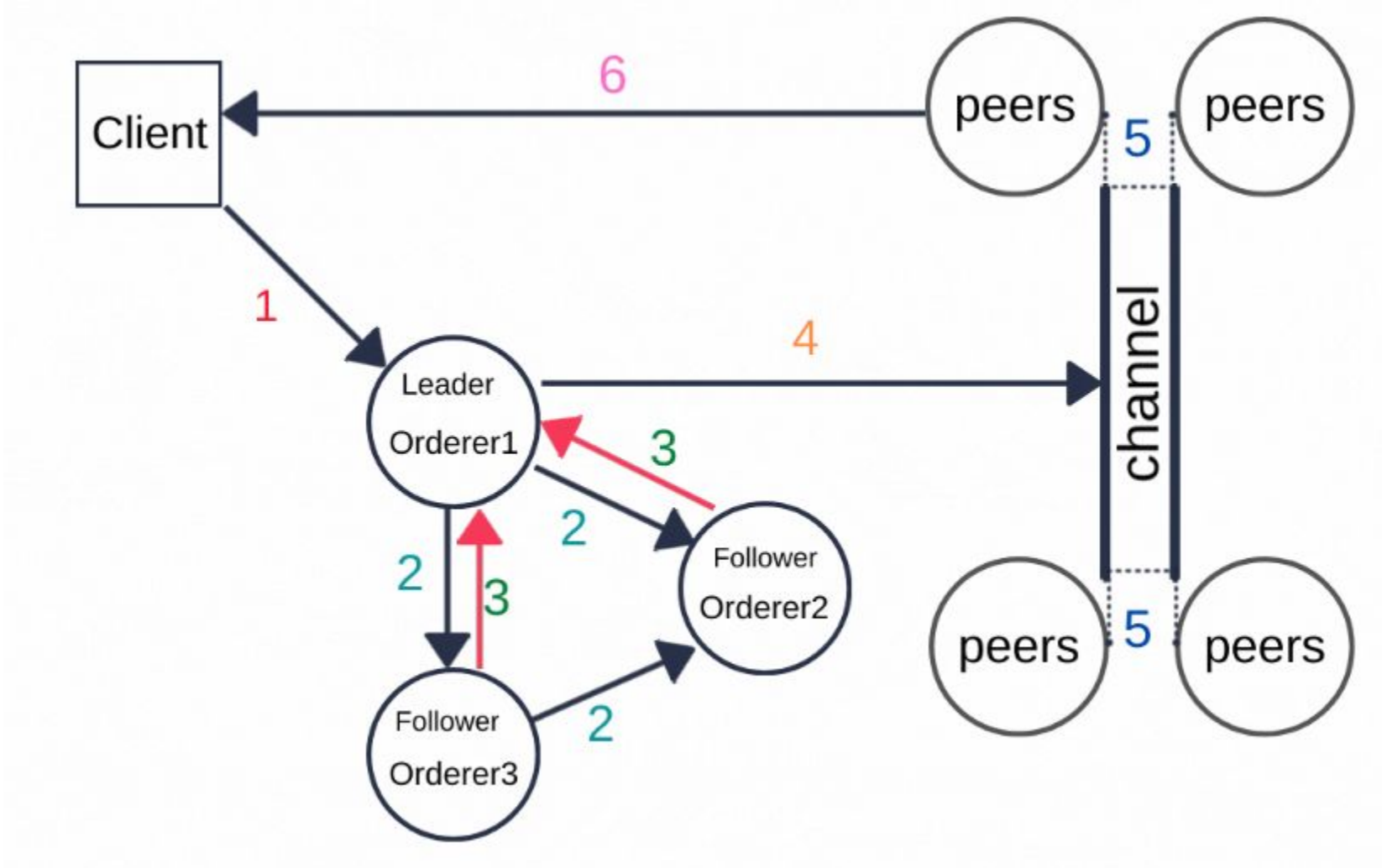
Working

- Step1 : Client sends a request which consists of a command to the leader node.
- Step2 : Leader node appends a new request and sends the same to follower nodes. Follower nodes also add this new request on their respective logs and revert with a confirmation message.
- Step 3 : Once it receives the majority of confirmation messages, the leader will commit its log to its state machine which produces some output as per the request.
- Step4 : Once the leader commits log, followers will also commit log onto their State Machine.
- Step 5 : Response will be sent back to the client.



RAFT in HYPERLEDGER FABRIC

RAFT in HYPERLEDGER FABRIC



RAFT In Hyperledger Fabric

Step 1: An orderer node approves the inclusion of transaction into blocks.

Step 2: If this orderer is the leader, it logs the block as an uncommitted log entry and then broadcasts this block to the follower orderers.

Step 3: Follower orderers, upon receiving the uncommitted block, stores it and respond to the leader with an acknowledgment.

Step 4: Once the leader orderer receives acknowledgments from a majority of the nodes, it commits the block to its ledger.

Step 5: After the leader has committed the block, it informs the follower orderers, and they too commit the block to their respective ledgers.

Note: In Raft, at least three nodes are required to maintain fault tolerance and ensure consensus

THANK YOU