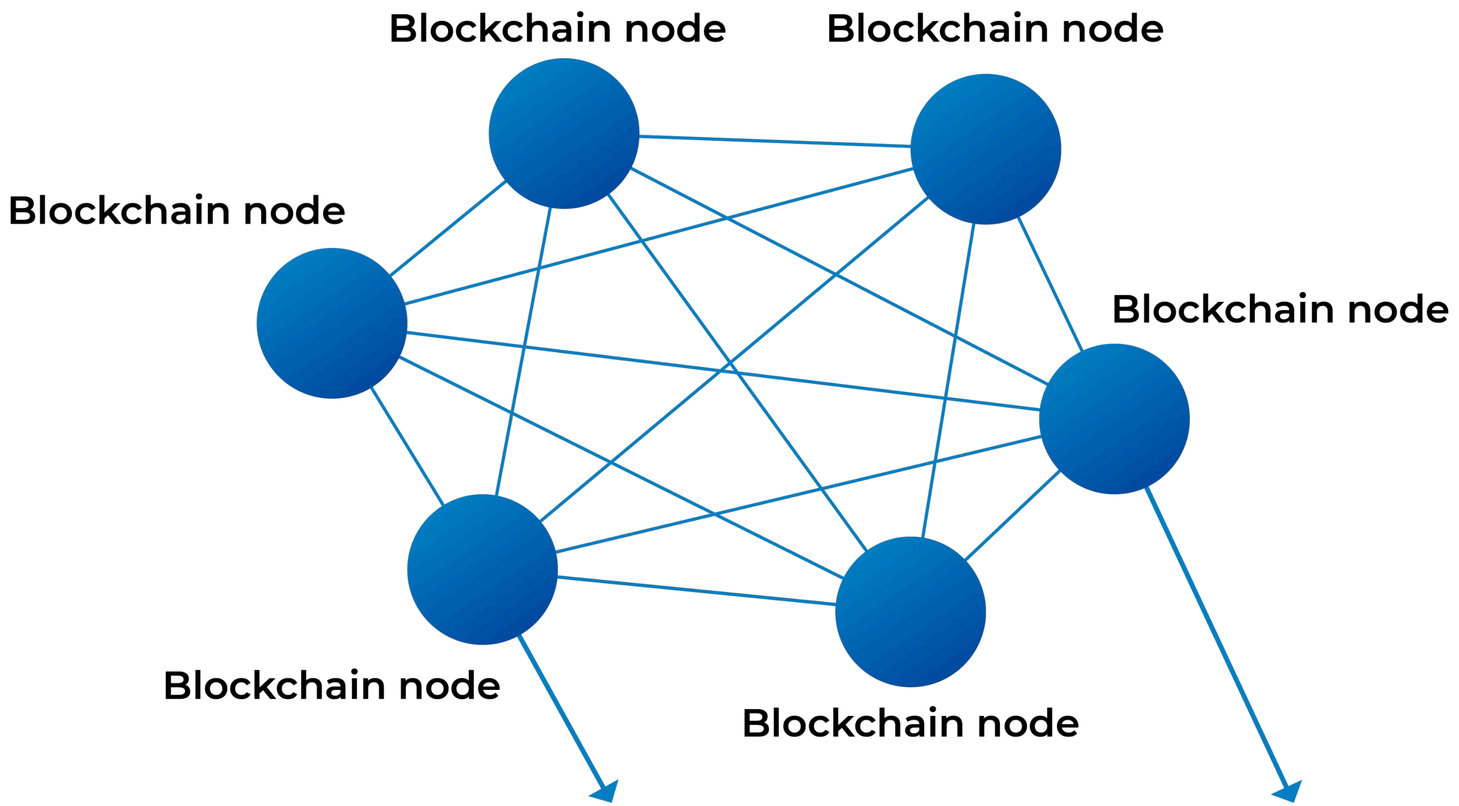
Building an Immutable Ledger for POS Payments

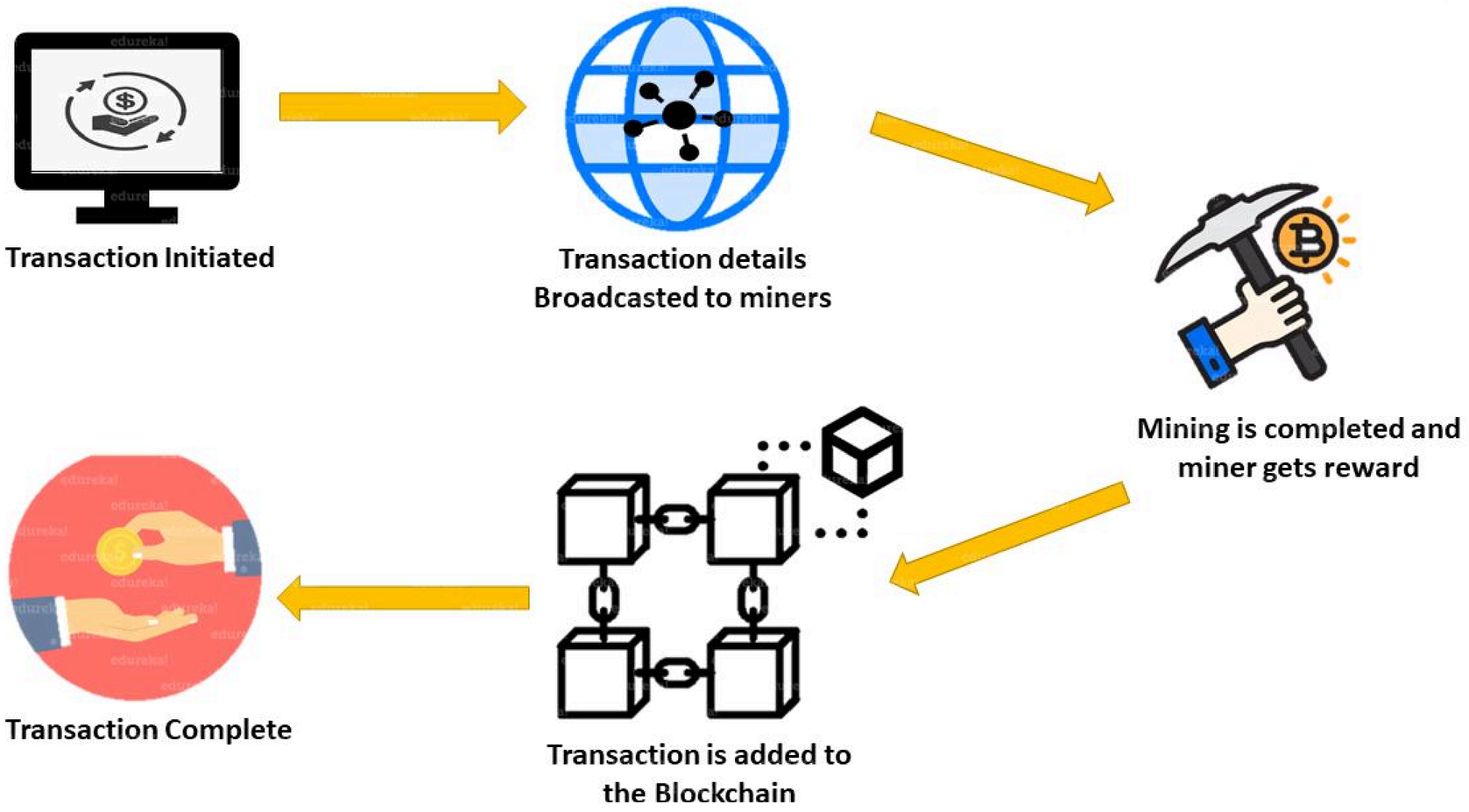
Strategic Roadmap for Integrating DLT into the Internal Wallet & Database System

Blockchain Network



Blockchain Ledger

Transaction



Consensus Mechanisms

PoW

Proof of work (mining), relies on miners. Advanced computing is used to find out the result of complex mathematical puzzles

PoS

Proof of Stake, leverages validators approach. Participants stake their assets on the blockchain. Validator is randomly selected.

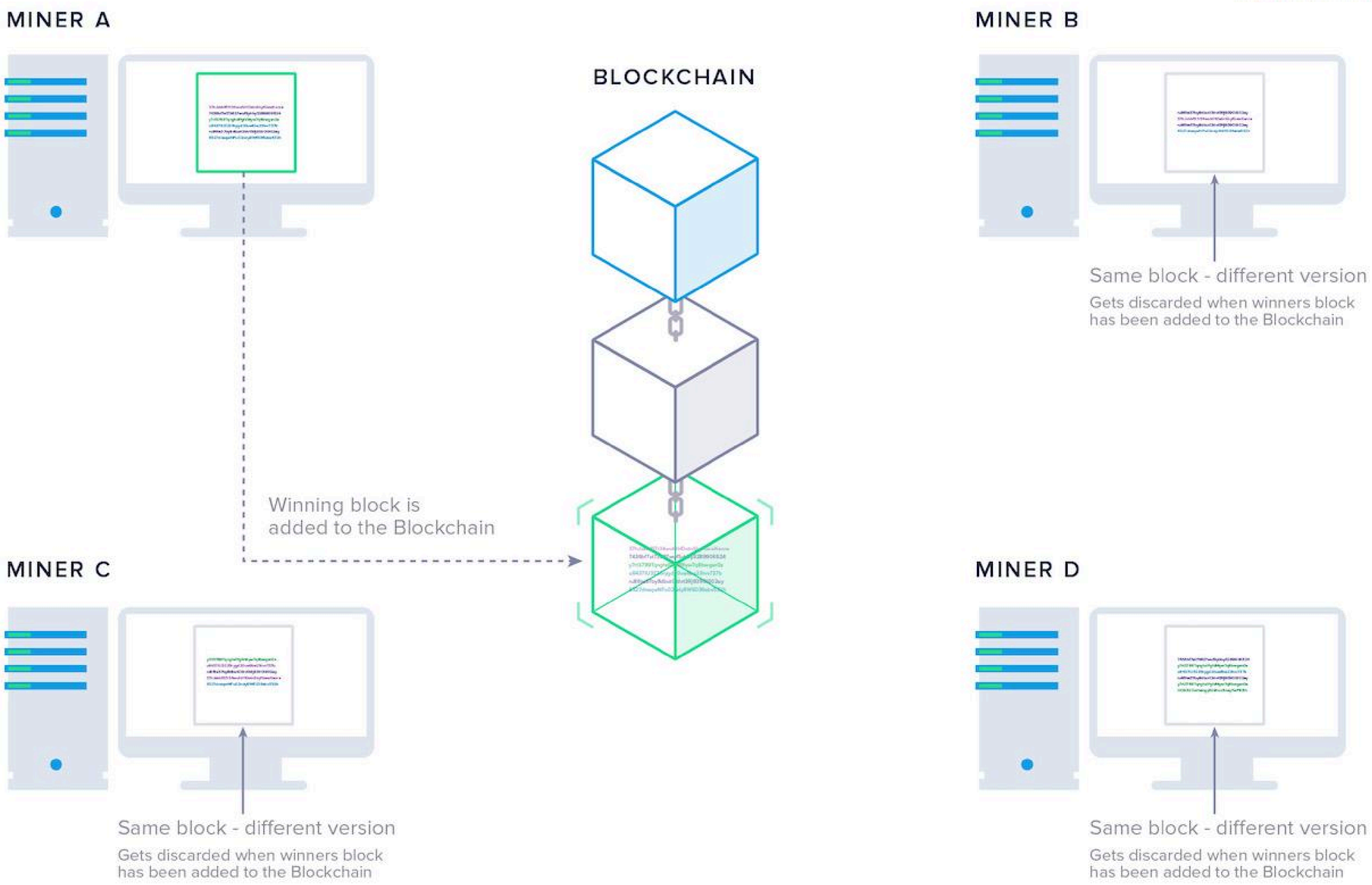
PoA

Proof of Authority, focuses on verification of validators identity. The validators are required to maintain their reputation to earn the authority of validation.

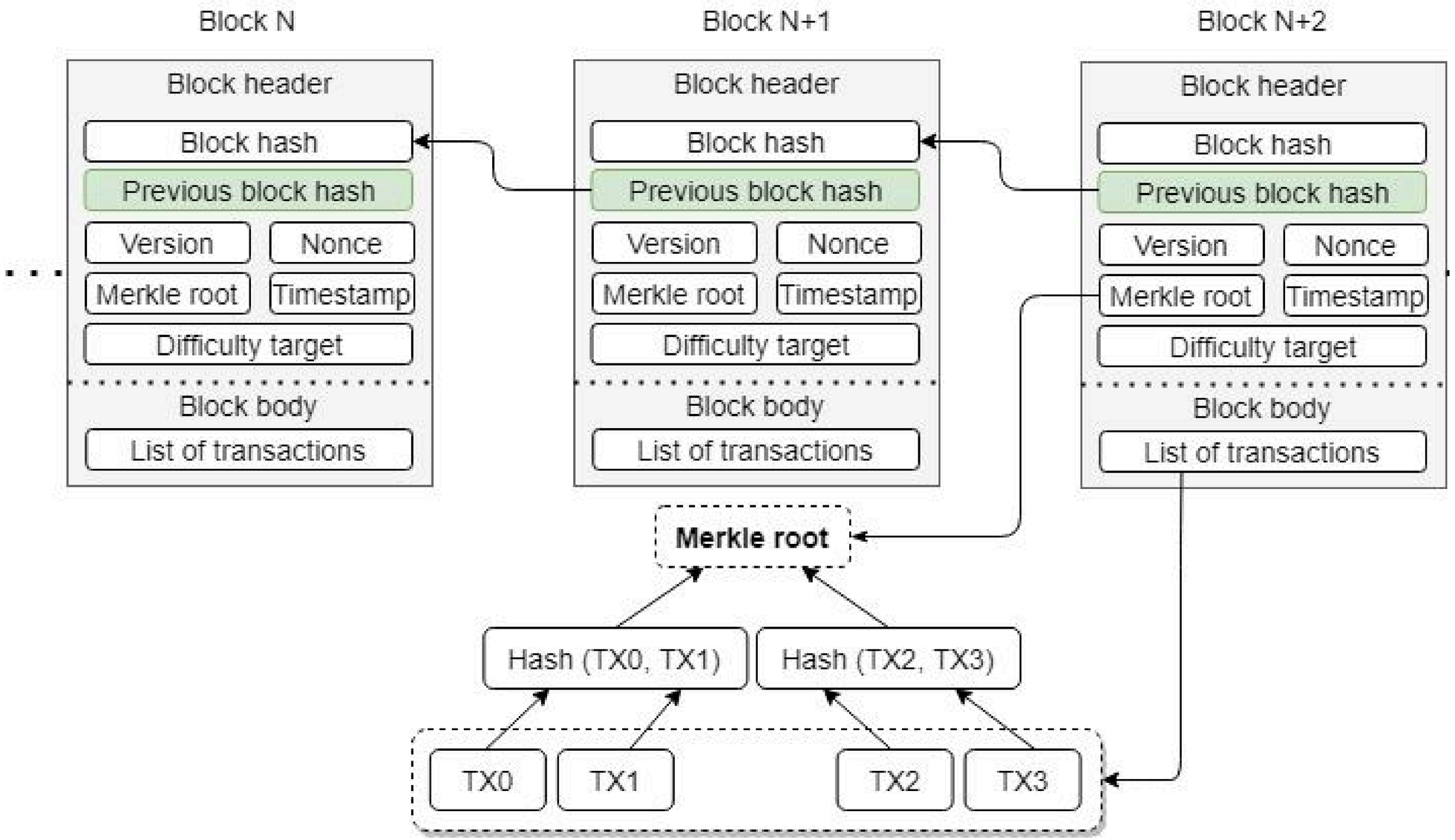
BFT

Byzantine Fault Tolerance. A fault tolerance mechanism to avoid disruption when the network has some suspicious participants.

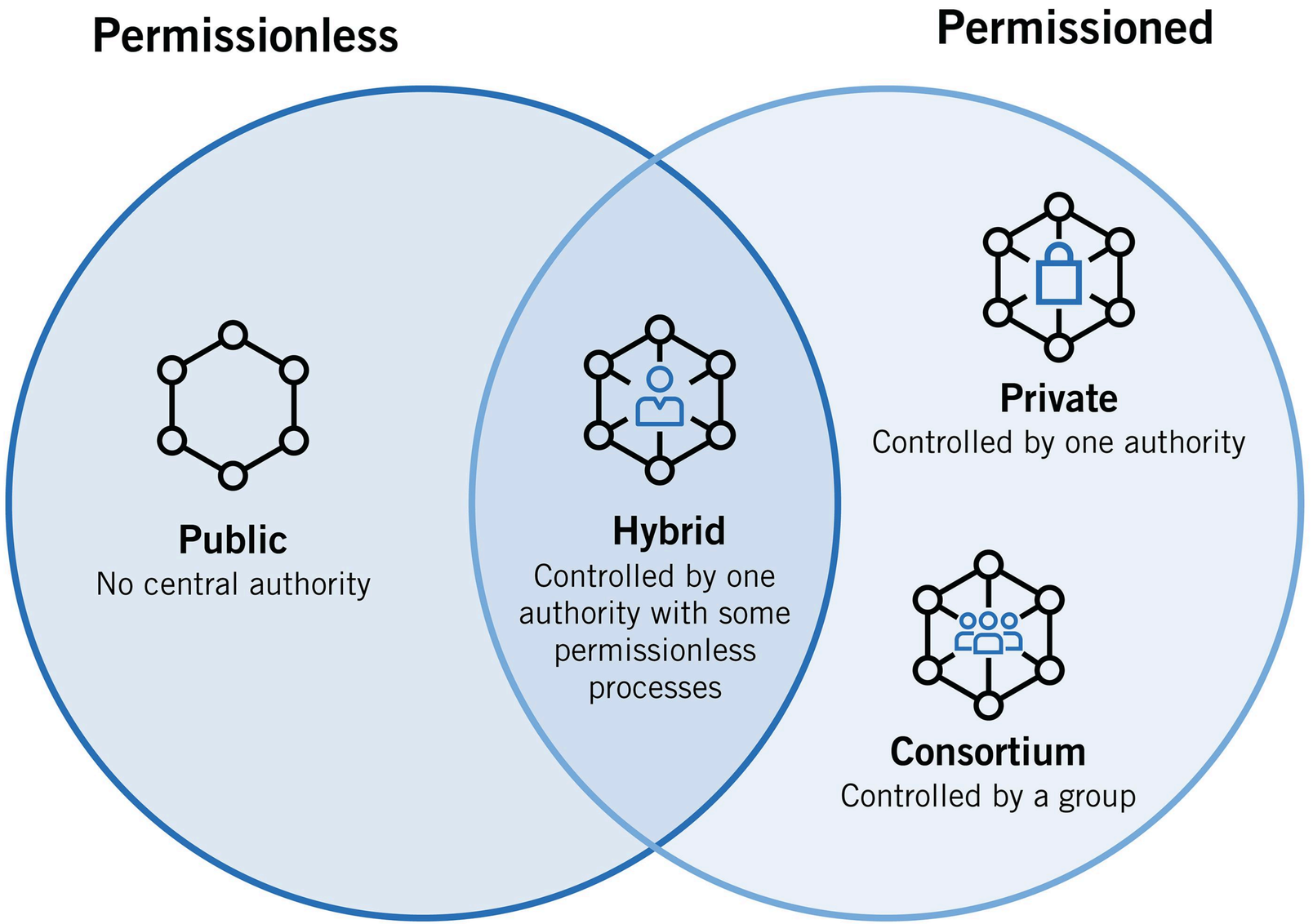
New Block


MINER A
BLOCKCHAIN
MINER B
Same block - different version
Gets discarded when winners block has been added to the Blockchain
Winning block is added to the Blockchain
MINER C
Same block - different version
Gets discarded when winners block has been added to the Blockchain
MINER D
Same block - different version
Gets discarded when winners block has been added to the Blockchain

Block Structure



Blockchain Types



Our Need

1. Data Integrity and Immutability
2. Security and Availability
3. Trust and Auditability
4. Logic Fragmentation

Solution

HYPERLEDGER
FABRIC

Financial Suitability

It prioritizes governance, privacy and control, which made it suitable for financial transactions. Also it requires zero gas fees.

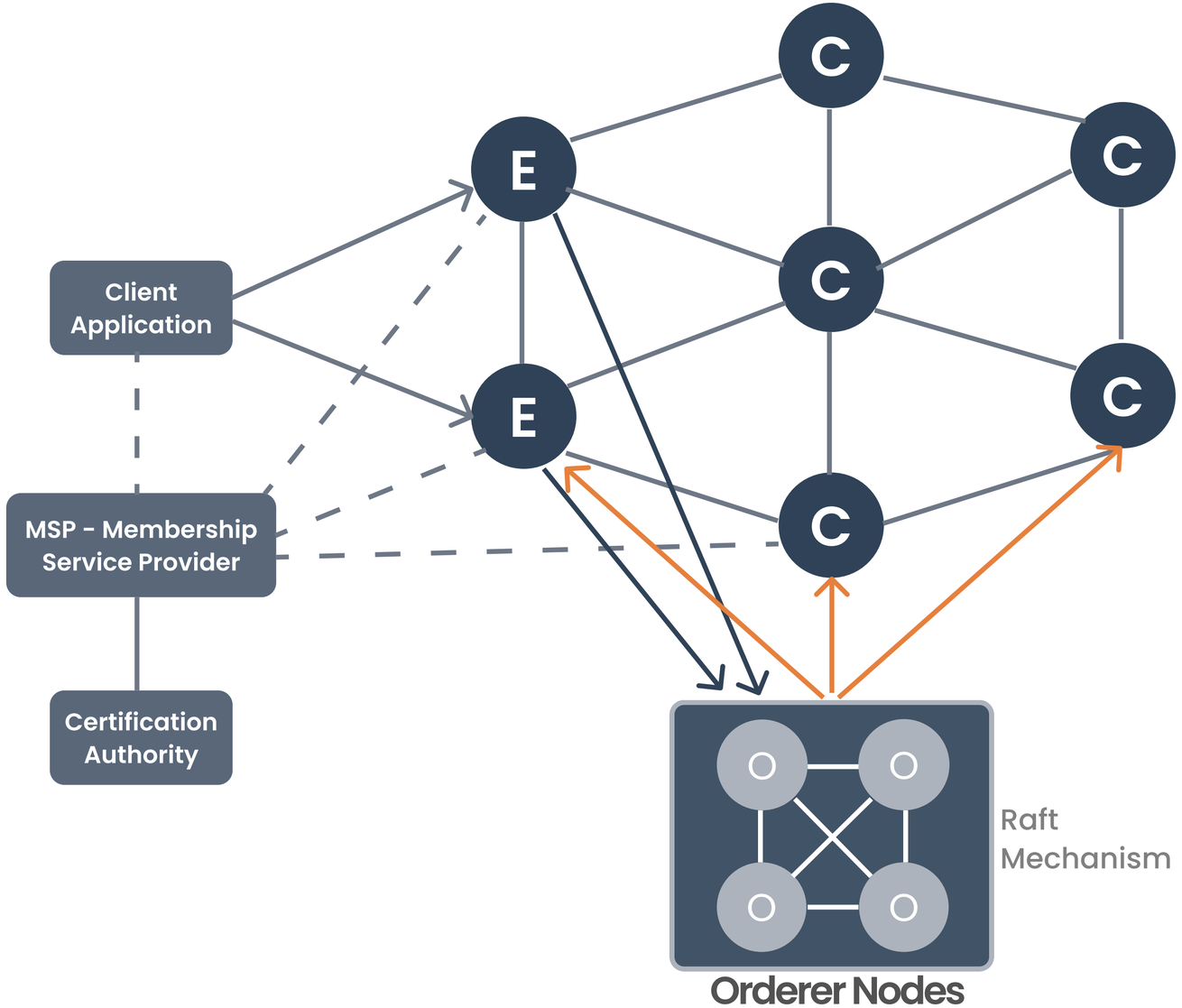
Security by Design

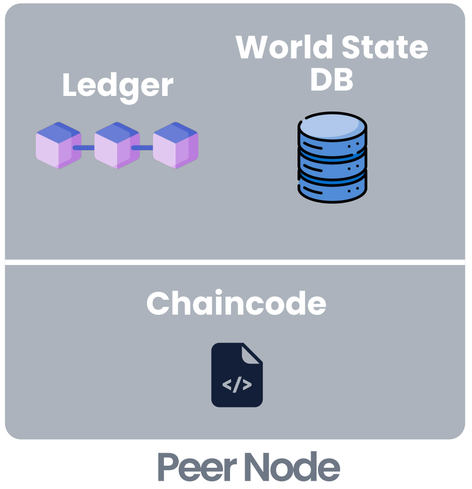
Build in identity mechanism via CA and MSP and access control via endorsement policies

Modularity & Scalability

It’s modular structure allows us to integrate peers and orderers as per our need.

Hyperledger Fabric Architecture





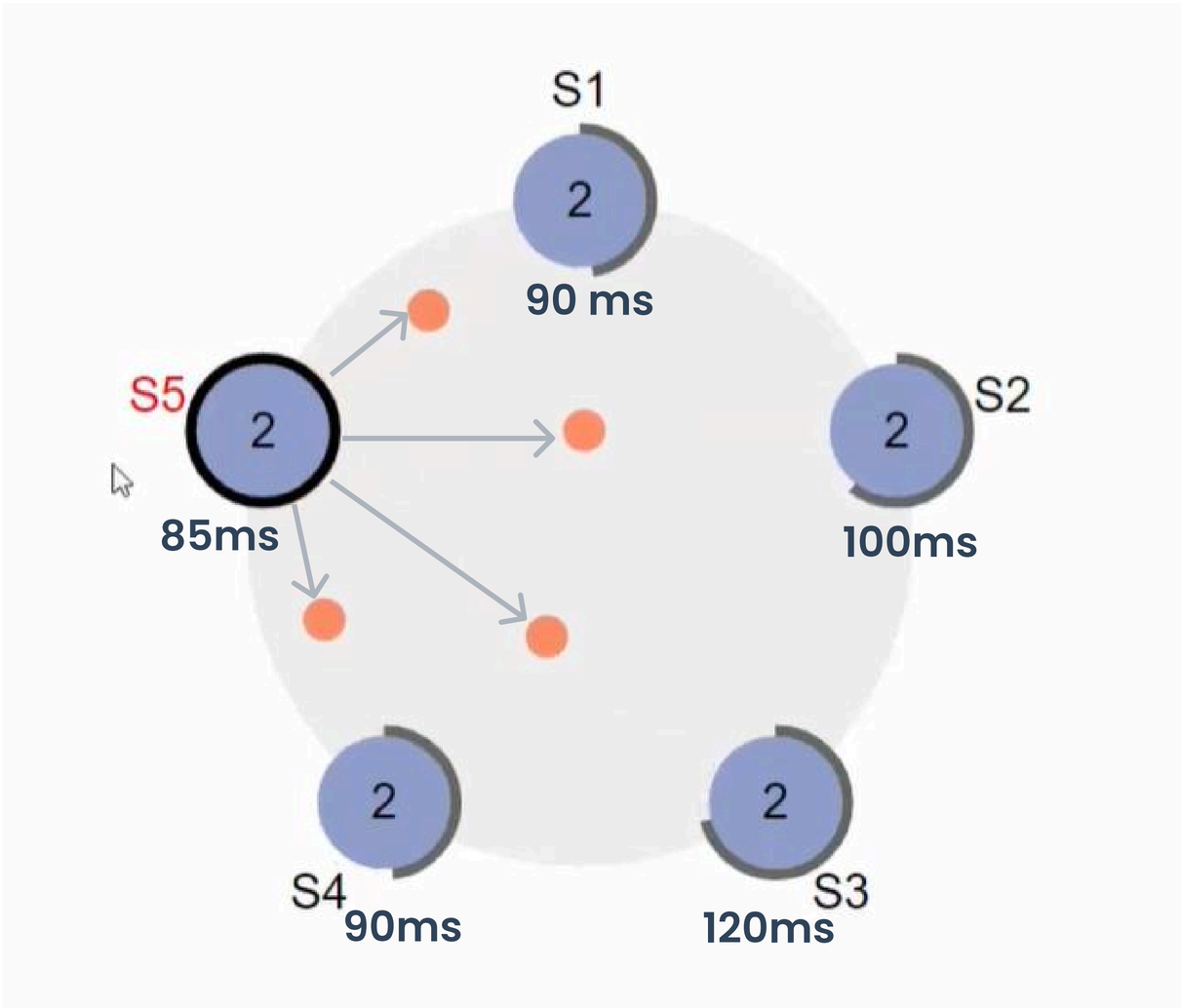
E - Endorser Node
C - Committer Node
O - Orderer Node

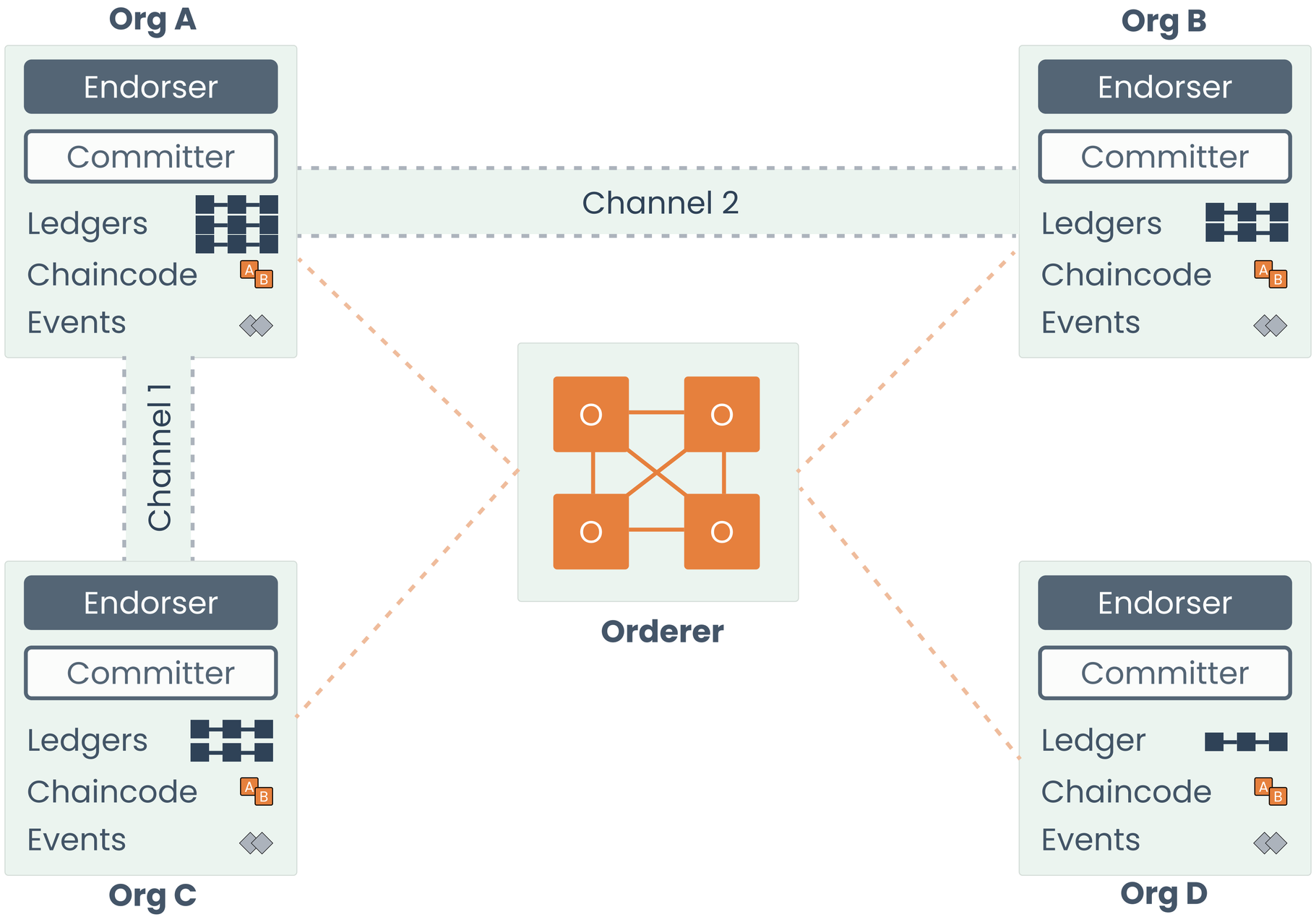
Consensus Mechanism

Ralf

Crash Fault Tolerance - CFT model

Leader Follower Model




Org A
Endorser
Committer
Ledgers
Chaincode
Events
Org B
Endorser
Committer
Ledgers
Chaincode
Events
Channel 2
Channel 1
Orderer
Endorser
Committer
Ledgers
Chaincode
Events
Org C
Endorser
Committer
Ledger
Chaincode
Events
Org D

Data Write Flow

Backend Invocation

backend application calls the appropriate chaincode function using Fabric SDK, creating transaction proposal

Execution (Peer)

Backend sends the proposal to the designated Endorsing Peers (peer1, peer2, etc.)

Read World State

Chaincode executes the logic (e.g. Debit = Credit rule). It first performs a read operation to get the current state and version of the affected wallets from its local WSDB (CouchDB/LevelDB)

Simulation & Signature

Endorsing Peer simulates the transaction and creates a Read/Write Set (RW-Set).The Endorsing Peer signs the RW-Set with its digital certificate.

Block Creation

The Ordering Service sequences the transaction and bundles it into an immutable Block.

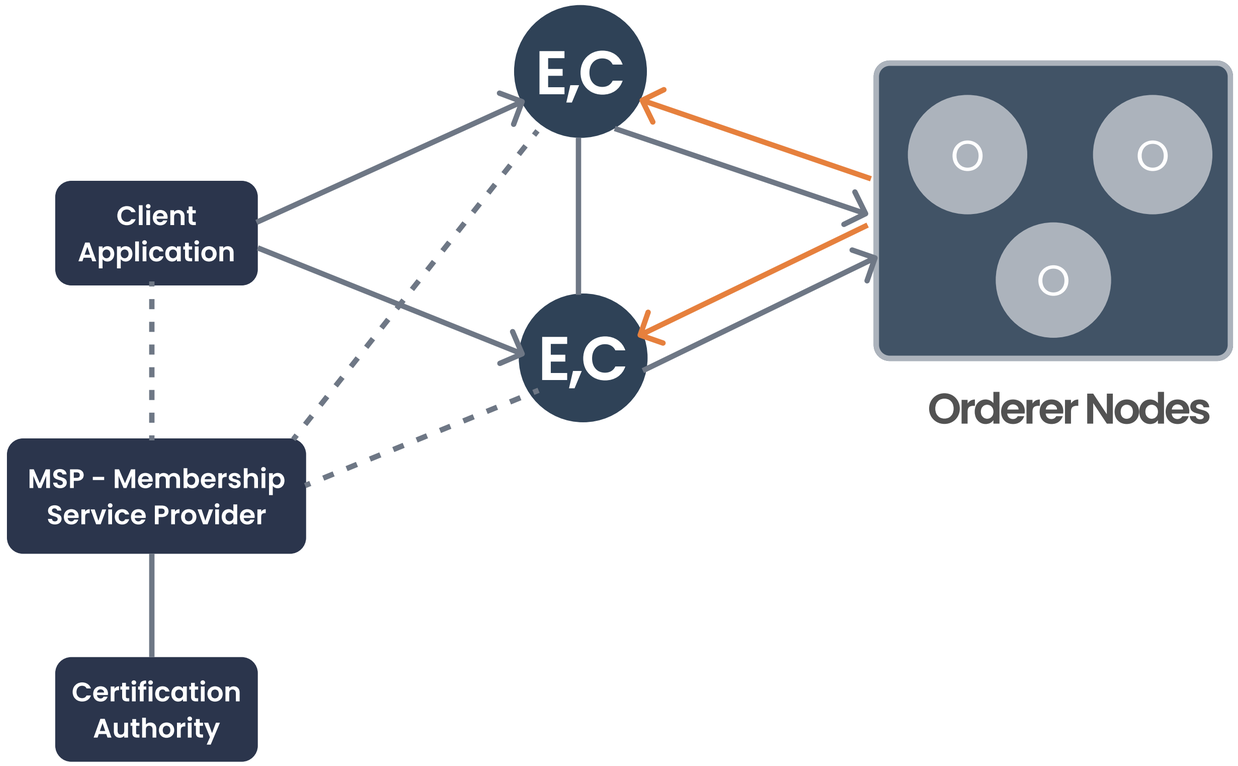
Commit (Peer)

The Peer Nodes receive the new Block and perform final validation (checking the endorsement policy and MVCC against their WSDB)

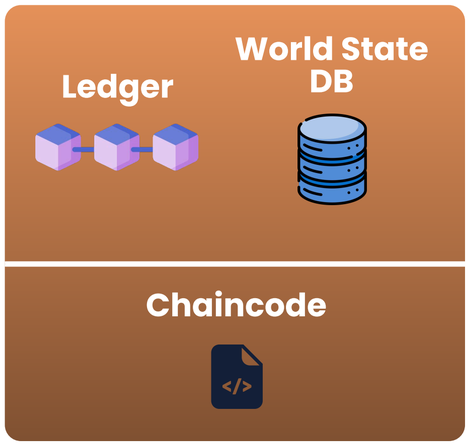
Data Storage

the Peer performs two atomic writes simultaneously. **Ledger**: ledger gets updated with the latest transaction and **WSDB**: updates the wallet balances to their latest values.

Minimum Requirements



2 Peer Nodes
3 Orderer Nodes



Chaincode

Business Logic

Enforce rules to read state

Transaction Instructions

Governed by system's endorsement policy

```
// ============================================================
// Invoke - Our entry point for Invocations
// ============================================================
nc (t *SimpleChaincode) Invoke(stub shim.ChaincodeStubInterface)
    function, args := stub.GetFunctionAndParameters()
    fmt.Println(" ")
    fmt.Println("starting invoke, for - " + function)

    // Handle different functions
    if function == "init" {                    //initialize the
        return t.Init(stub)
    } else if function == "read" {             //generic read l
        return read(stub, args)
    } else if function == "write" {            //generic writes
        return write(stub, args)
```

Word State Database

LevelDB

- Embedded Key-Value Store (NoSQL)
- Limited Query capabilities, Supports only key lookup, key range queries, and composite keys.
- Faster for simple operations

CouchDB

- External Document Store (NoSQL)
- JSON Documents with IDs. Enables flexible schema.
- Supports wide range of queries based on data content & indexing

Off-Chain Storage

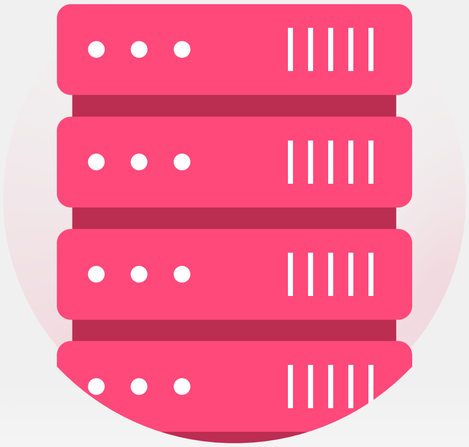
MongoDB

- Acts as a synchronized Cache.
- Operational Data: Customer Names, Addresses, Order Data, etc.
- Standard application database queries
- High-Speed Flexibility & Reporting

Network Topology

Cloud Hosted Network

Peer Nodes are hosted by any cloud providers



Self hosted network

Environment Network owned by us, connects with an HTTPs channel

API

01 REST & JSON RPC APIs

02 CLI Administer

03 SDK (GoLang, Java, JavaScript)

Performance

Speed

1,000 to over 5,000+ TPS (Transactions Per Second) in optimized setups.

Transaction Latency

Seconds, providing near-instant finality. Once committed it transaction cannot be reversed.

Read Throughput

World State Database are executed locally by the Peer and do not involve consensus, which makes it as fast as traditional database query.

Platform Licensing

The Fabric software itself is open-source. No recurring software licensing fees.

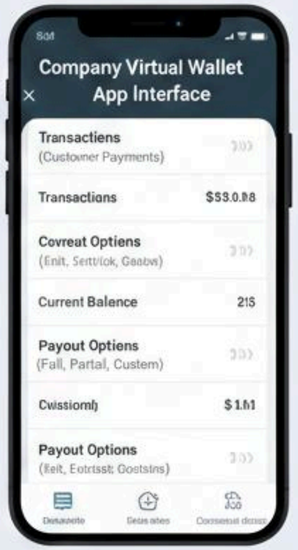
Thanks

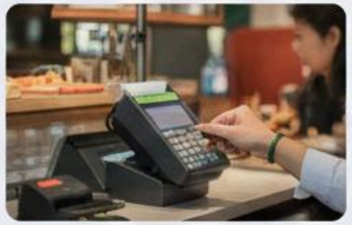
The future is decentralized. Let’s lead the way.

Mehadi Hasan

Junior Blockchain Developer

Seamless Enterprise End Payment Ecosystem



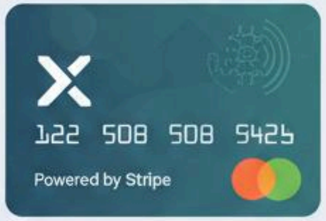


Customer Payment
Stripe Account (Company X)

Stripe Account
(Company X)

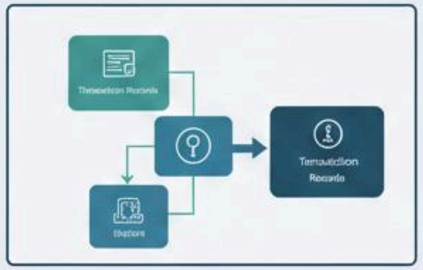
Company X Virtual Wallet

Payouts to X-Card



Restaurant X-Branced Card
(Issued by Stripe)

Company Payout System



Hyperlegrer Fabric Blockchain Ledger

Immubiility & Integrity

Benefits

Stripe → Company
Card Issuing Commission

Benefits

- Reduced 3rd party dependence
- Enhanced Security with Blockchain
- Client's Trust
- Operational Efficiency