# Last Year Engineering

# 18BTIT732-Professional Elective – II: Blockchain Technology

## Class - L.Y. (SEM-I)

## Unit - V
## Blockchain Application Development

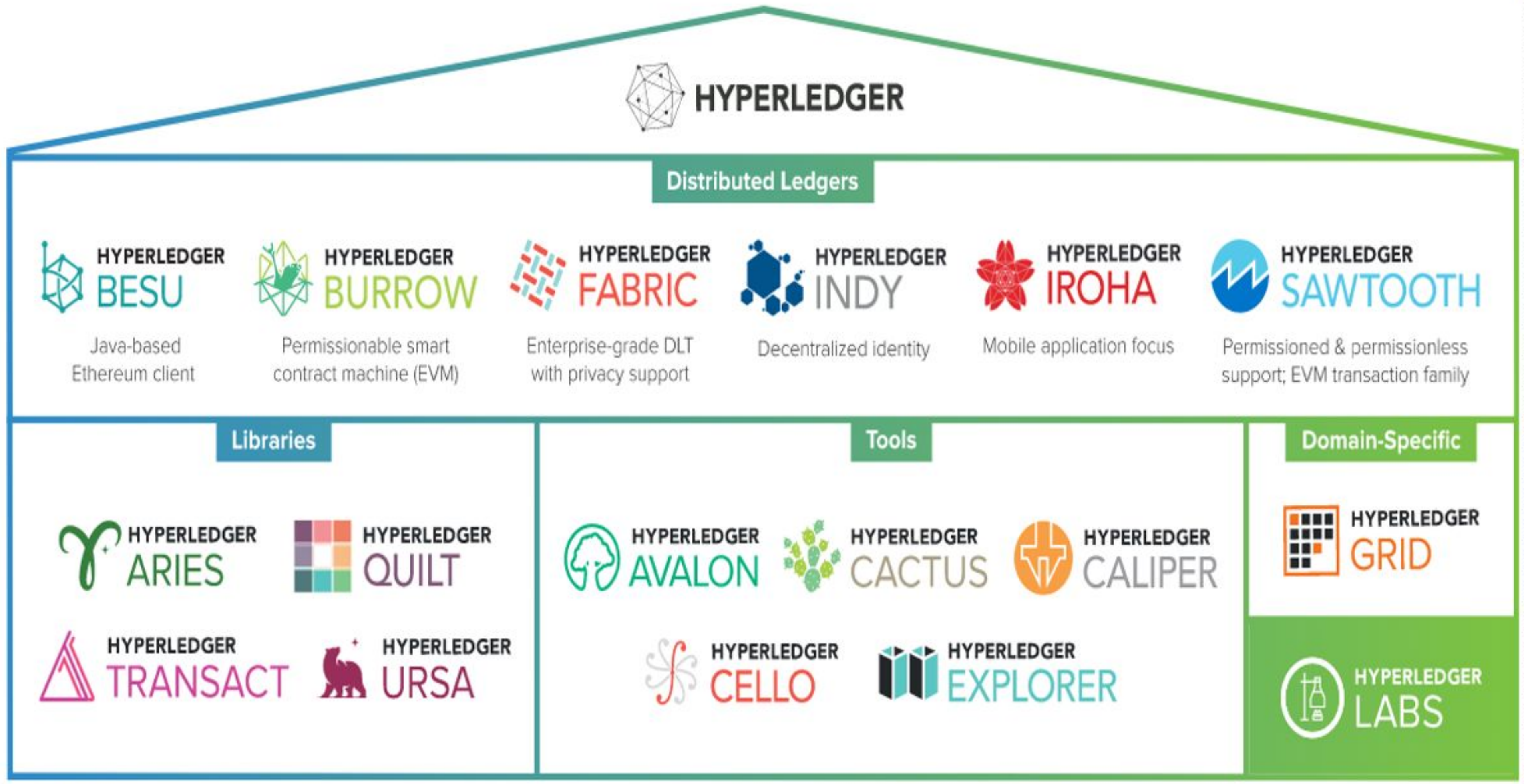
AY 2023-2024 **SEM-I**

# Unit-V Syllabus

- Hyperledger Fabric- Architecture, Identities and Policies, Membership and Access Control, Channels, Transaction Validation,
- Writing smart contract using Hyperledger Fabric, Writing smart contracts using Ethereum,
- Overview of Ripple and Corda

# Hyperledger

- Open source enterprise-grade permissioned distributed ledger technology (DLT) platform
- Collaborative blockchain effort hosted by Linux Foundation
- Mission
  - Create enterprise grade, open source distributed ledger
  - Frameworks



HYPERLEDGER

Distributed Ledgers

- HYPERLEDGER BESU: Java-based Ethereum client
- HYPERLEDGER BURROW: Permissionable smart contract machine (EVM)
- HYPERLEDGER FABRIC: Enterprise-grade DLT with privacy support
- HYPERLEDGER INDY: Decentralized identity
- HYPERLEDGER IROHA: Mobile application focus
- HYPERLEDGER SAWTOOTH: Permissioned & permissionless support; EVM transaction family

Libraries

- HYPERLEDGER ARIES
- HYPERLEDGER QUILT
- HYPERLEDGER TRANSACT
- HYPERLEDGER URSA

Tools

- HYPERLEDGER AVALON
- HYPERLEDGER CACTUS
- HYPERLEDGER CALIPER
- HYPERLEDGER CELLO
- HYPERLEDGER EXPLORER

Domain-Specific

- HYPERLEDGER GRID

HYPERLEDGER LABS

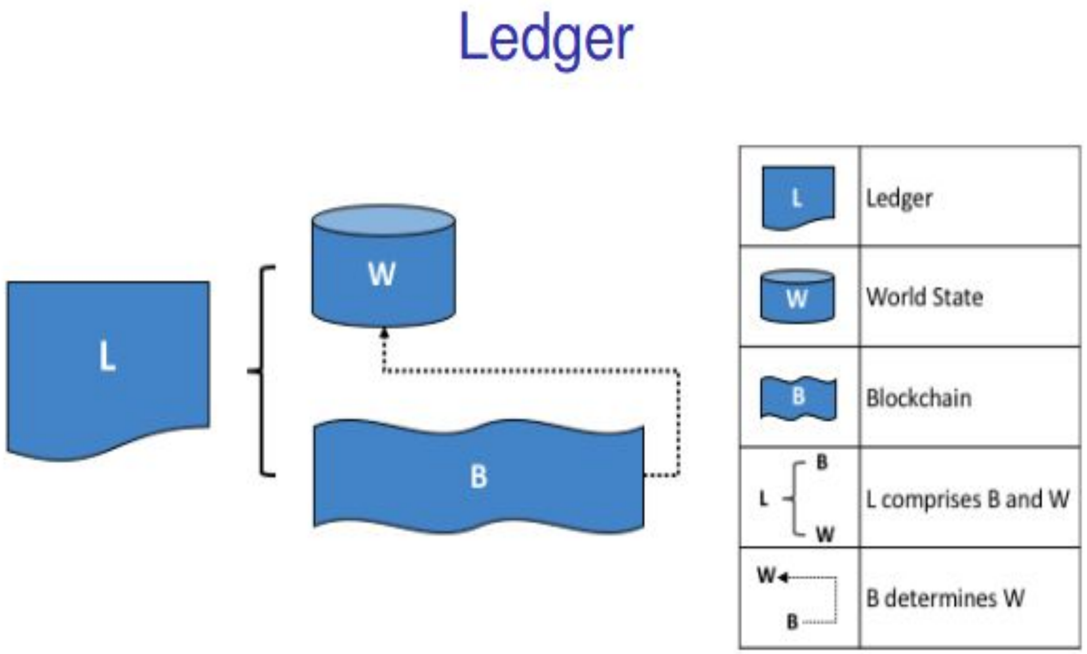
# Hyperledger Fabric

- Permissioned distributed ledger framework with smart contracts
- Members of a Fabric network enroll through a Membership Service Provider
- A group of participants can create a channel (shared ledger)
- Copies of the channel ledger present only with channel participants
- Each ledger contains world state and transaction log
- Transactions are used to update state
- Smart contracts (called chaincode) are written in Go
- Pluggable consensus mechanism
- Client SDKs available in Node.js and Java
  - Querying ledger for transactions or blocks
  - Installing chaincode in peer nodes
  - Creating transactions calling chaincode functions

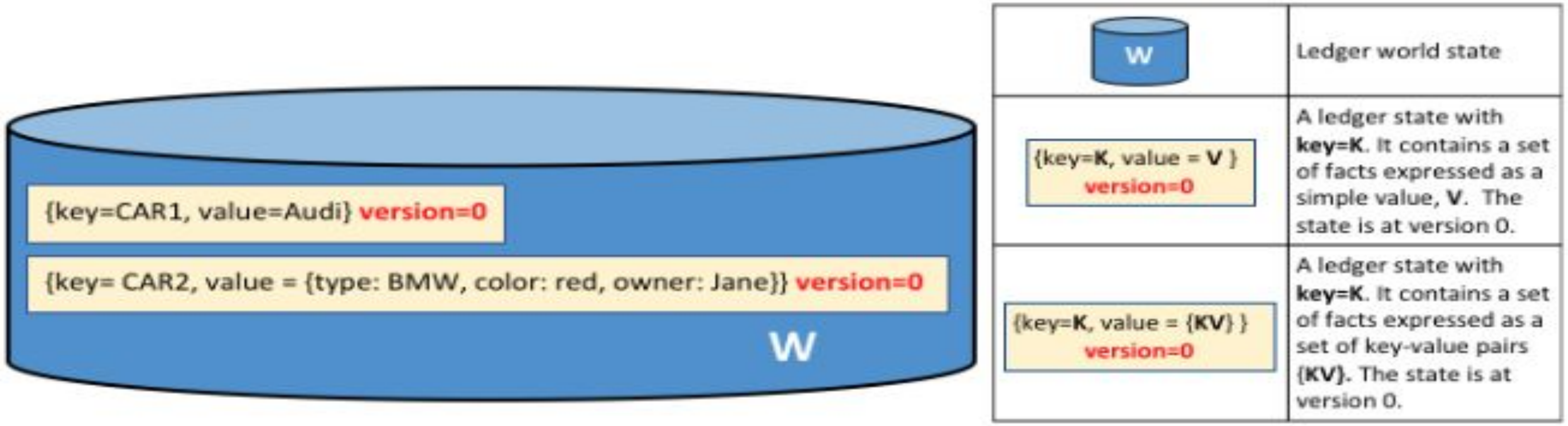
# Ledger

Ledger has two types

- World state – which holds current transactions
- Transaction state – which holds history of transactions



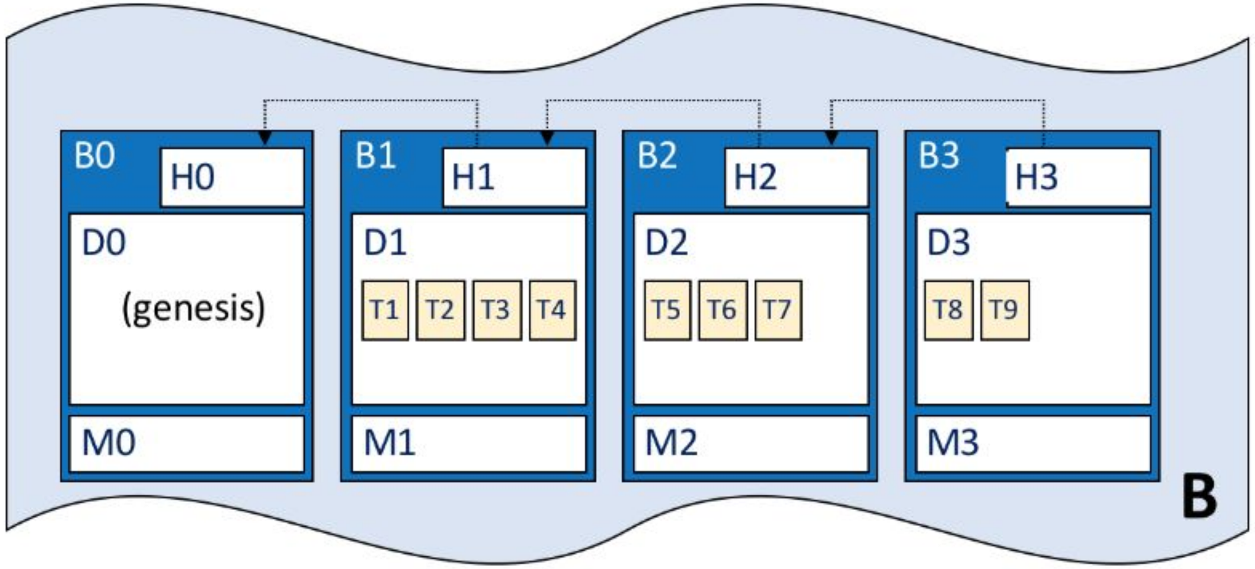
# World State


{key=CAR1, value=Audi} version=0
{key= CAR2, value = {type: BMW, color: red, owner: Jane}} version=0
W

| | |
|---|---|
| W | Ledger world state |
| {key=**K**, value = **V** }<br>version=0 | A ledger state with **key=K**. It contains a set of facts expressed as a simple value, **V**. The state is at version 0. |
| {key=**K**, value = {**KV**} }<br>version=0 | A ledger state with **key=K**. It contains a set of facts expressed as a set of key-value pairs {**KV**}. The state is at version 0. |

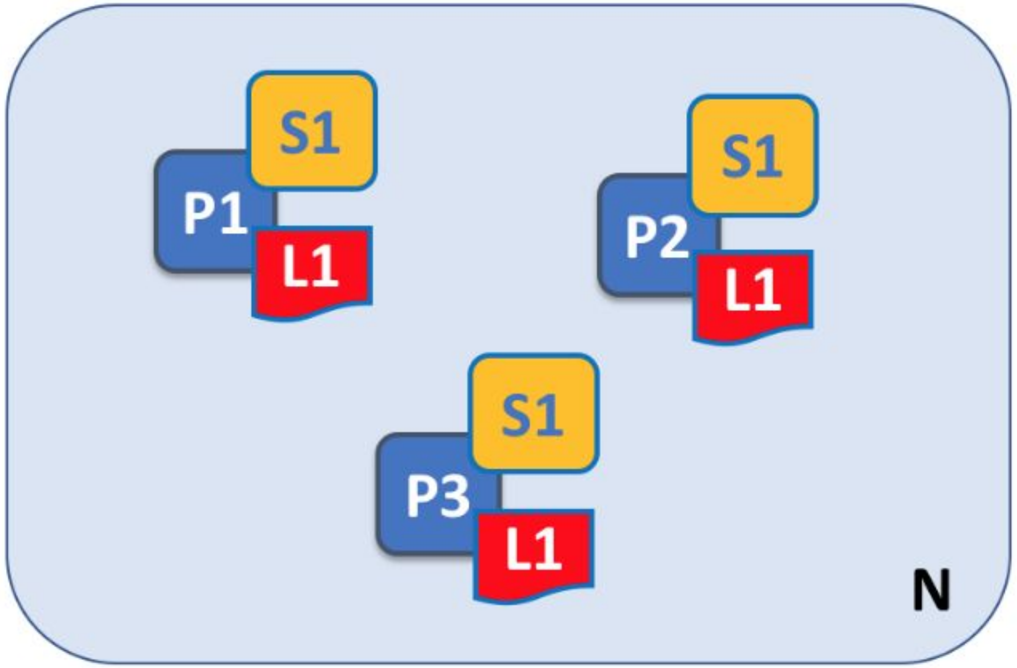
Image credit: https://hyperledger-fabric.readthedocs.io/en/release-1.3/ledger/ledger.html

# Blockchain



| | |
|---|---|
| B | Blockchain |
| B1 | Block |
| H3 | Block header |
| D1 | Block data |
| T5 | Transaction |
| M3 | Block metadata |
| H1 H2 | H2 is chained to H1 |

# Peers



| | |
|---|---|
| N | Blockchain network |
| P | Peer node |
| S | Smart contract (aka chaincode) |
| L | Ledger |

# Hyperledger Fabric components

- Fabric CA
- Membership Service provider
- Client
- Peer
- Orderer
- Channel
- Chaincode

# Fabric CA

- Registration of identity or connects to LDAP as the user registry
- Issuance of enrollments certificate
- Certificate renewal and revocation
- Certificates are x509 standards
- Consist of both server and client components

# Peer

- Peer is place where the ledger and the blockchain data is stored
- One peer may be part of multiple channels
- Every single channel inside the peer
- It endorse and update the ledger
- You can create backup of the ledger from the peer

# Types of Peer or Node

## Endorser Peer

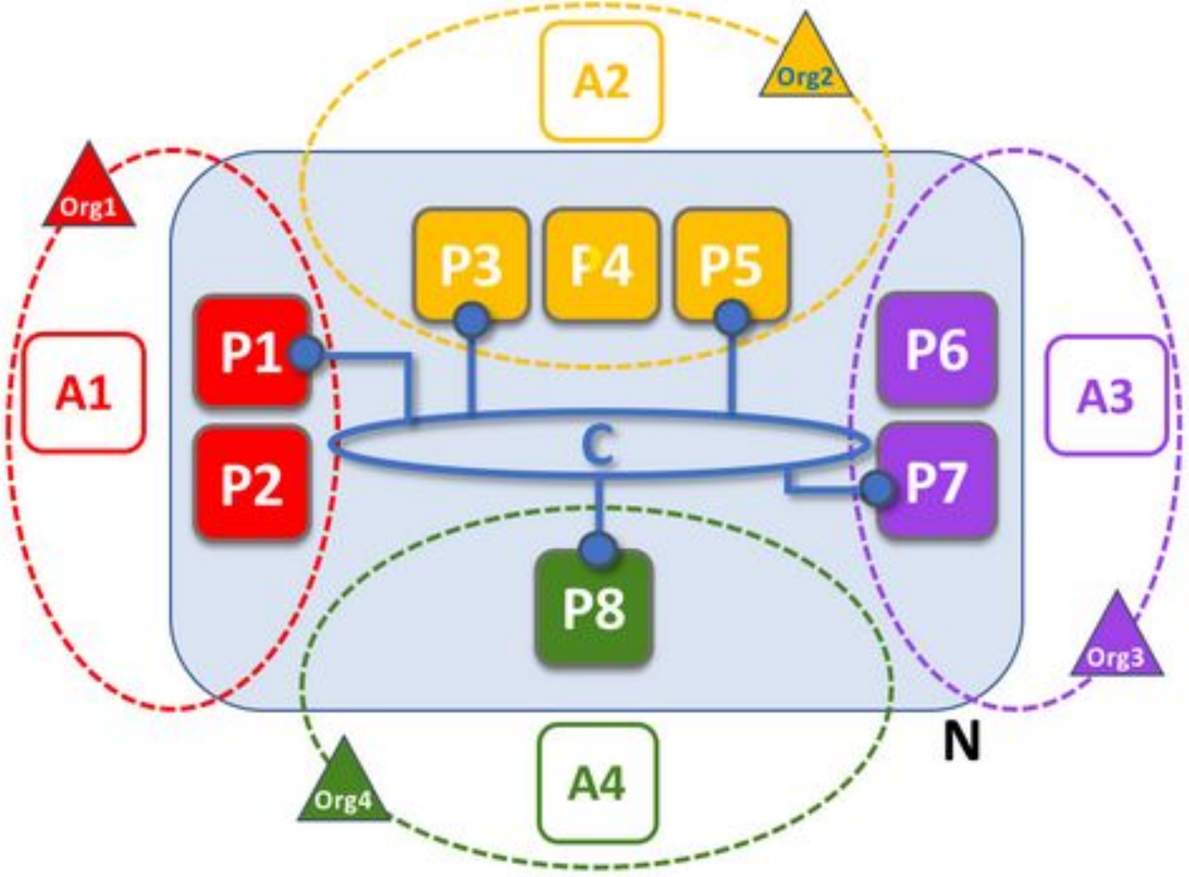
- Validation the transaction
- Execute the chaincode but does not update the ledger
- Approve and Disapproved transactions

## Anchor Peer

- When channel is created anchor peer will also create
- Broadcast the blocks to other peers in the organization

## Orderer Peer

- Central communication channel
- Include transaction into blocks



| | | | |
|---|---|---|---|
| N | Blockchain Network | L | Ledger |
| C | Channel | A | Application |
| P | Peer | PA<br>C | Principal PA (e.g. A1, P5) communicates via channel C. |
| | | Org | Organization |
| A1 R<br>P1 P2 | | Organization R owns application A1 and peers P1, P2. | |

# Orderer service

- Ordering service is actually the heart of the consensus algorithm and the heart of Hyperledger fabric
- Provide the order of operations
- Before committing anything to ledger it must pass through the ordering service
- It is responsible for verification, security, policy, verification etc.
- Include the transaction into blocks
- Create the block and deliver to other peers
- Ordering service uses following algorithm
  - Solo
  - Kafka
  - RAFT

# Channels

- Channel is private “subnet” of communication between two or more specific networks members
- A channel is defined by members (org), anchor peer per member, the shared ledger, chaincode application and ordering service
- Each peer that joins a channel has its own identity given by MSP(membership service provider)
- Channels are completed isolated
- They have different ledger, different height of blocks, policies and stories rules
- Never exchange data
- Outside of channel organization or peer not able to access
- You can make access policies
- Every single peer or party inside a channel must agree another other parties or peer

# Chaincode

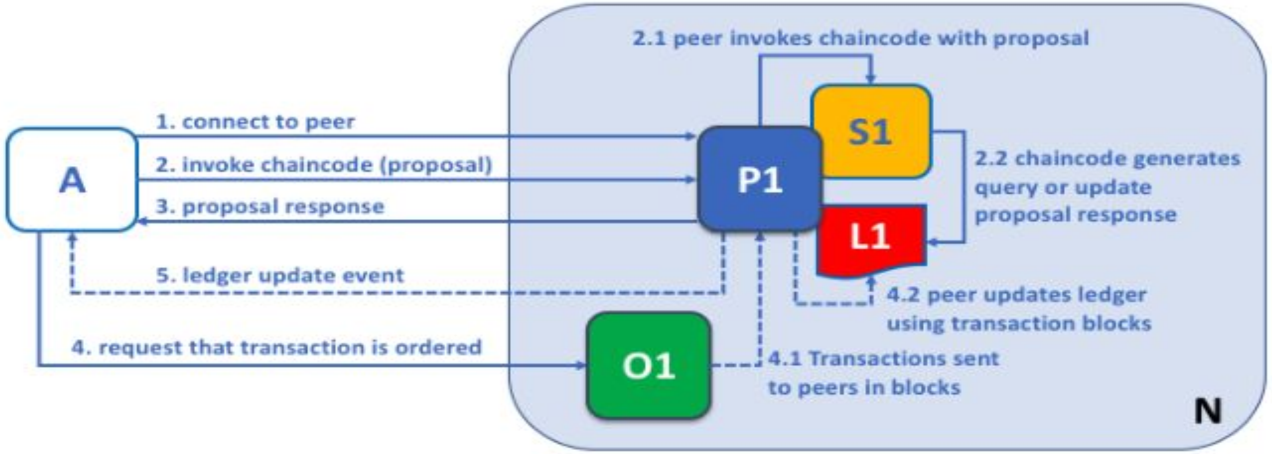
- A chaincode typically handles business logic agreed to by members of the network, So, it similar to smart contract
- All your business logic is inside chaincode
- It’s written in Go, NodeJS or JAVA
- Chaincode may installed on every channel and peer
- Policy must be provided

# MSP (Membership Service Provider)

- Abstract away all cryptographic mechanisms and protocols behind issuing certificates, Validating certificates, and user authentication

- Local MSP
  - Defined on file system of the node or user to which they reply
  - Only one msp per node
- Channel MSP
  - This will be available for all the nodes in channel

# Application-Peer Interaction

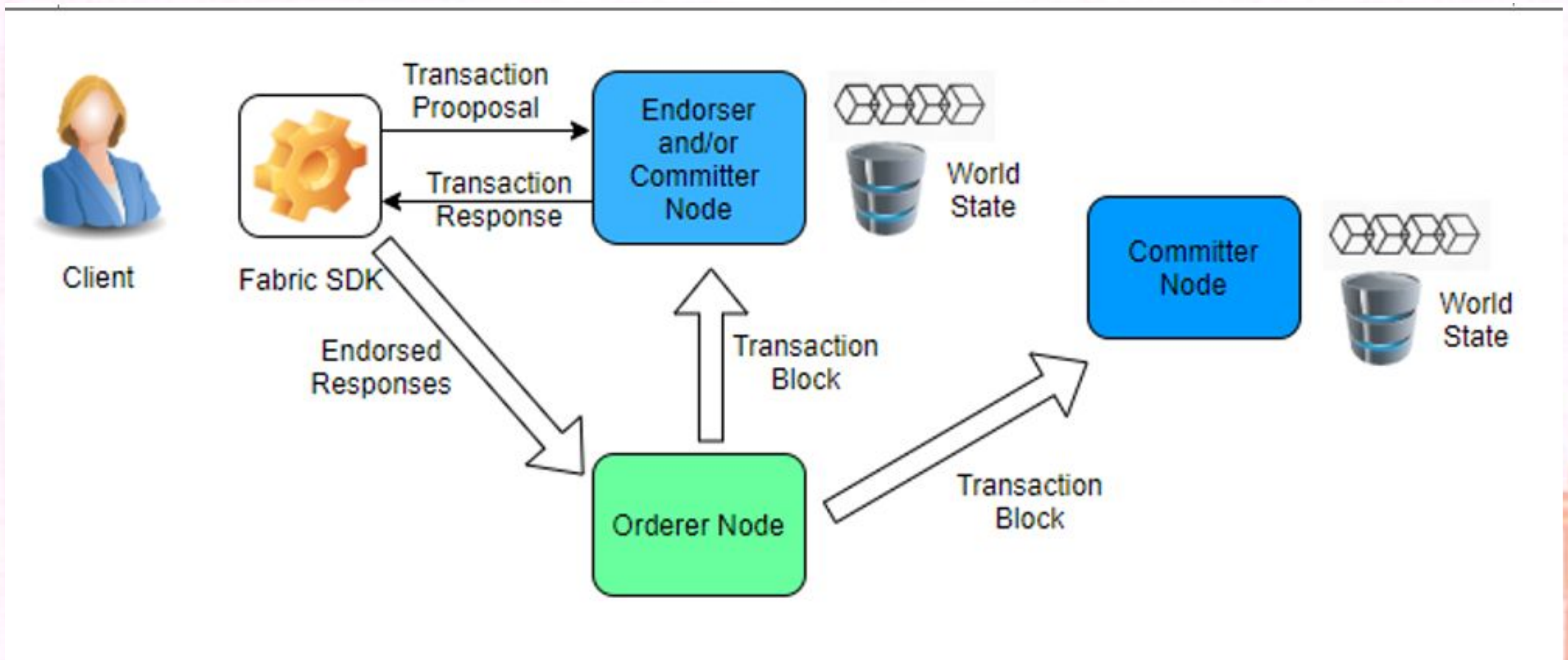

2.1 peer invokes chaincode with proposal
1. connect to peer
2. invoke chaincode (proposal)
3. proposal response
5. ledger update event
4. request that transaction is ordered
A
P1
S1
L1
O1
2.2 chaincode generates query or update proposal response
4.2 peer updates ledger using transaction blocks
4.1 Transactions sent to peers in blocks
N

| | |
|---|---|
| N | Blockchain Network |
| A | Application |
| P | Peer |
| S | Chaincode |
| L | Ledger |
| O | Orderer |

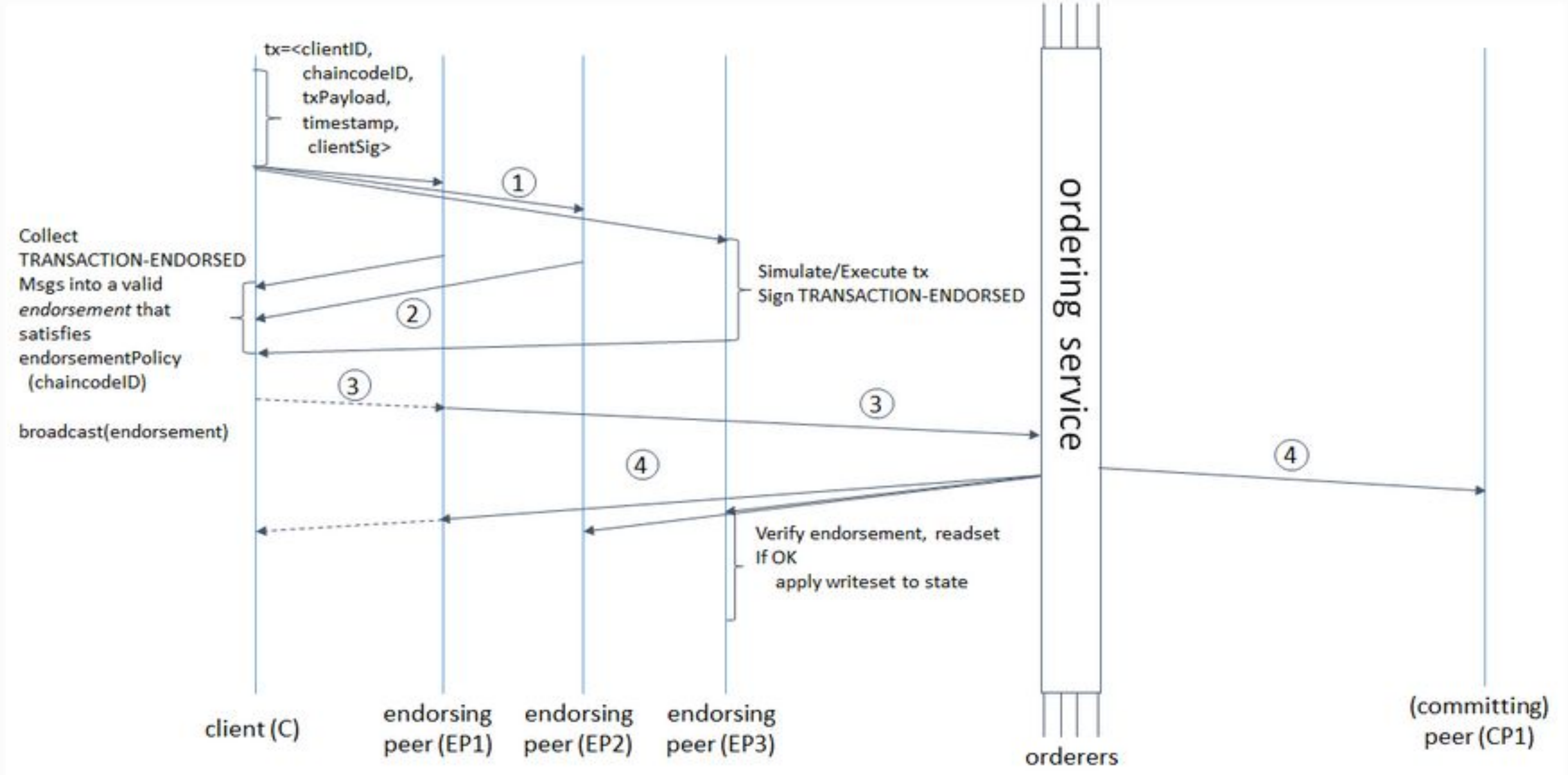
Image credit: https://hyperledger-fabric.readthedocs.io/en/release-1.3/peers/peers.html

- Ledger queries involve only first three steps
- Ledger updates involve all five steps
- Application needs to send proposed updates to several peers

# Transaction Flow of Hyperledger fabric



# Transaction Flow of Hyperledger fabric



# Chaincode Lifecycle

- Package
- Install
- QueryInstalled
- Approve
- Check Readineess
- Commit
- QueryCommit