Permission Blockchains (Part 1)

Permissioned blockchain

- Each node has an identity provided by the Membership Service Provider(MSP)
- Various level of identities: Trusted to non-trusted
- Certain tasks are assigned to certain identities
- Many voting based PoS blockchains can be converted to permissioned blockchains by allowing voting/mining rights to only certain identities
- Allow only certain identities to mine

Food Trust

What?

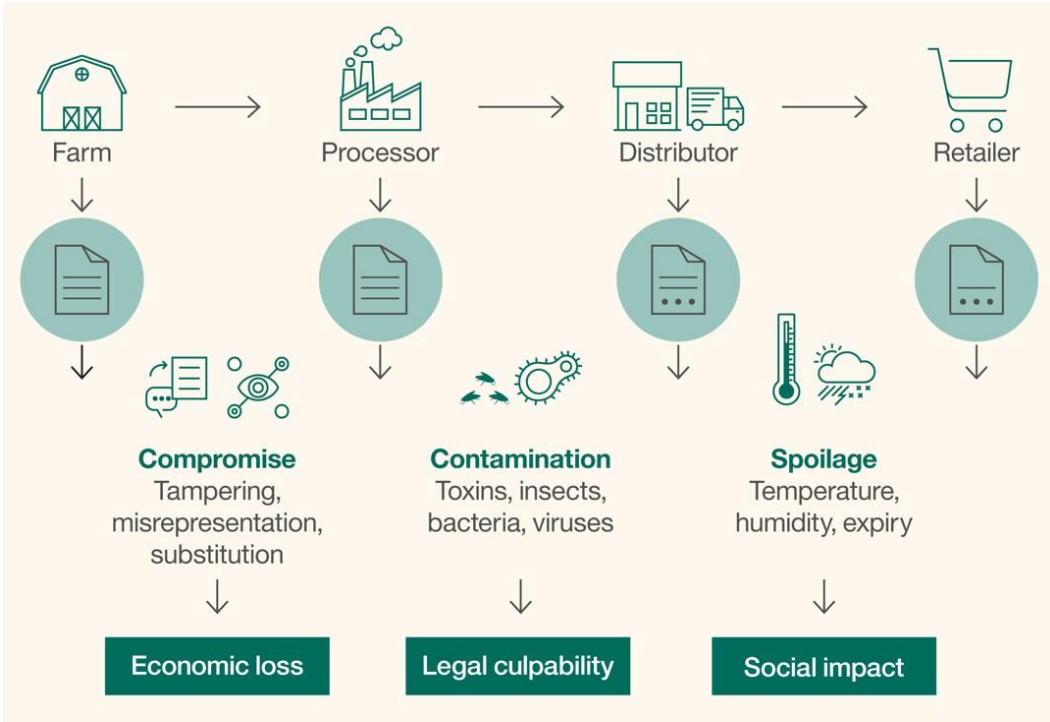
- Provide a trusted source of information and traceability to improve transparency and efficiency across the food network.

How?

- Shared ledger for storing digital compliance documentation, test results and audit certificates network.

Benefits

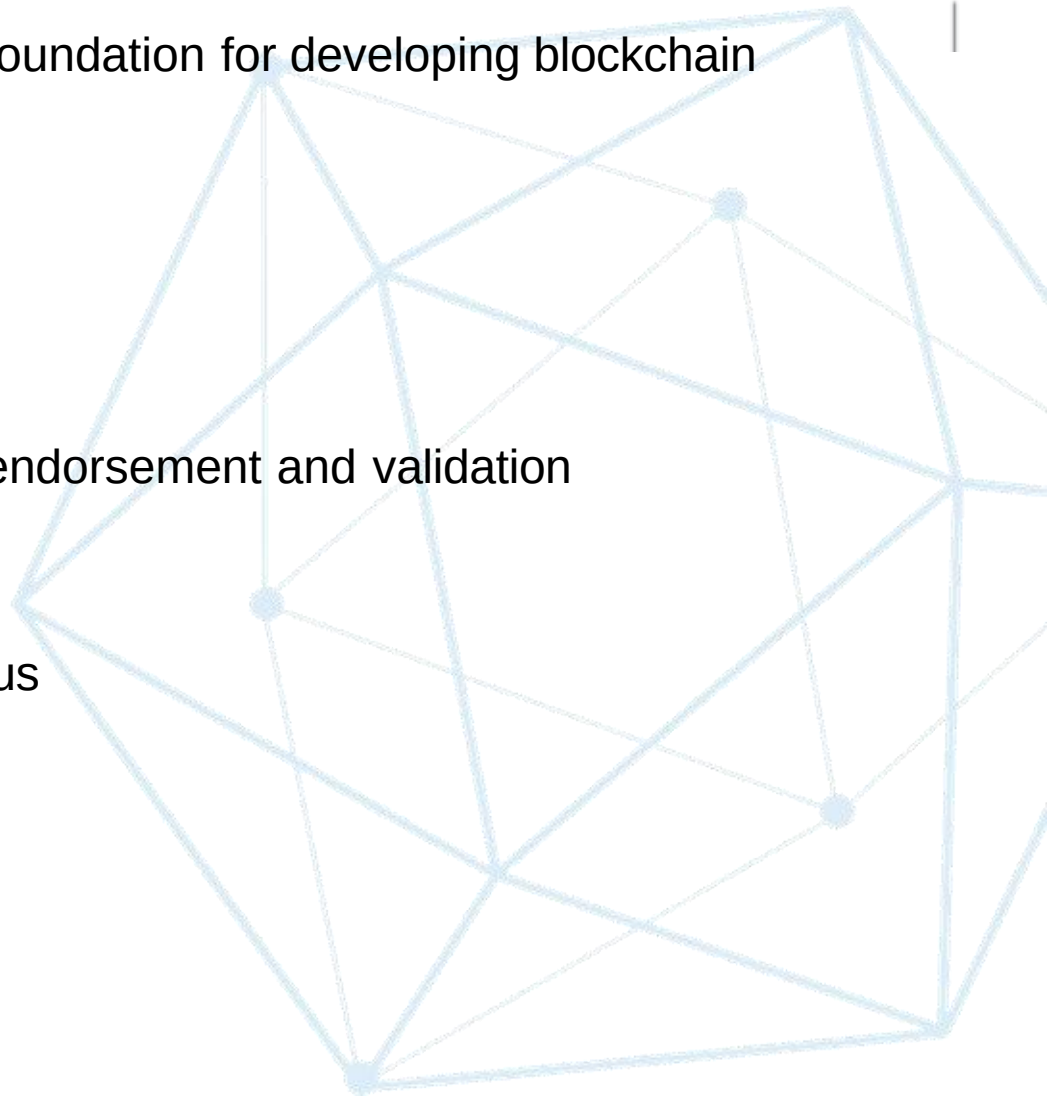
- Reduce impact of food recalls through instant access to end-to-end traceability data to verify history in the food network and supply chain.
- Help to address the 1 in 10 people sickened and 400,000 fatalities which occur every year from food-born illnesses.



What is Hyperledger Fabric

- An implementation of blockchain technology that is intended as a foundation for developing blockchain applications for the enterprise

- Key characteristics:
  - Permissioned
  - Highly modular
    - Pluggable consensus, ledger, membership services, endorsement and validation
  - Smart contracts in general purpose languages
  - Privacy
  - No “mining” or native crypto-currency required for consensus
  - Execute-order-validate vs order-execute



Why need a permissioned blockchain

- Identities on blockchain can be linked to identities in real world
- Can track misbehavior (like certain DoS attacks) to identities
  - Weaker adversary assumption justified

- Implications
  - No need for sybil/spam resistance
  - No need for on-chain incentives
  - No need for currency implementation
  - Potentially very efficient and scalable

Hyperledger Fabric

- Modular permissioned blockchain
- Privacy preserving
- Horizontal compute scaling: not requiring all nodes to execute State Transition function.
- Main contribution: Order-Execute to Execute-Order-Validate structure

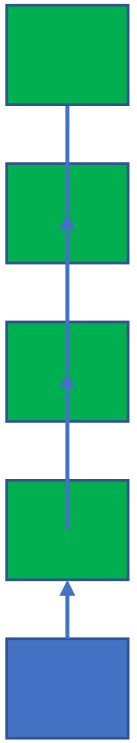
Order-Execute

- Consensus or atomic broadcast
- Deterministic (!) execution
- Persist state on all peers

Source:https://arxiv.org/pdf/1801.10228.pdf

Order-execute: Bitcoin

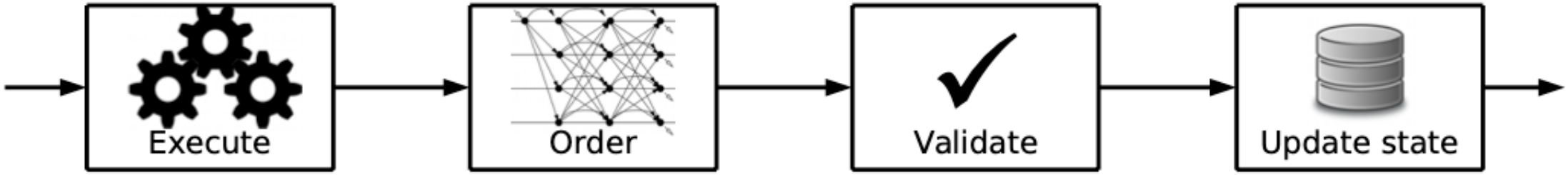
State transition function is executed
After ordering the block



Limitations of Order-execute

- Sequential execution can be throttled by a compute heavy state transition
  - Bypassed by transactions paying for execution cost
  - Cannot be done without any native cryptocurrency
- Non-deterministic code
  - Eg. Map iterator in go
  - Can lead to forks after ordering
  - Need to use limited languages like Solidity
- Confidentiality in execution

Hyperledger: Execute-Order-validate



- Simulate trans. and endorse
- Create rw-set
- Collect endorse-ments

- Order rw-sets
- Atomic broadcast (consensus)
- Stateless ordering service

- Validate endorse-ments & rw-sets
- Eliminate invalid and conflicting trans.

- Persist state on all peers

Source:https://arxiv.org/pdf/1801.10228.pdf

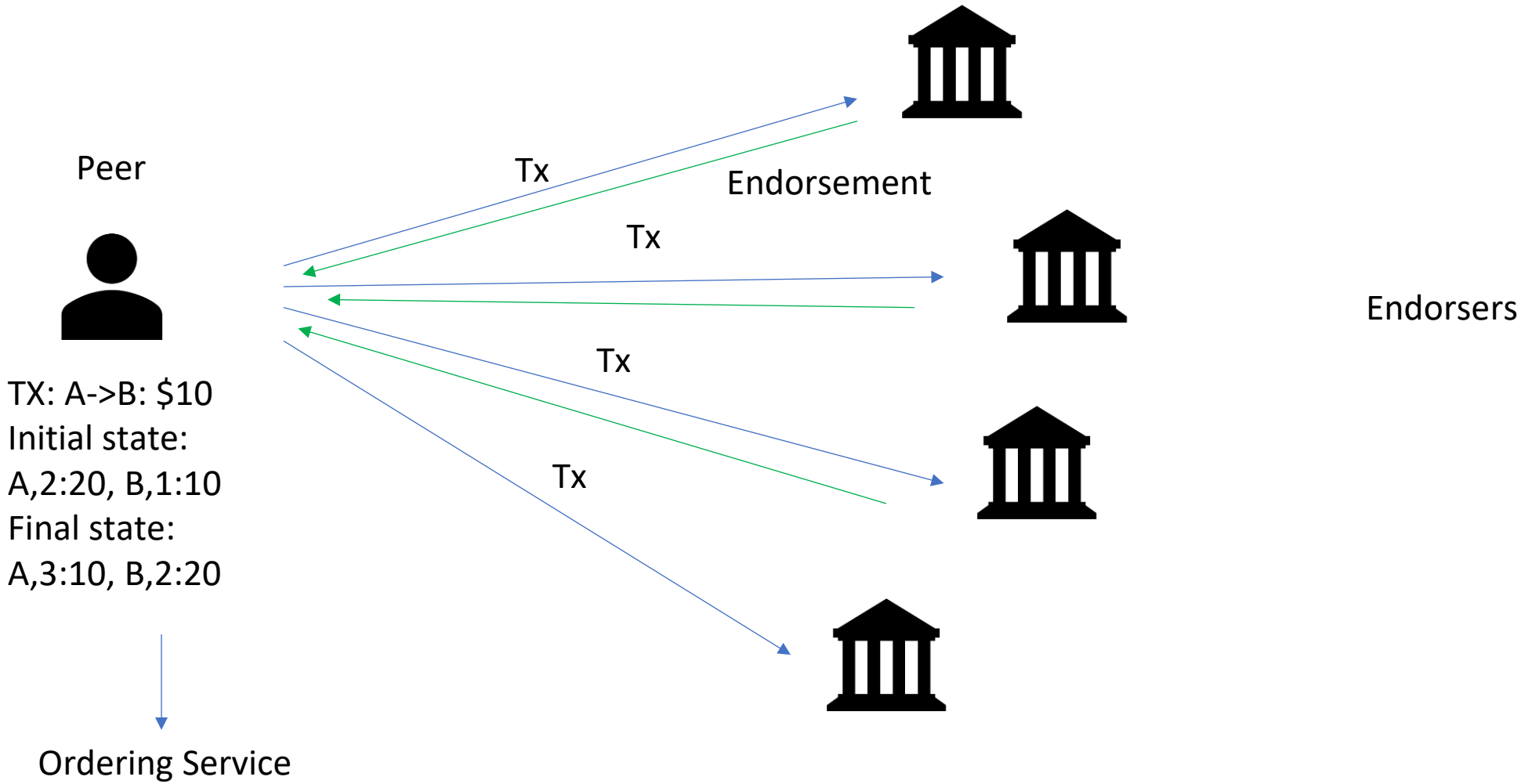
Chaincode and Endorsements

- Chaincode: smart contract + endorsement policy
- Smart contract implements the state transition function for an application
- State created by one Chaincode cannot be accessed by another Chaincode
- Chaincode can invoke another chaincode
- Each Chaincode specifies an endorsement policy
  - Eg. Need to collect 4 out of 10 signatures from a list of endorsers
  - Endorsers sign transactions that performs correct execution

State management

- State is maintained in a versioned key-value store
- (key, value, version)
- Each transaction has
  - Readset: (key, value, version) accessed by the transaction
  - Writeset: (key, value, version) changed by the transaction
- Endorsers sign a transaction along with the readset and writeset

Execution-Phase



Ordering Service

- A set of nodes responsible for consensus
- Receive transactions with endorsements from peers
- Orders them through some consensus mechanism
  - Can be Total-order broadcast of transactions
  - Can be any BFT based consensus algorithm
  - Can be Bitcoin like algorithm

Validation

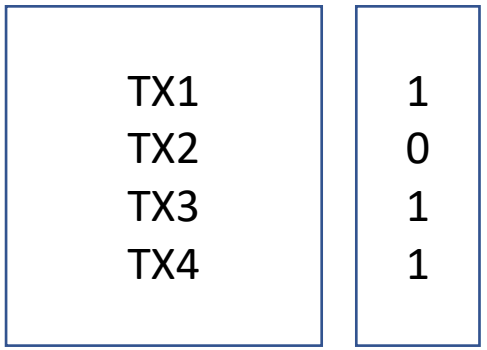
- Performed by all peers
- Endorsement policy evaluation: Check if the transaction has sufficient endorsements as specified by the Chaincode policy
  - can be done in parallel
- State update check: Check if the readset uses the correct version of keys and the writeset updates the key version (Sequential, however do not need to perform execution)
- Local state update: Update the local state (sequential)

Endorsement policy evaluation

- Example 1:
  - Chaincode A endorsement policy: need 2 of 3 endorsement from endorsers X,Y,Z
  - Chaincode A transaction: Contains endorsement from X,Z: Pass
  - Chaincode A transaction: Contains endorsement from W,X: Fail
  - Chaincode A transaction: Contains endorsement from Y: Fail
- Example 2:
  - Chaincode B endorsement policy: need endorsement from central bank

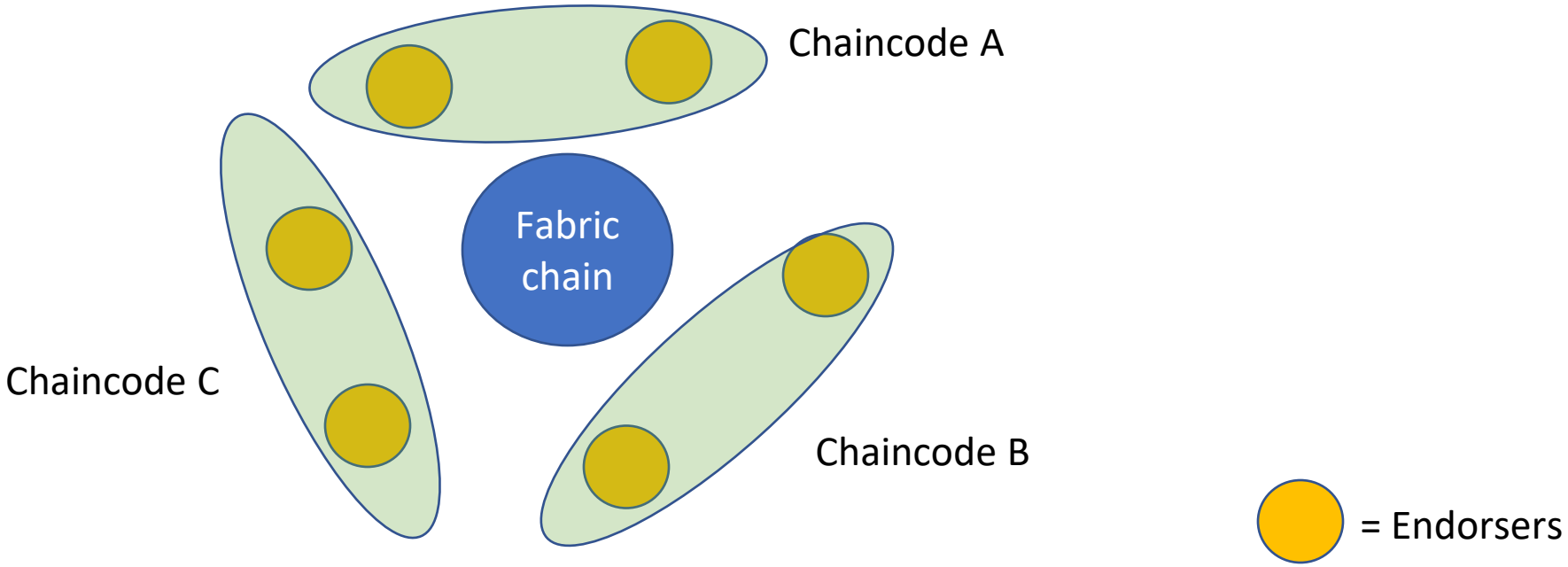
State update

- State update check example:
  - Tx: Readset: (Alice:20,1),(Bob:10,2) , Writeset: (Alice:10,2),(Bob:20,3)
  - Local state: (Alice:20,3), (Bob:10,2)
  - Tx readset does not match with local state: Fail
- State update:
  - Locally store a validity bitmask



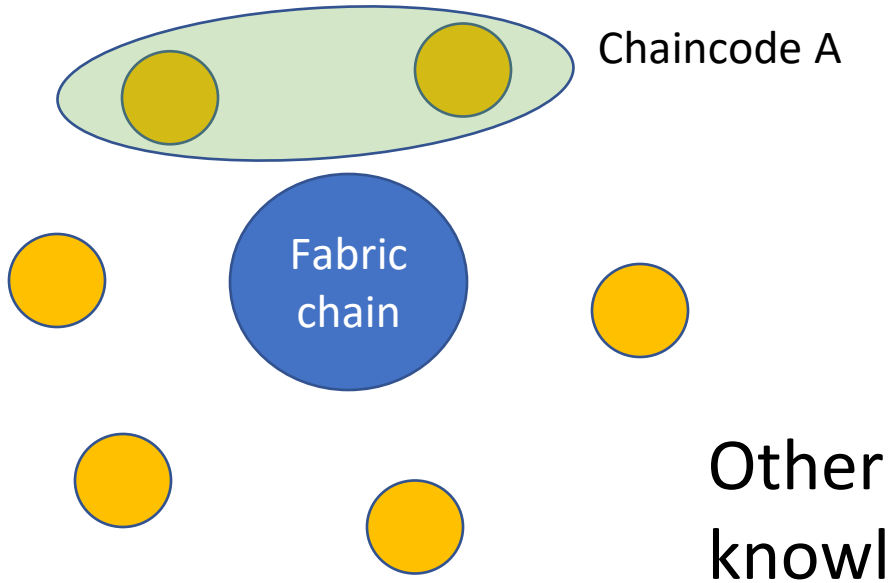
Concurrent execution

- Each Chaincode can use different set of endorsers



Confidentiality

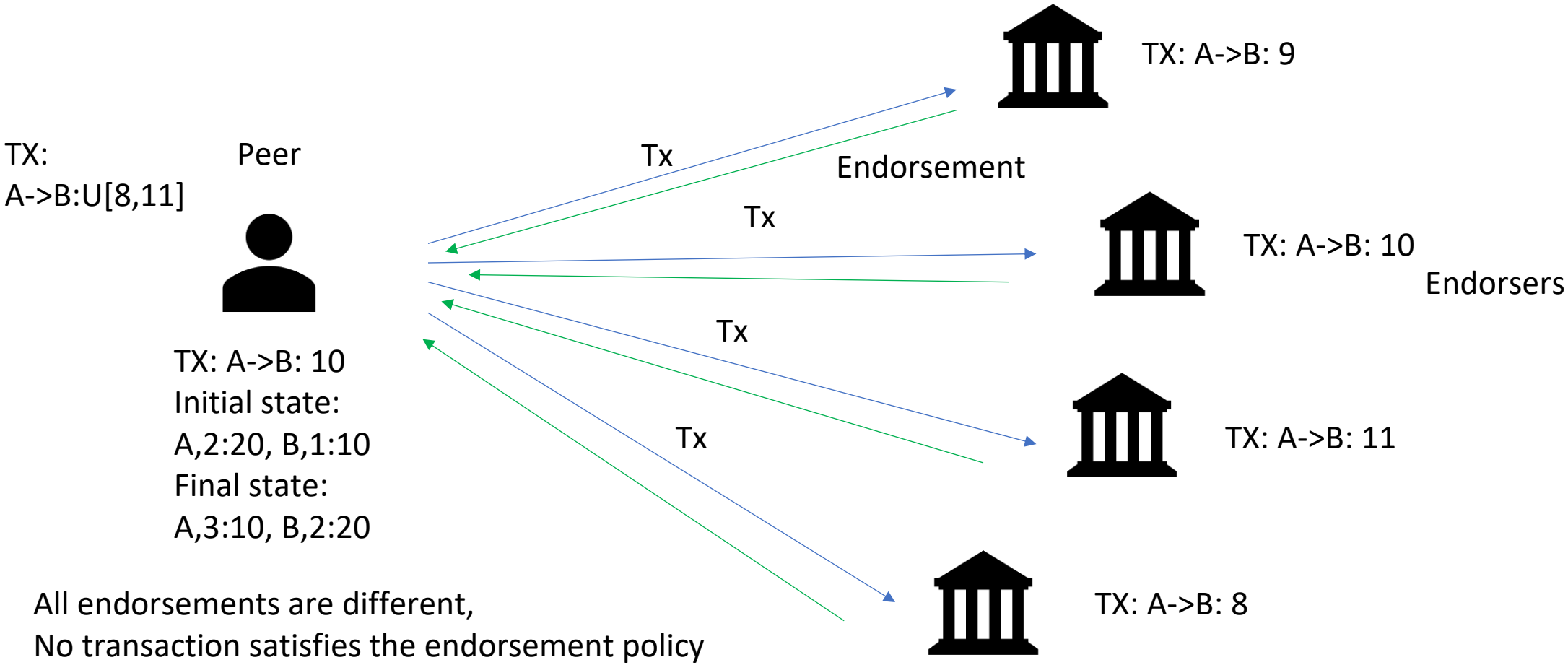
- The Chaincode execution code can be made private, only make the chaincode endorsement policy and other metadata public



Other endorsers have limited knowledge of Chaincode A's execution

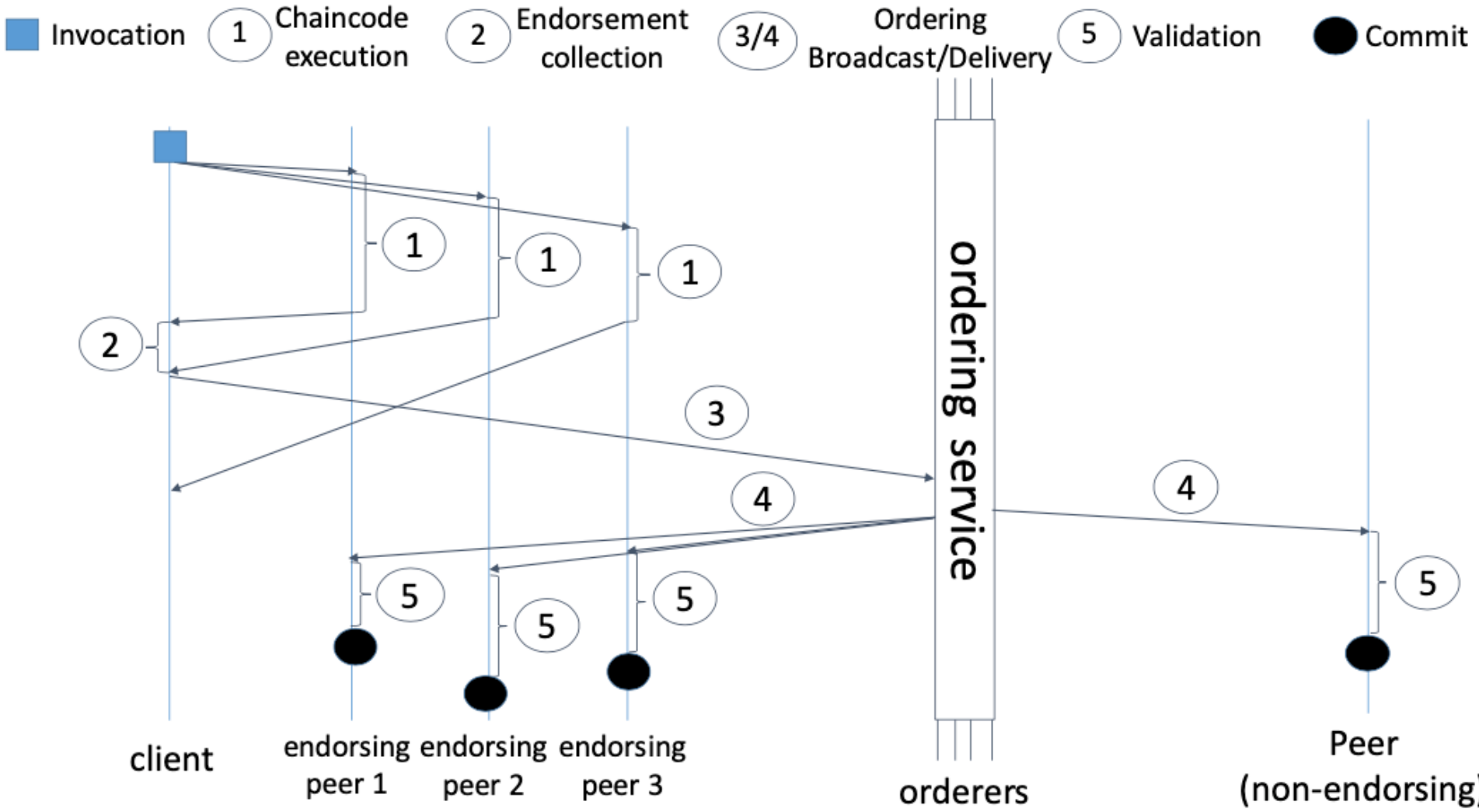
Deterministic execution

- Random execution won't collect enough endorsements



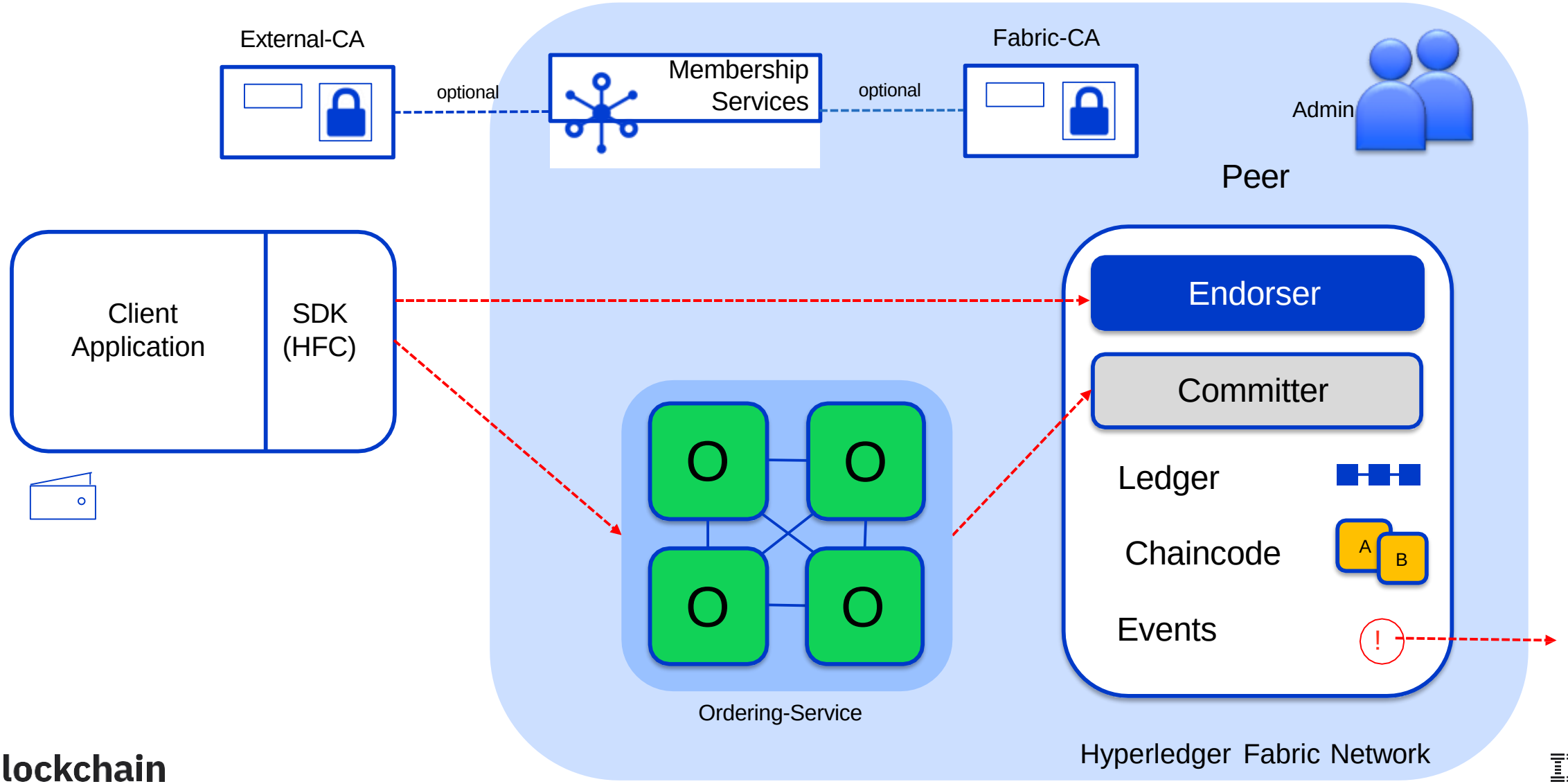
All endorsements are different,
No transaction satisfies the endorsement policy

Overview

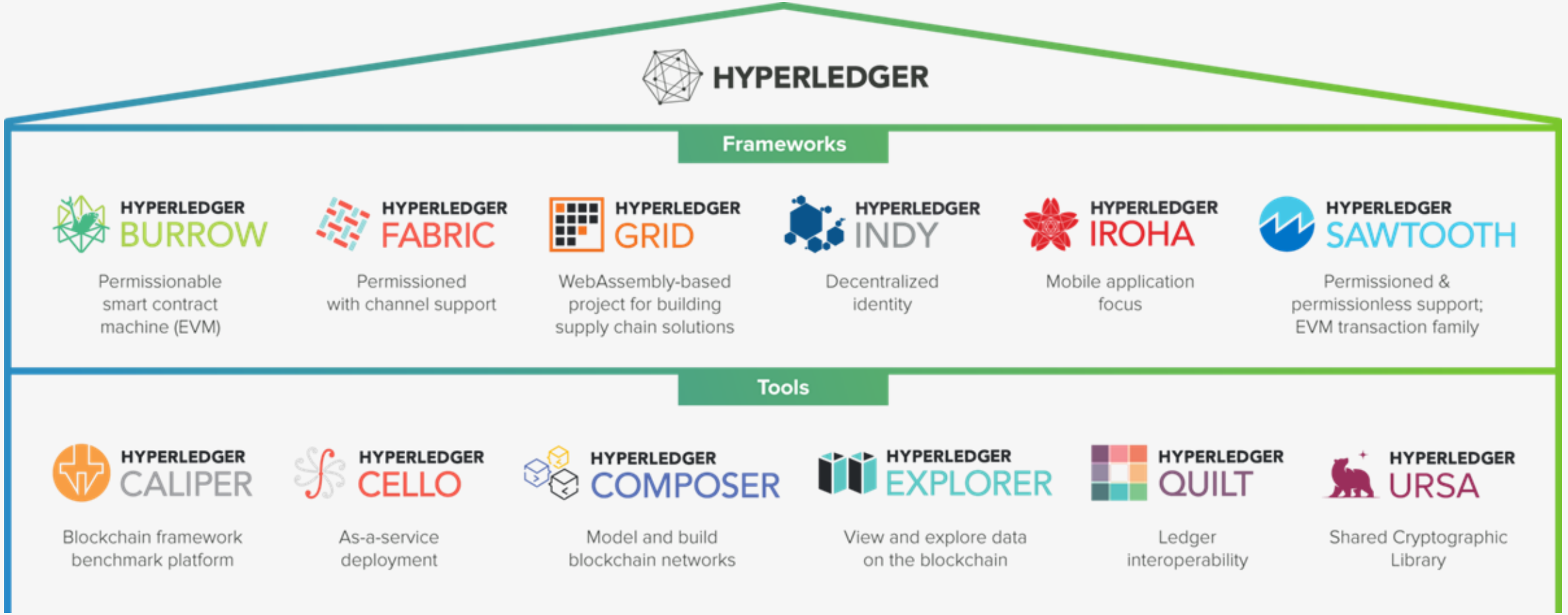


Source:https://arxiv.org/pdf/1801.10228.pdf

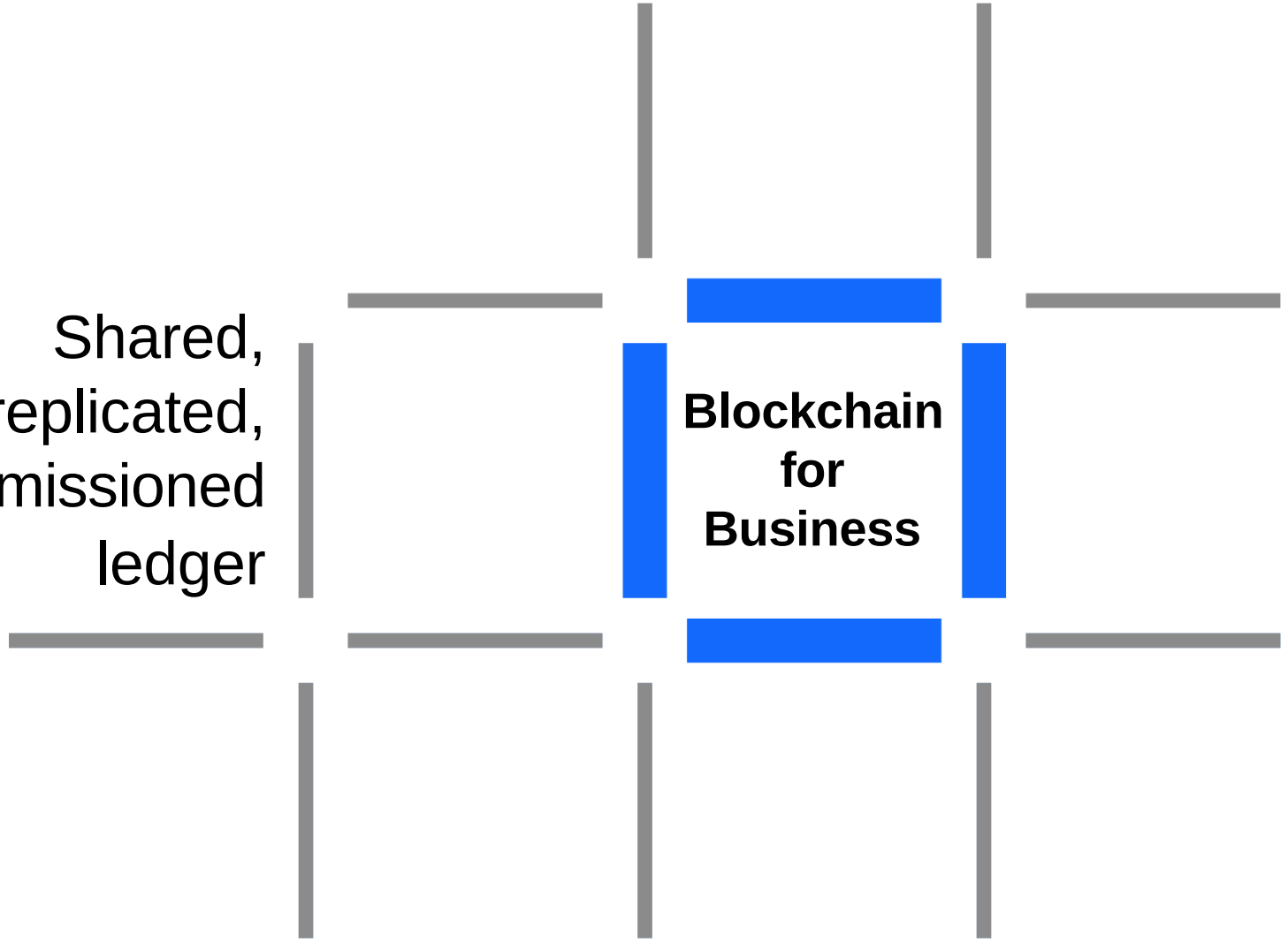
Hyperledger Fabric Architecture



Hyperledger Modular Approach



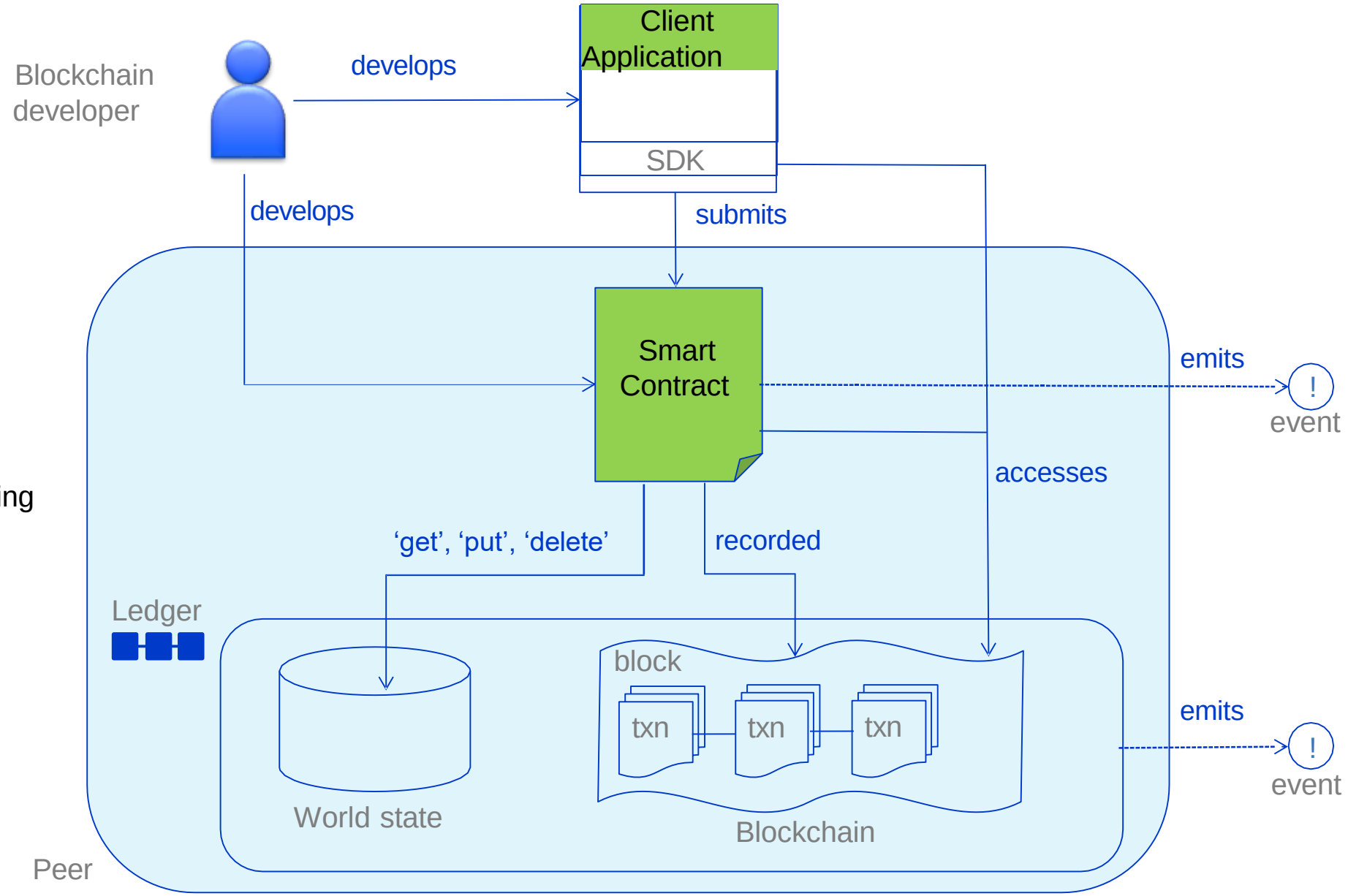
Blockchain for Business…



How applications interact with the ledger

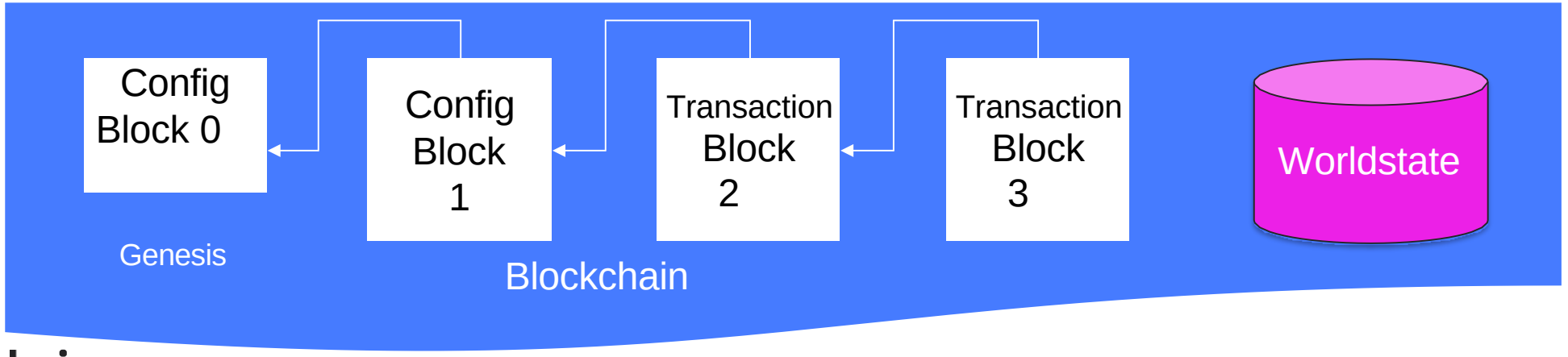
1. Client Application in using Hyperledger Fabric Client (HFC) SDK

2. Smart Contract implemented using chaincode – managing the World state

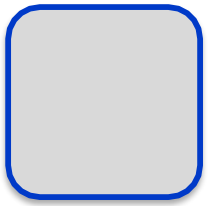
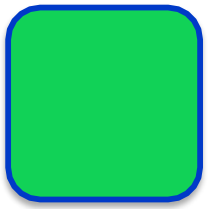


Fabric Ledger

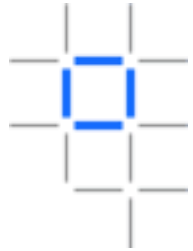
- The Fabric ledger is maintained by each peer and includes the blockchain and worldstate
- A separate ledger is maintained for each channel the peer joins
- Transaction read/write sets are written to the blockchain
- Channel configurations are also written to the blockchain
- The worldstate can be either LevelDB (default) or CouchDB
  - LevelDB is a simple key/value store
  - CouchDB is a document store that allows complex queries
- The smart contact Contract decides what is written to the worldstate



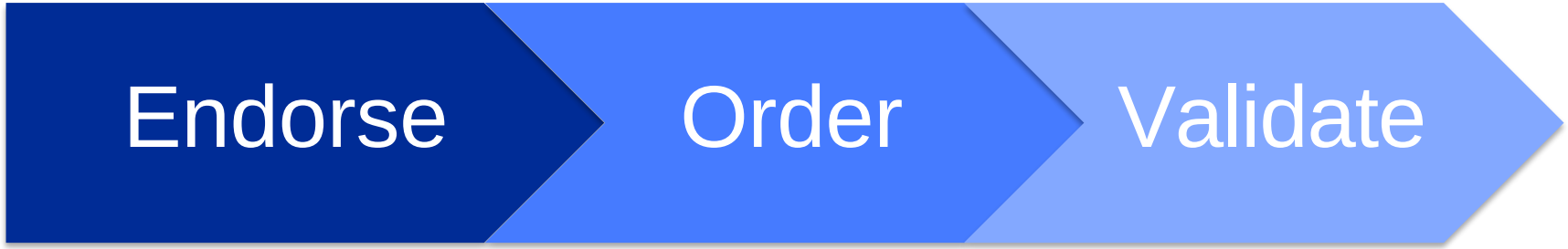
Nodes and roles

	**Committing Peer:** Maintains ledger and state. Commits transactions. May hold smart contract (chaincode).
	**Endorsing Peer:** Specialized peer also endorses transactions by receiving a transaction proposal and responds by granting or denying endorsement. Must hold smart contract.
	**Ordering Node:** Approves the inclusion of transaction blocks into the ledger and communicates with committing and endorsing peer nodes. Does not hold smart contract. Does not hold ledger.

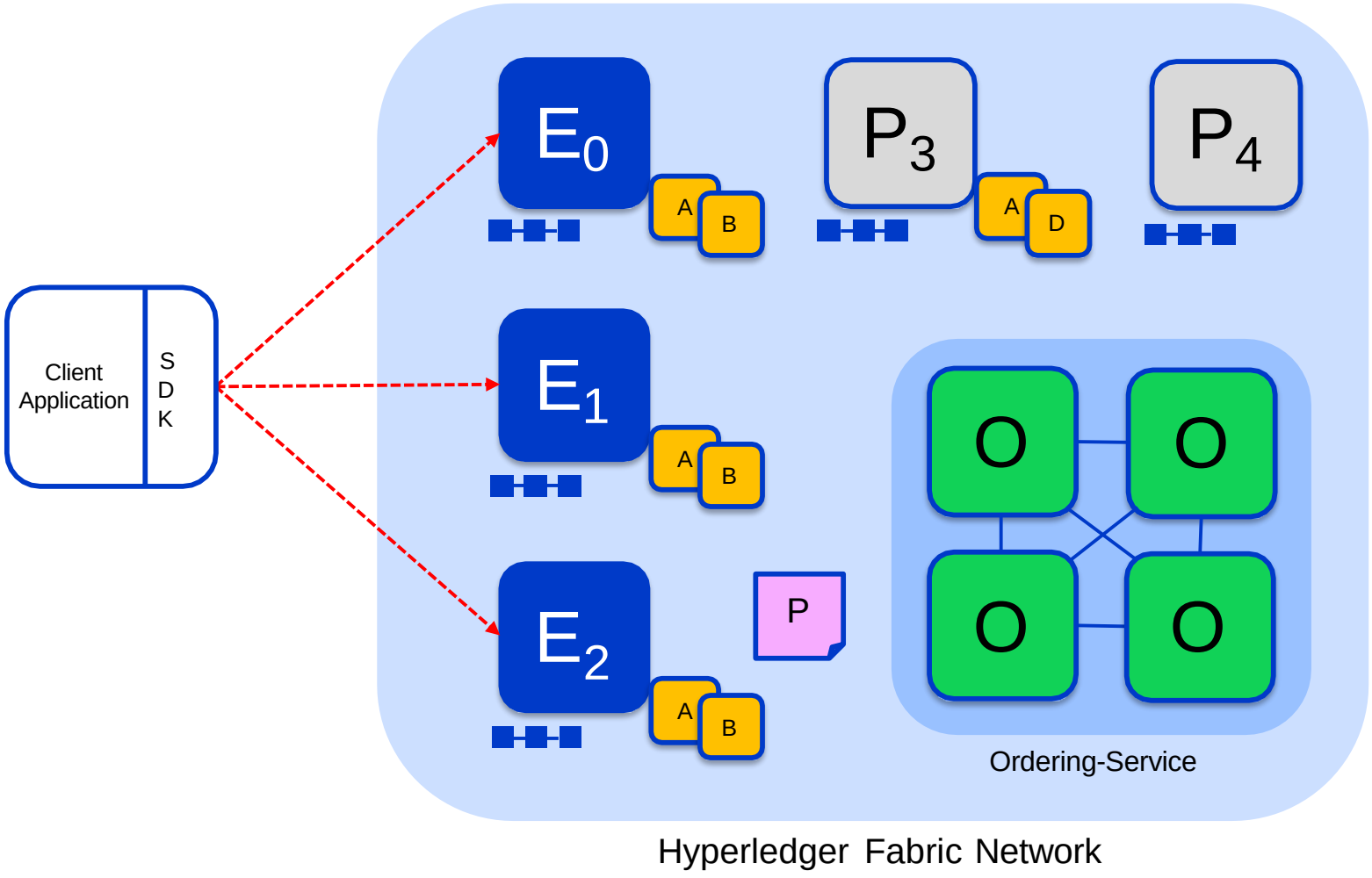
Hyperledger Fabric Consensus



Consensus is achieved using the following transaction flow:



Sample transaction: Step 1/7 – Propose transaction

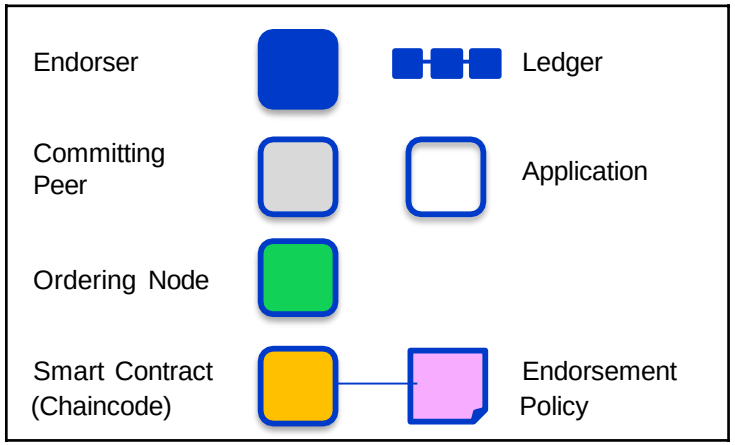


Application proposes transaction

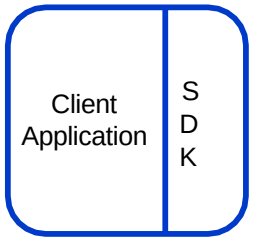
Endorsement policy:
- "E_0, E_1 and E_2 must sign"
- (P_3, P_4 are not part of the policy)

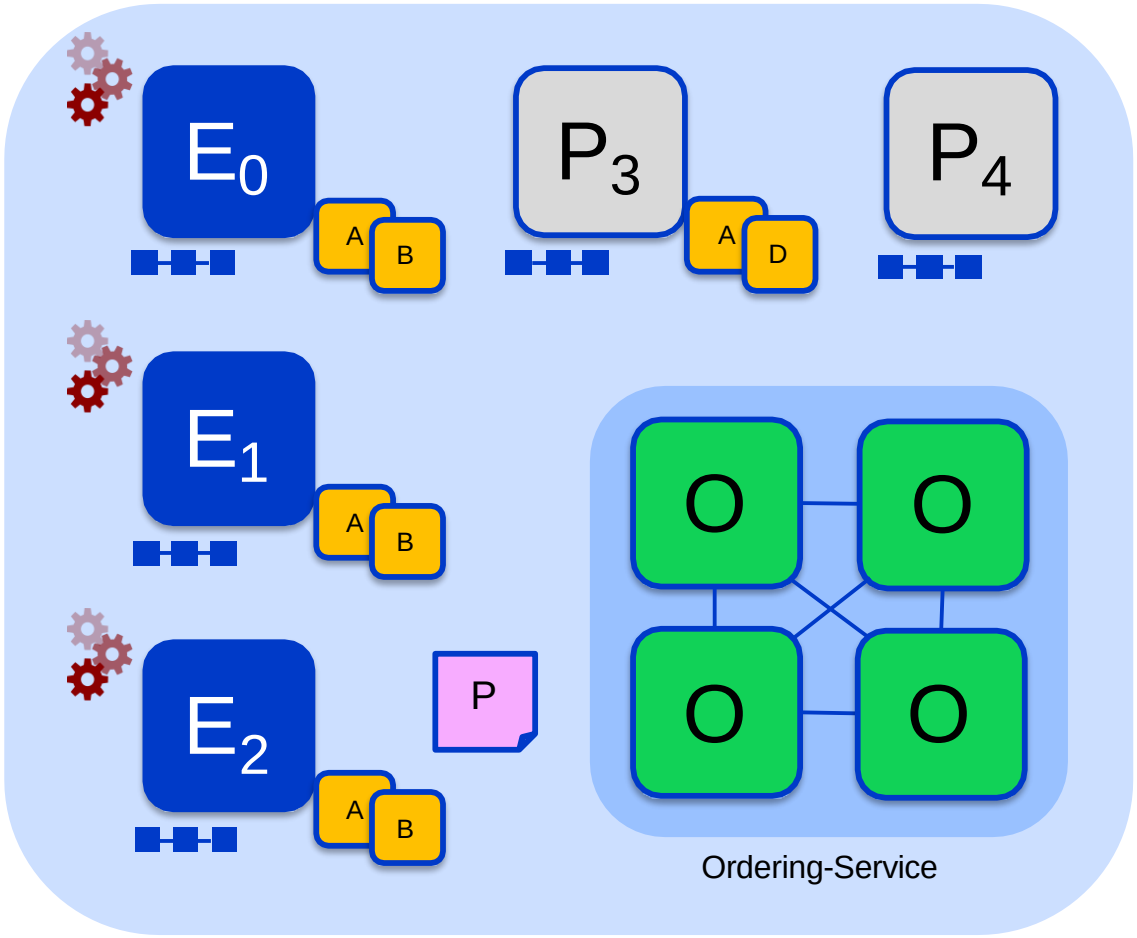
Client application submits a transaction proposal for Smart Contract A. It must target the required peers {E_0, E_1, E_2}

Key:



Sample transaction: Step 2/7 – Execute proposal





Hyperledger Fabric Network

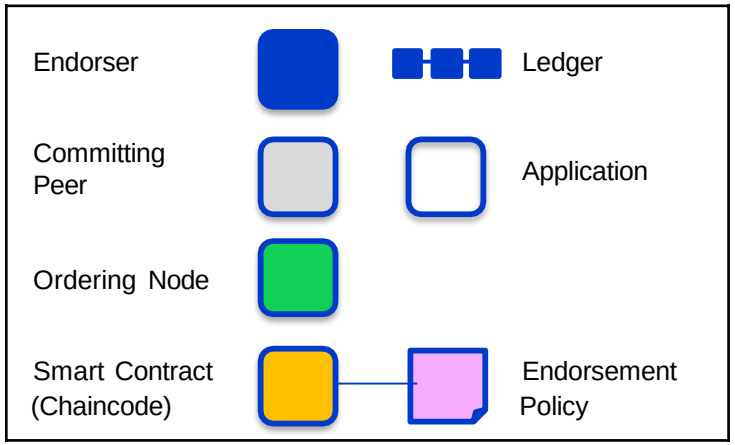
Endorsers Execute Proposals

E_0, E_1 & E_2 will each execute the proposed transaction. None of these executions will update the ledger

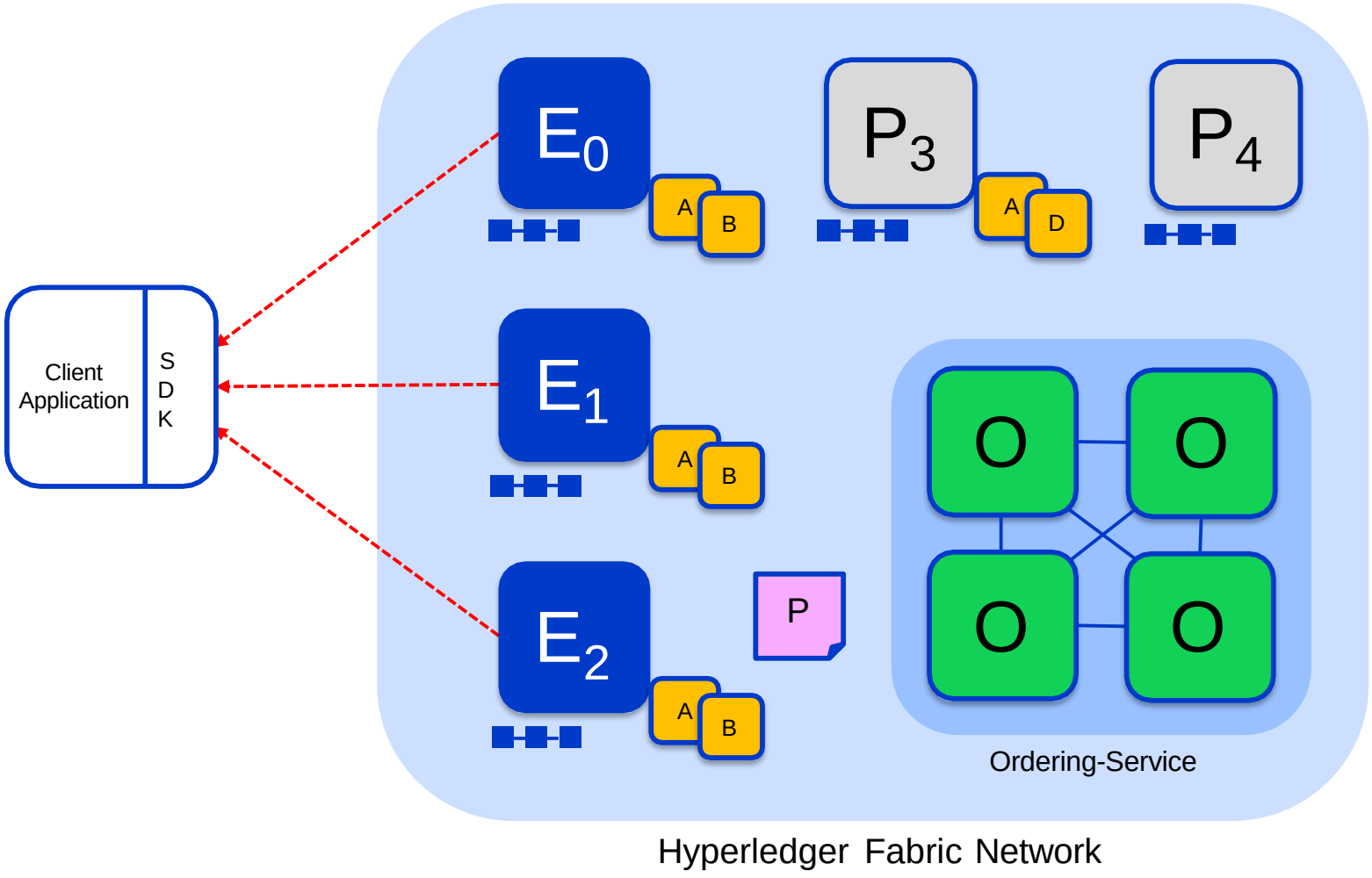
Each execution will capture the set of Read and Written data, called RW sets, which will now flow in the fabric.

Transactions can be signed & encrypted

Key:



Sample transaction: Step 3/7 – Proposal Response



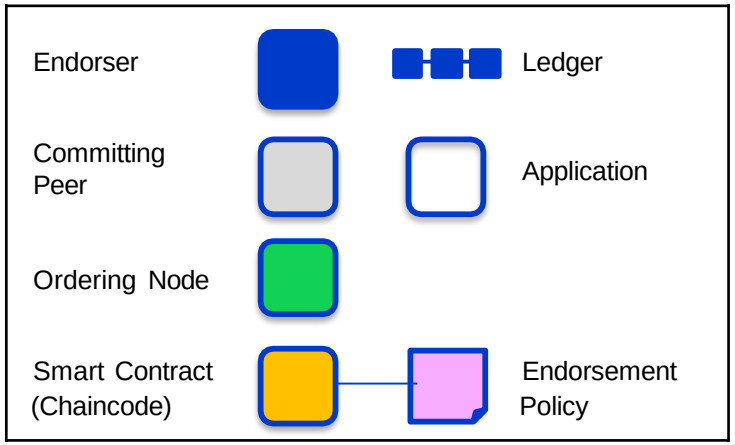
Application receives responses

RW sets are asynchronously returned to application

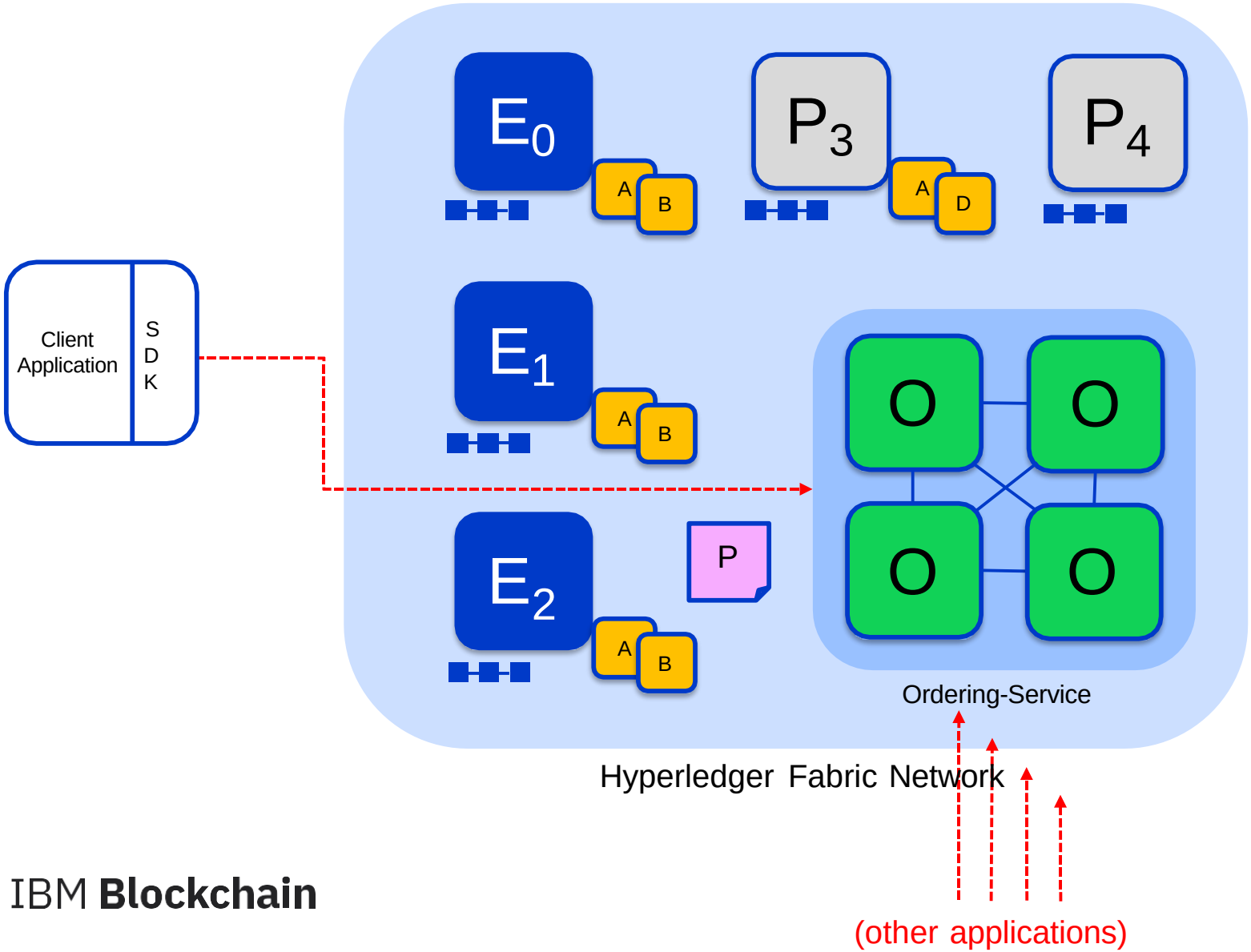
The RW sets are signed by each endorser, and also includes each record version number

(This information will be checked much later in the consensus process)

Key:



Sample transaction: Step 4/7 – Order Transaction

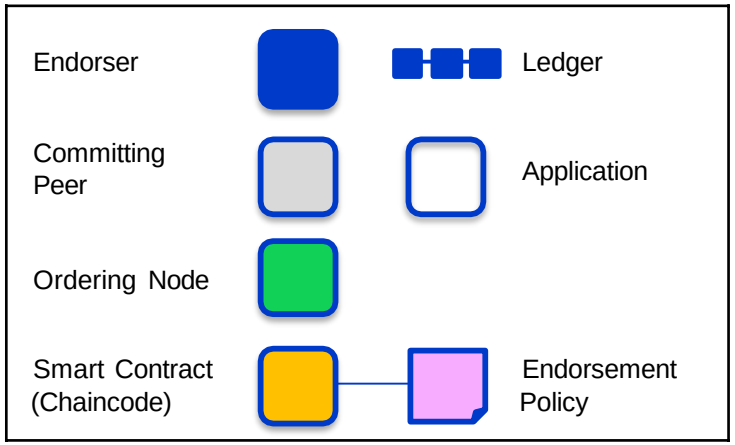


Responses submitted for ordering

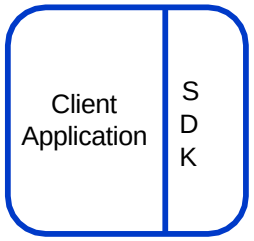
Application submits responses as a transaction to be ordered.

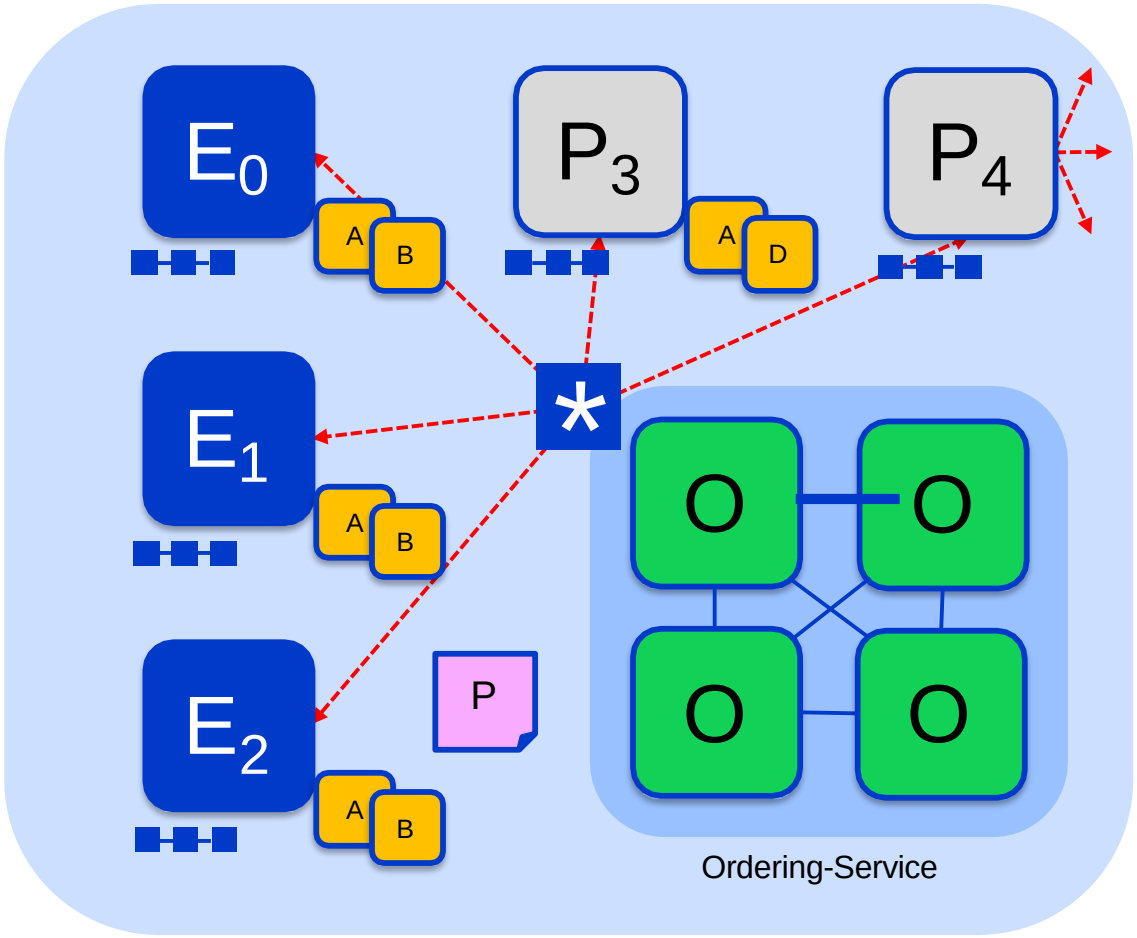
Ordering happens across the fabric in parallel with transactions submitted by other applications

Key:



Sample transaction: Step 5/7 – Deliver Transaction





Hyperledger Fabric Network

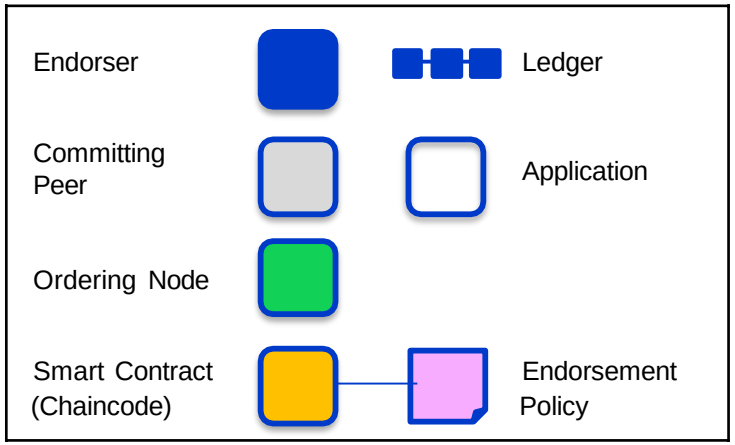
Orderer delivers to committing peers

Ordering service collects transactions into proposed blocks for distribution to committing peers. Peers can deliver to other peers in a hierarchy

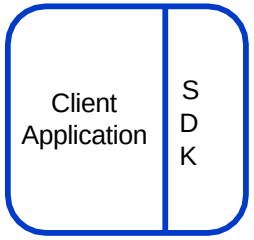
Different ordering algorithms available:

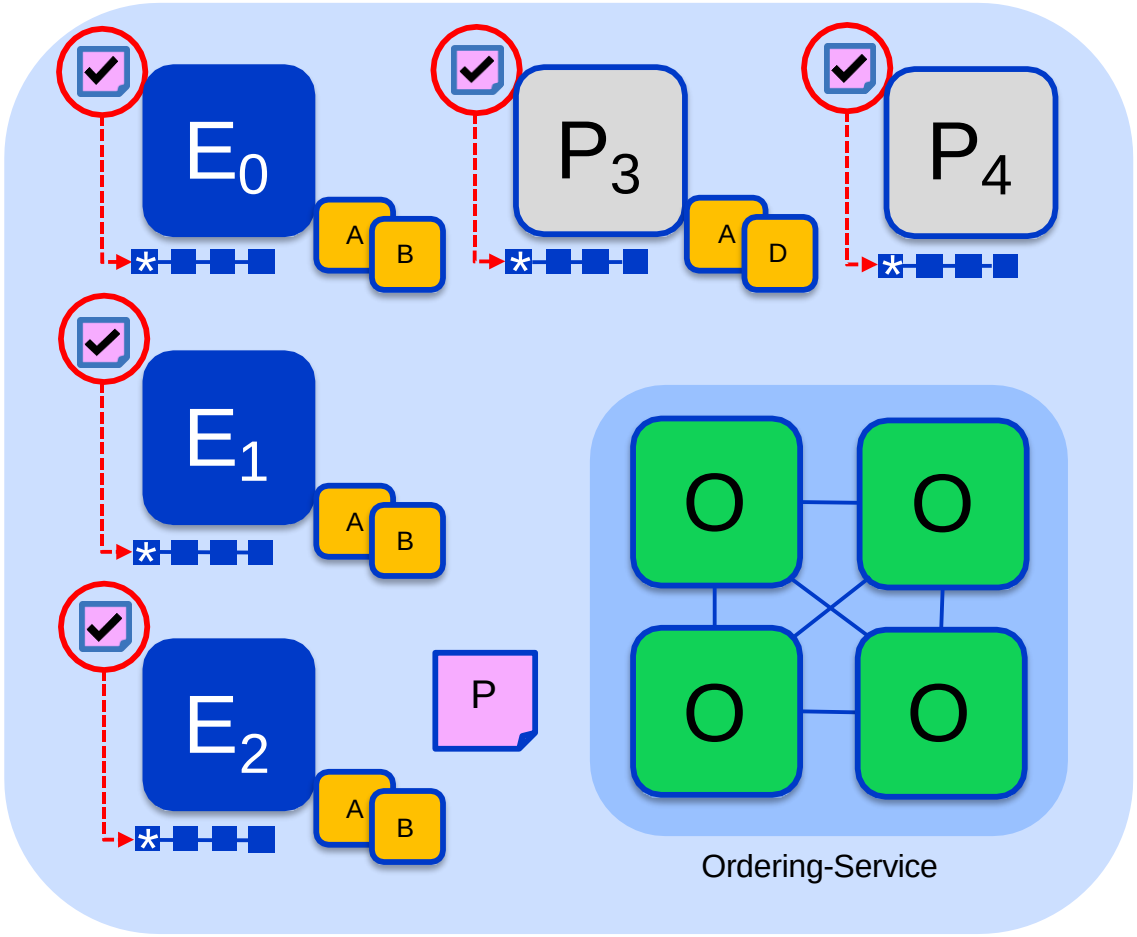
- SOLO (Single node, development)
- Kafka (Crash fault tolerant)
- Raft (Crash fault tolerant)

Key:



Sample transaction: Step 6/7 – Validate Transaction





Hyperledger Fabric Network

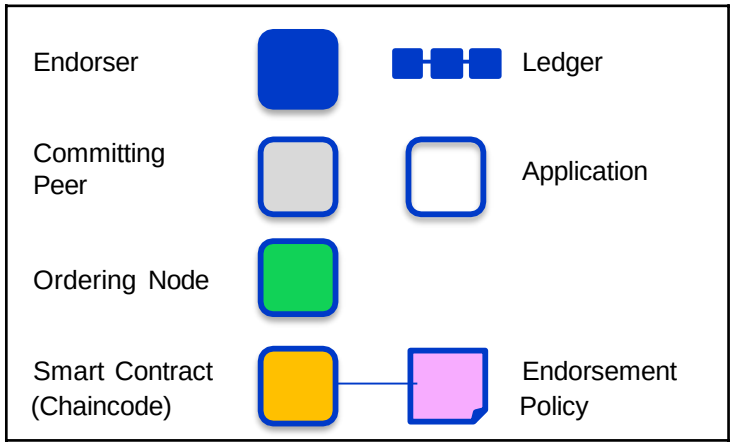
Committing peers validate transactions

Every committing peer validates against the endorsement policy. Also check RW sets are still valid for current world state

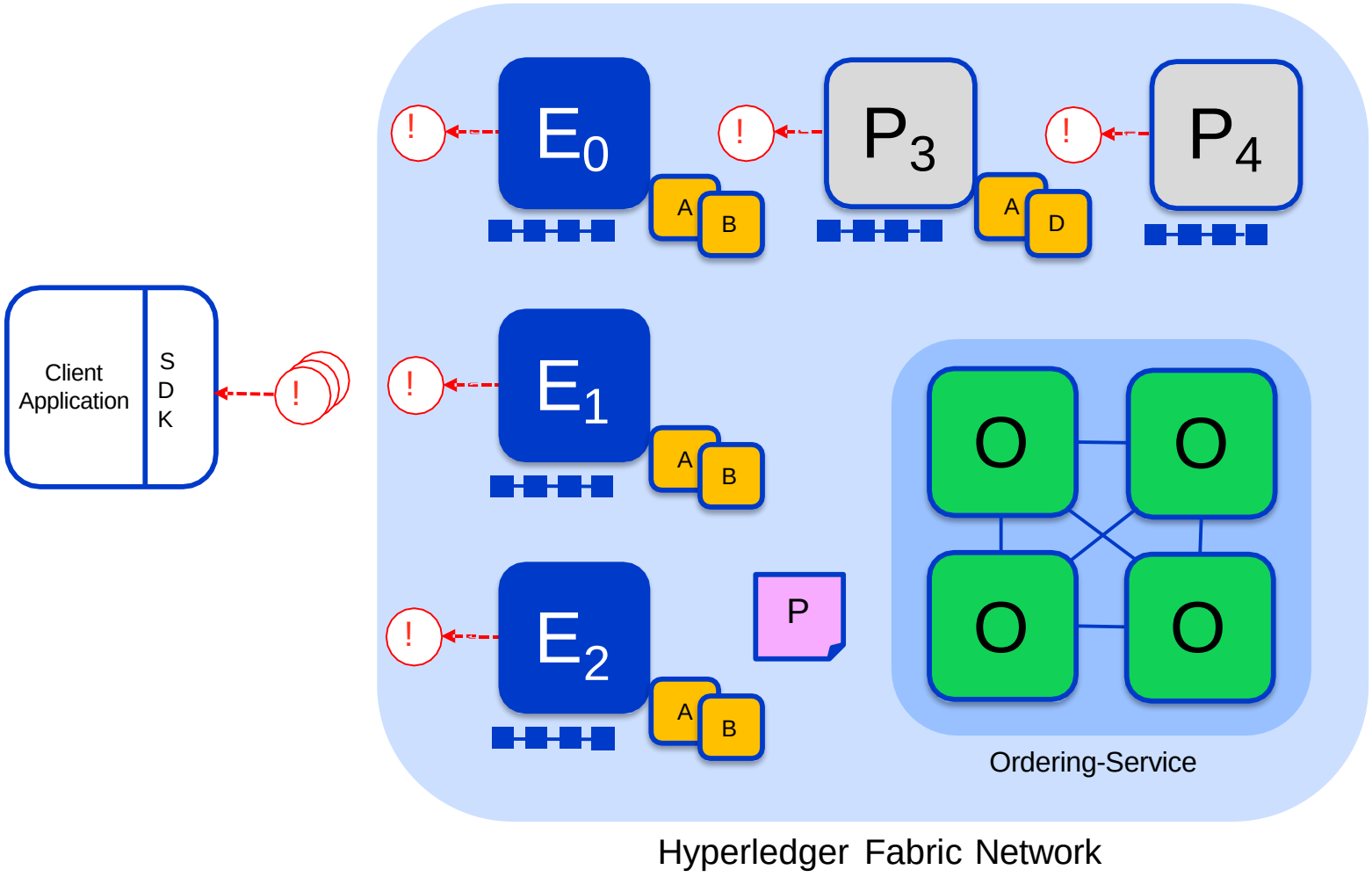
Validated transactions are applied to the world state and retained on the ledger

Invalid transactions are also retained on the ledger but do not update world state

Key:



Sample transaction: Step 7/7 – Notify Transaction



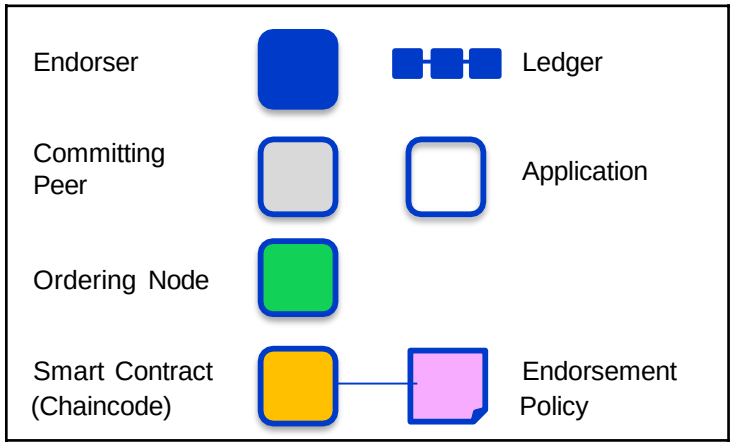
Hyperledger Fabric Network

Committing peers notify applications

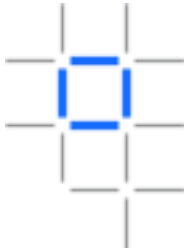
Applications can register to be notified when transactions succeed or fail, and when blocks are added to the ledger

Applications will be notified by each peer to which they are connected

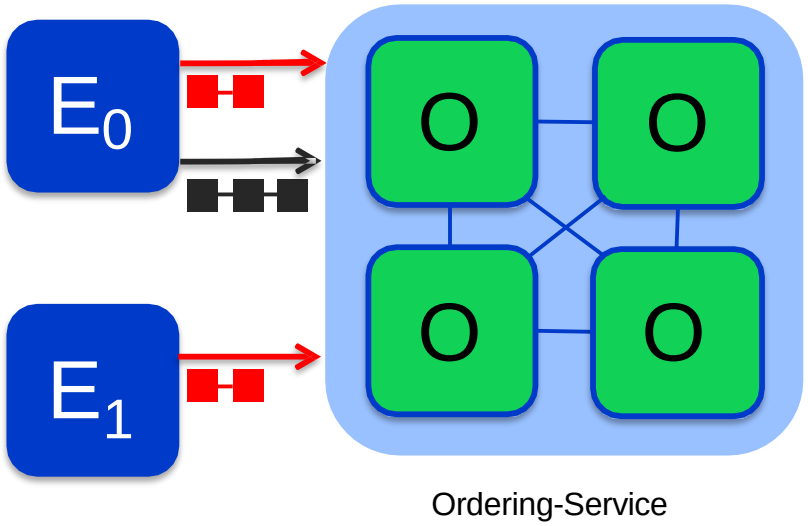
Key:



Channels



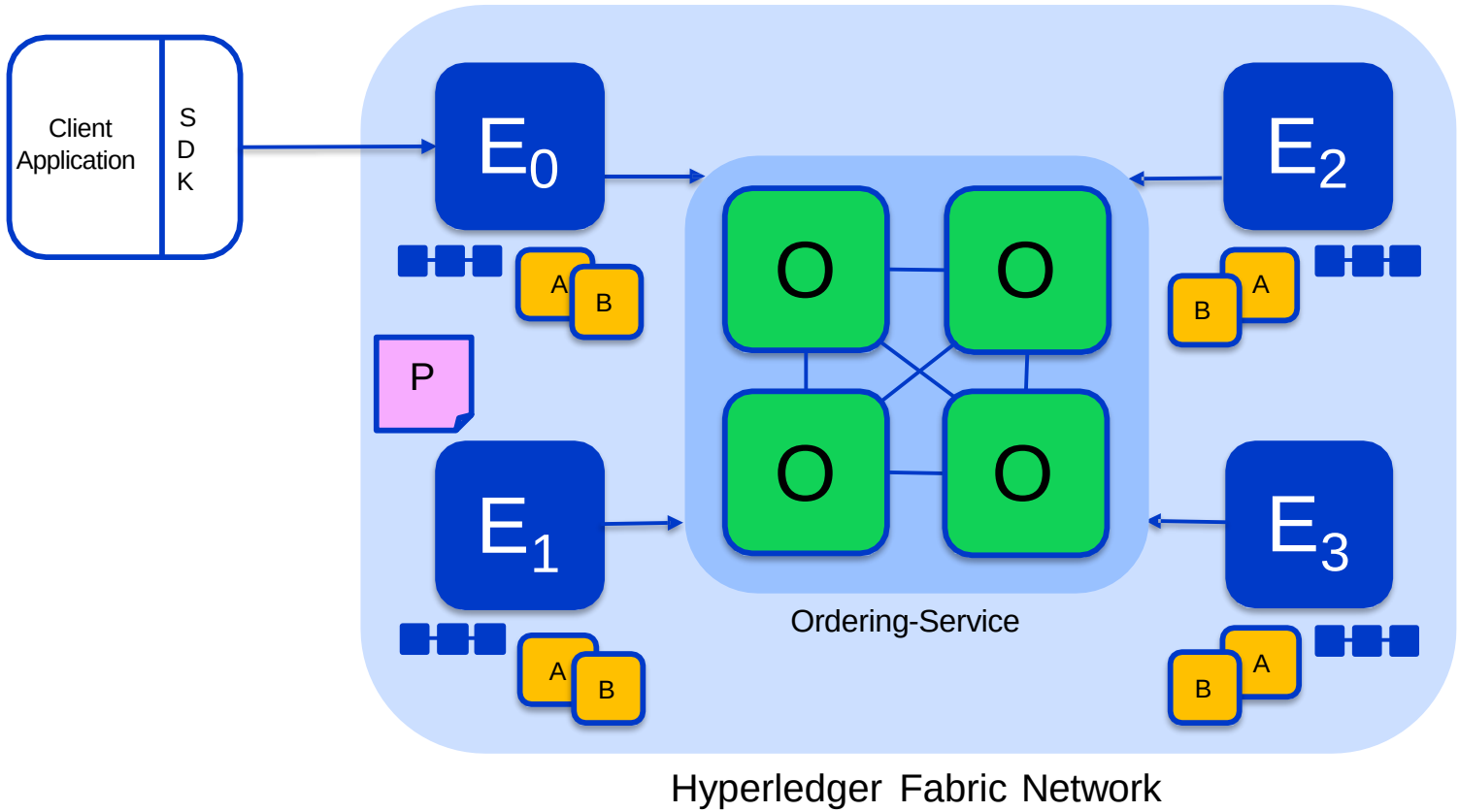
Channels provide privacy between different ledgers



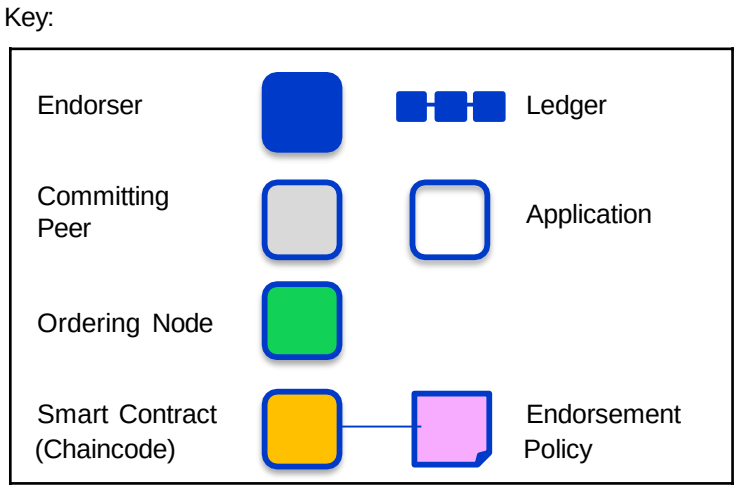
Ordering-Service

- Ledgers exist in the scope of a channel
  - Channels can be shared across an entire network of peers
  - Channels can be permissioned for a specific set of participants
- Chaincode is installed on peers to access the worldstate
- Chaincode is instantiated on specific channels
- Peers can participate in multiple channels
- Concurrent execution for performance and scalability

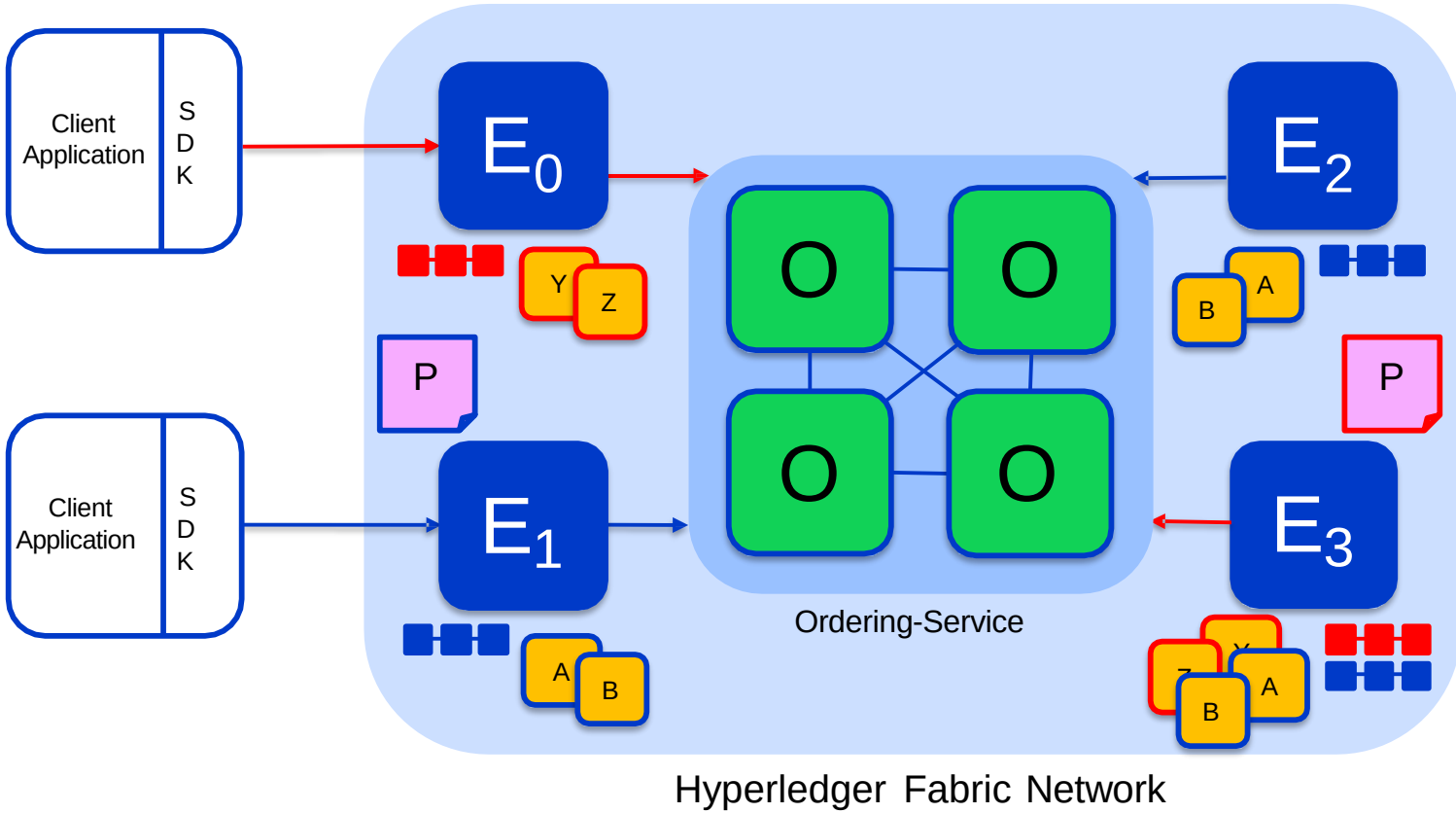
Single Channel Network



- All peers connect to the same system channel (blue).
- All peers have the same chaincode and maintain the same ledger
- Endorsement by peers $E_{0,}$ $E_{1,}$ E_2 and E_3

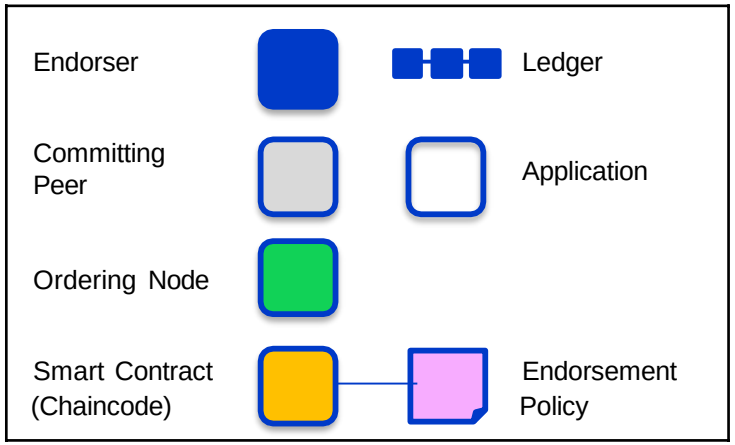


Multi Channel Network



- Peers E_0 and E_3 connect to the red channel for chaincodes Y and Z
- E_1, E_2 and E_3 connect to the blue channel for chaincodes A and B

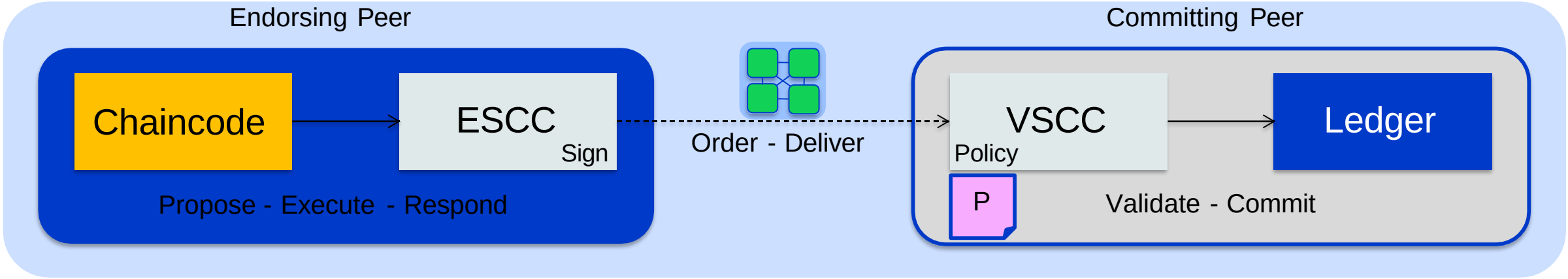
Key:



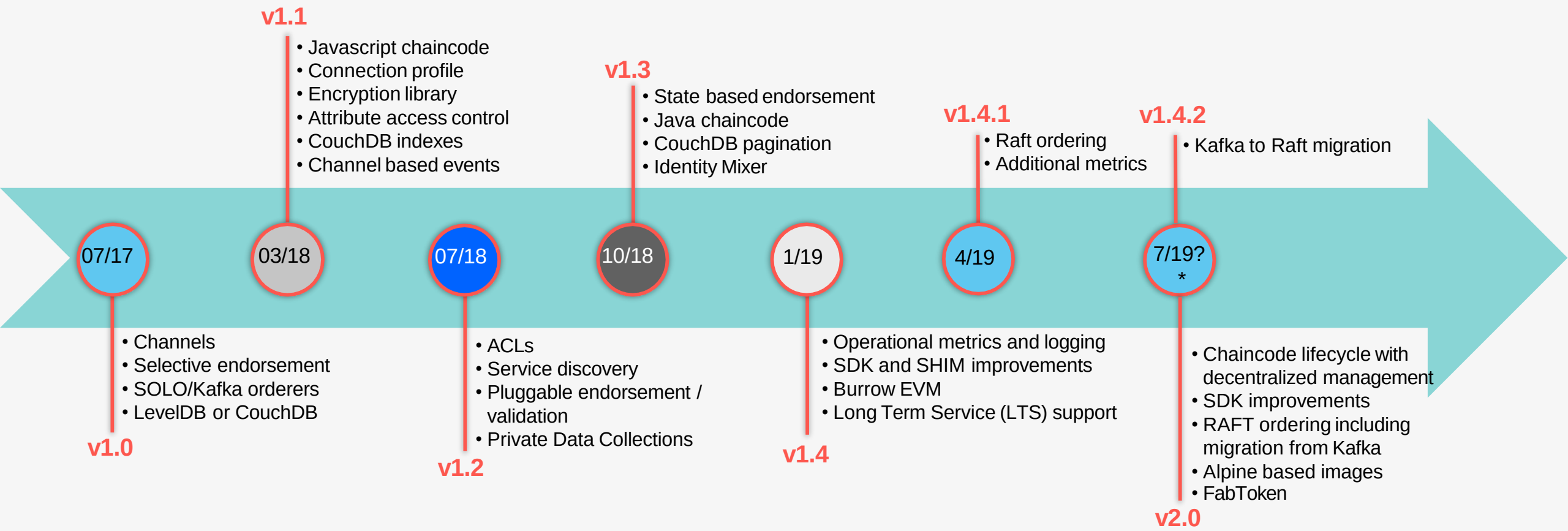
Endorsement Policies

An endorsement policy describes the conditions by which a transaction can be endorsed. A transaction can only be considered valid if it has been endorsed according to its policy.

- Each chaincode is deployed with an Endorsement Policy
- **ESCC** (Endorsement System Chaincode) signs the proposal response on the endorsing peer
- **VSCC** (Validation System Chaincode) validates the endorsements

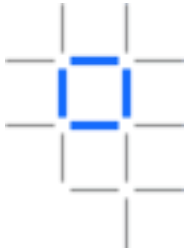


Roadmap



Based on https://wiki.hyperledger.org/display/fabric/Hyperledger+Fabric+Roadmap - Dates determined by the Hyperledger community - (*) Subject to change

Getting started with Hyperledger Fabric



- Build Your First Network (BYFN) – Network administrator
  - A simple network with 2 organizations running 2 peers, with one channel, a simple chaincode
  - Dockerhub images
  - Uses predefined enrollment certificates and « Solo » Ordering Service
- Extend Your First Network – Network administrator
  - Adds a 3rd organization to BYFN
- Develop Your First Application – Application developer
  - A simple Node.js application
- Start in devmode (minimal set up), then move to network (several peers), and security (membersrvc)
- Several examples to start from (fabcar)

IBM Blockchain Platform

IBM Blockchain Platform is a fully integrated enterprise-ready blockchain platform designed to accelerate the development, governance, and operation of a multi-institution business network

- **Developer tools** to quickly build your blockchain application
- Hyperledger Fabric provides the ledger, which is managed through a set of intuitive **operational tools**
- **Governance tools** for democratic management of the business network
- Flexible deployment options, including a highly secure and performant **IBM Cloud** environment