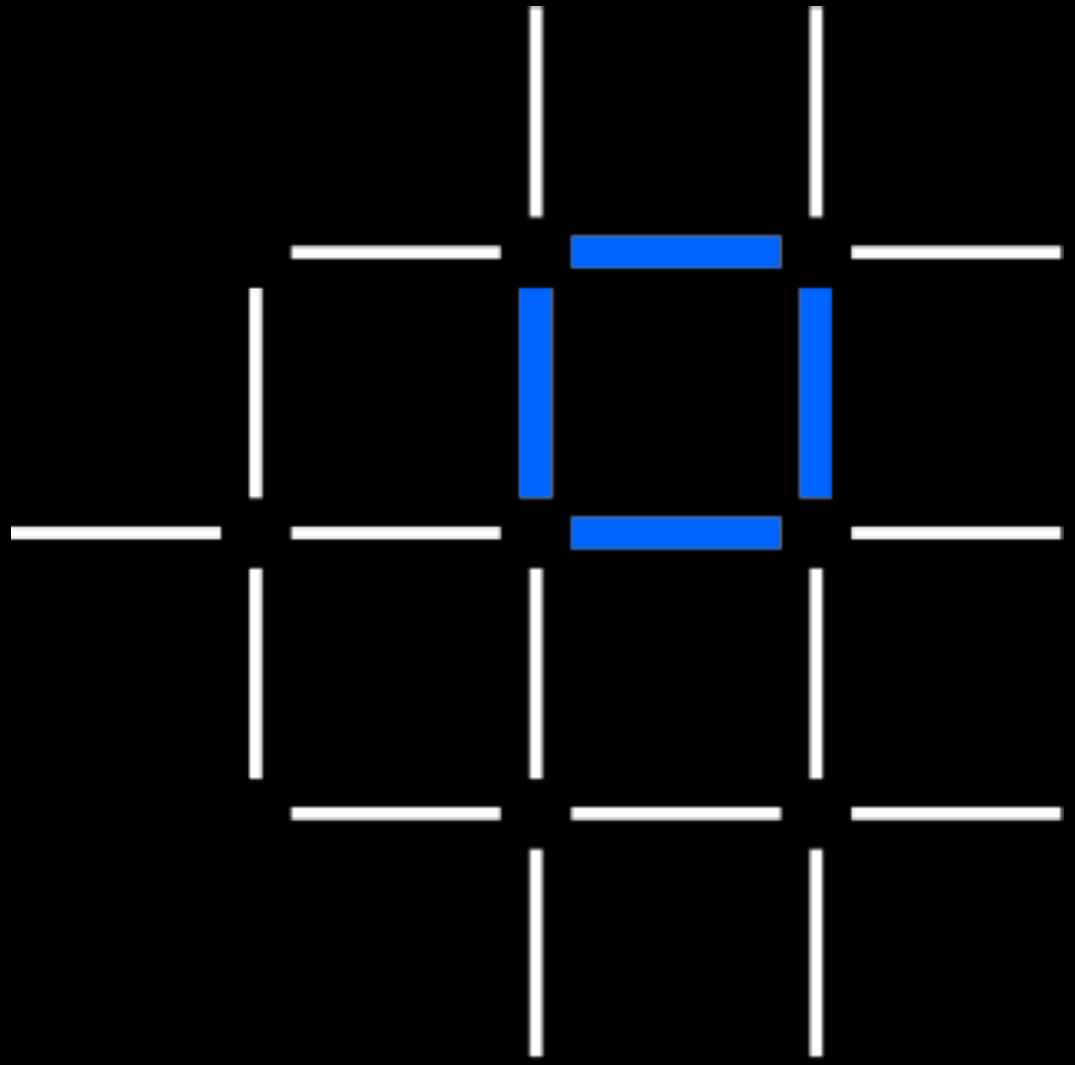
Hyperledger Fabric Fundamentals



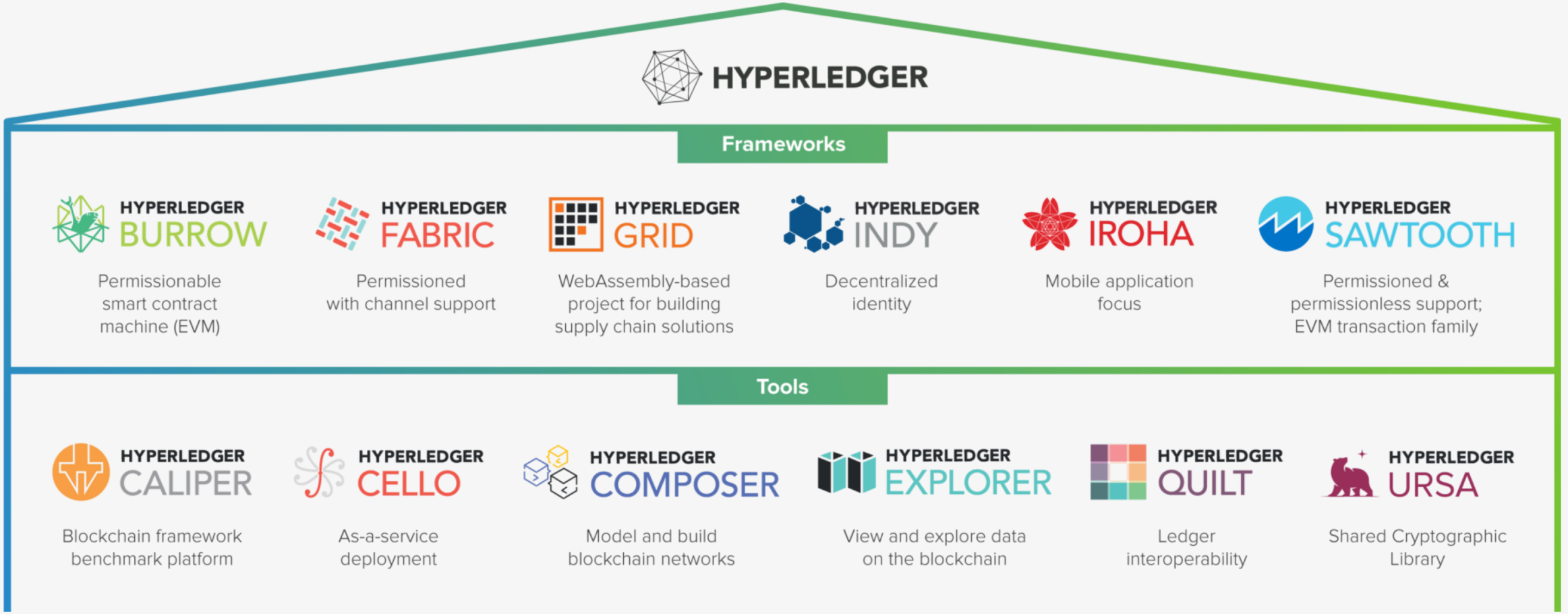
Swetha Repakula
Open Source Contributor, IBM Open Technologies
srepaku@us.ibm.com
@swayr93

Morgan Bauer
Open Source Contributor, IBM Open Technologies
mbauer@us.ibm.com
@ibmhb

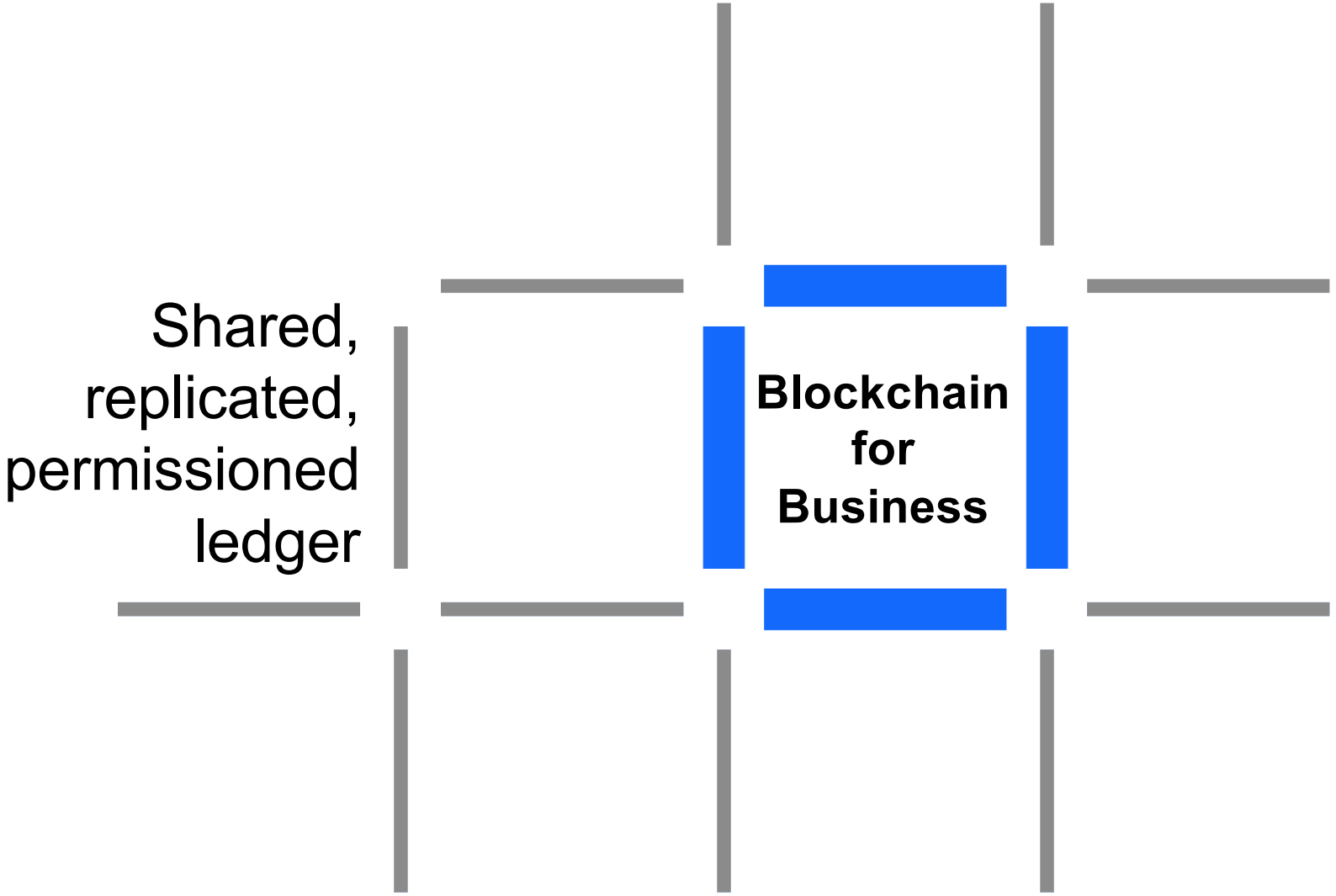
IBM **Blockchain**

IBM

Hyperledger Modular Approach



Introducing Blockchain for Business…



Food Trust

What?

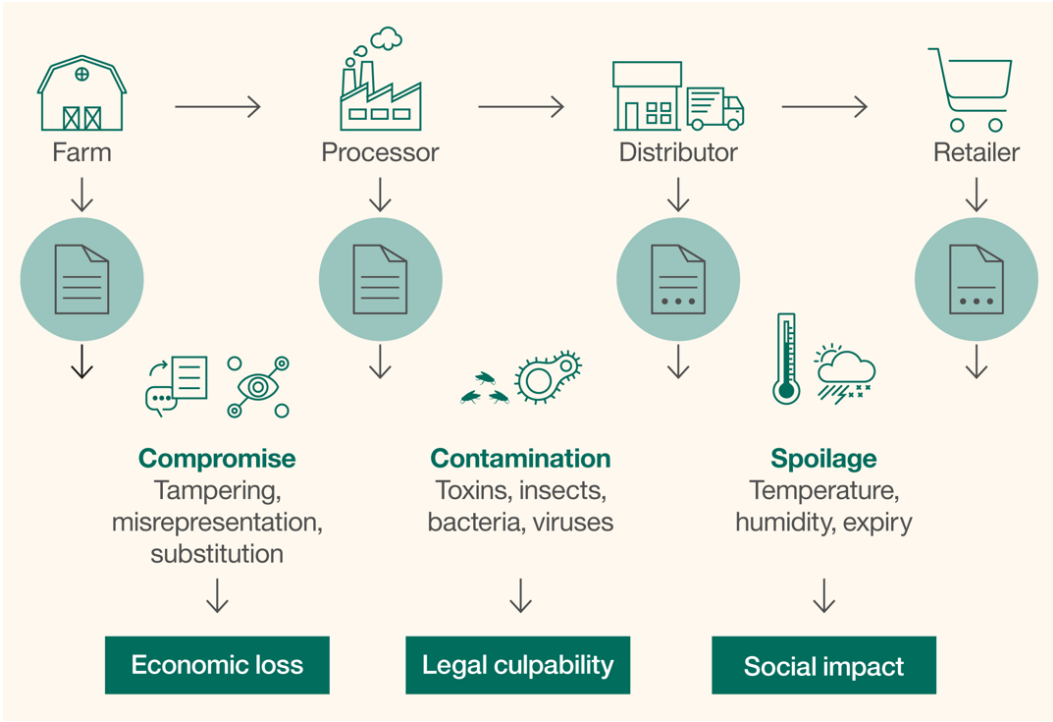
- Provide a trusted source of information and traceability to improve transparency and efficiency across the food network.

How?

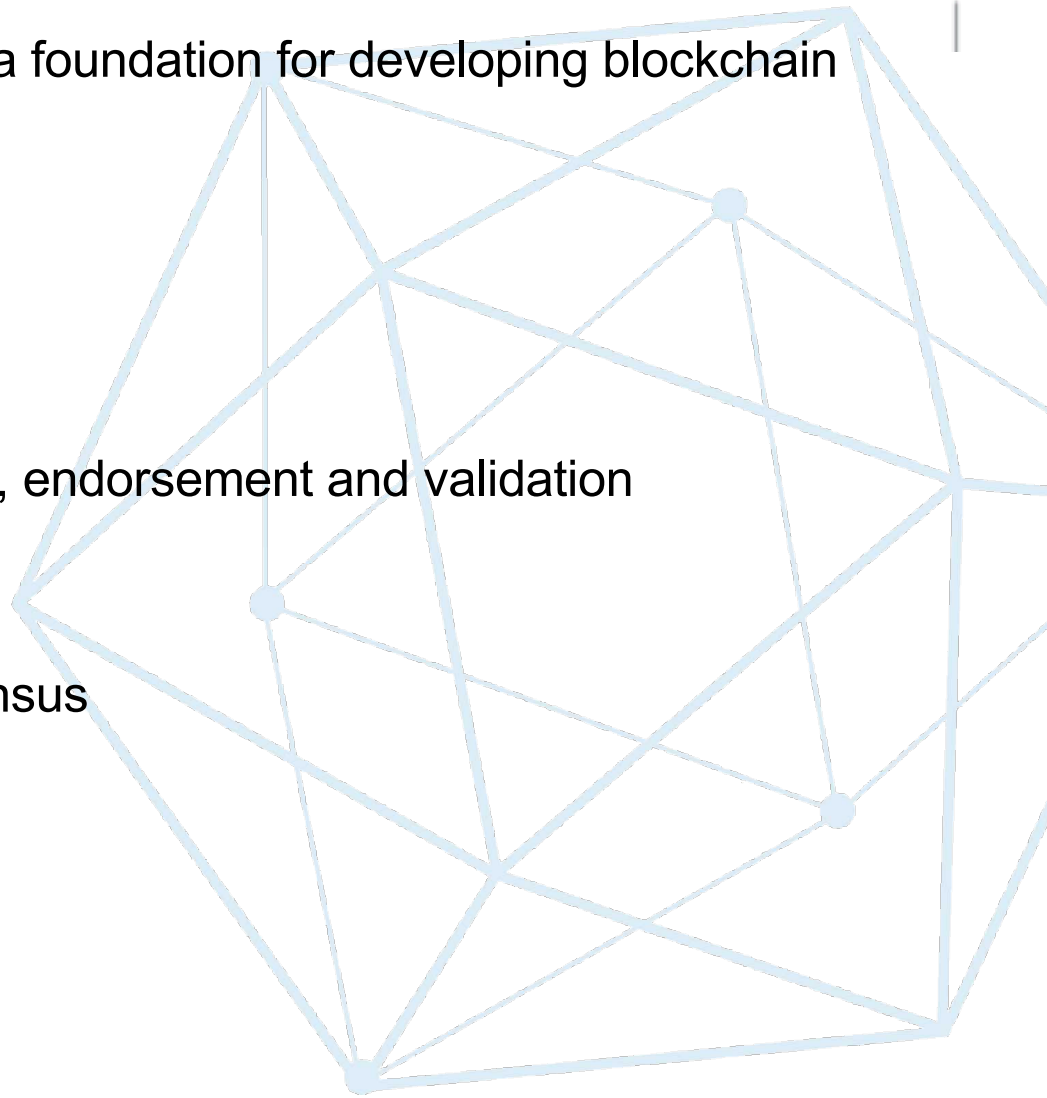
- Shared ledger for storing digital compliance documentation, test results and audit certificates network.

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



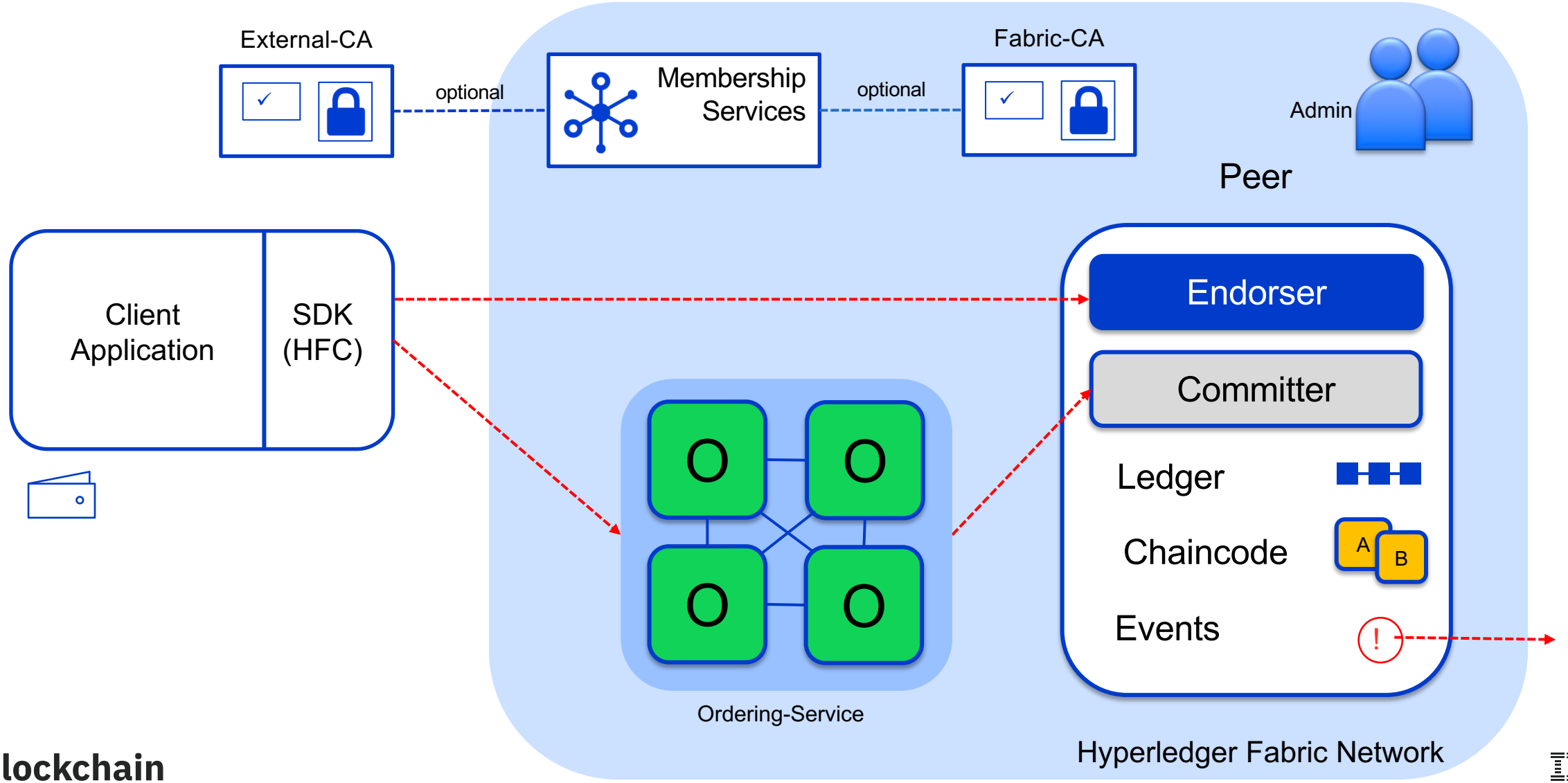
- An implementation of blockchain technology that is intended as a foundation for developing blockchain applications for the enterprise

- 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

Technical Deep Dive

- [Architectural Overview]
- Network Consensus
- Channels and Ordering Service
- Components
- Network setup
- Endorsement Policies
- Membership Services
- Roadmap

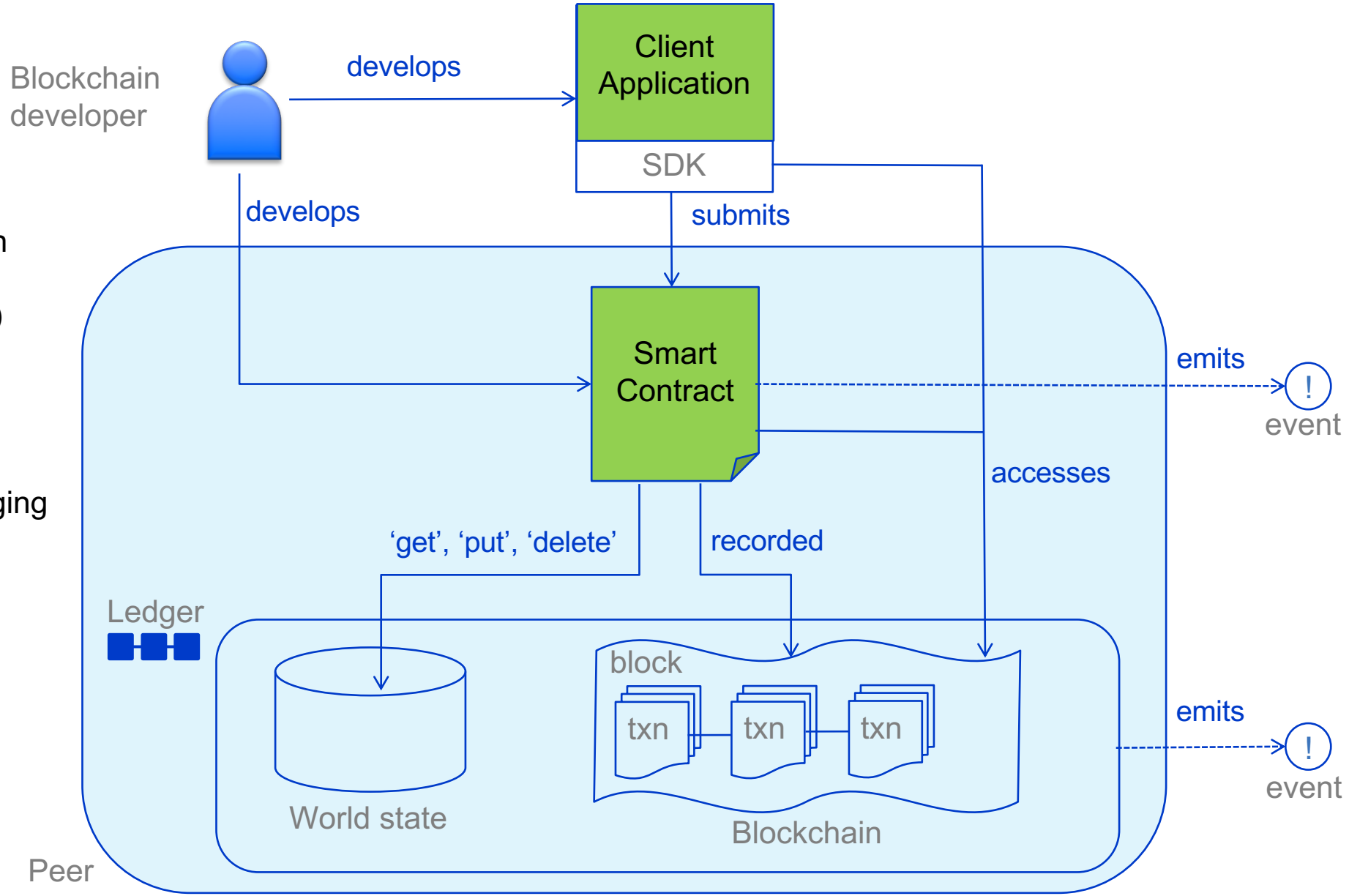
Hyperledger Fabric V1 Architecture



How applications interact with the ledger

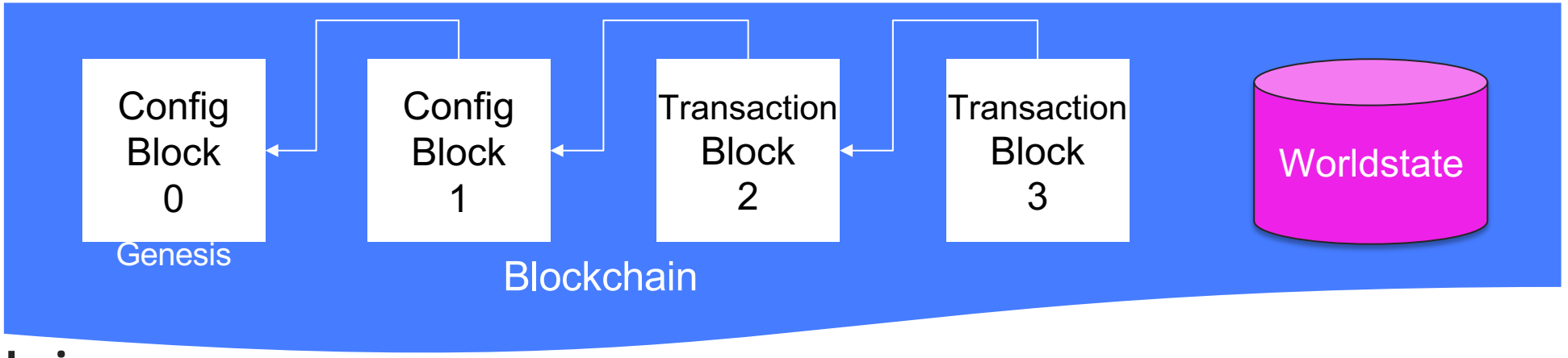
1. Client Application in using Hyperledger Fabric Client (HFC) SDK

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



Fabric Ledger

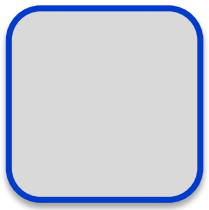
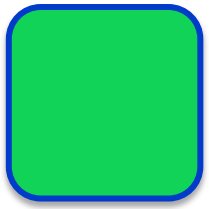
- The Fabric ledger is maintained by each peer and includes the blockchain and worldstate
- A separate ledger is maintained for each channel the peer joins
- Transaction read/write sets are written to the blockchain
- Channel configurations are also written to the blockchain
- The worldstate can be either 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



Technical Deep Dive

- Architectural Overview
- [Network Consensus]
- Channels and Ordering Service
- Components
- Network setup
- Endorsement Policies
- Membership Services
- Roadmap

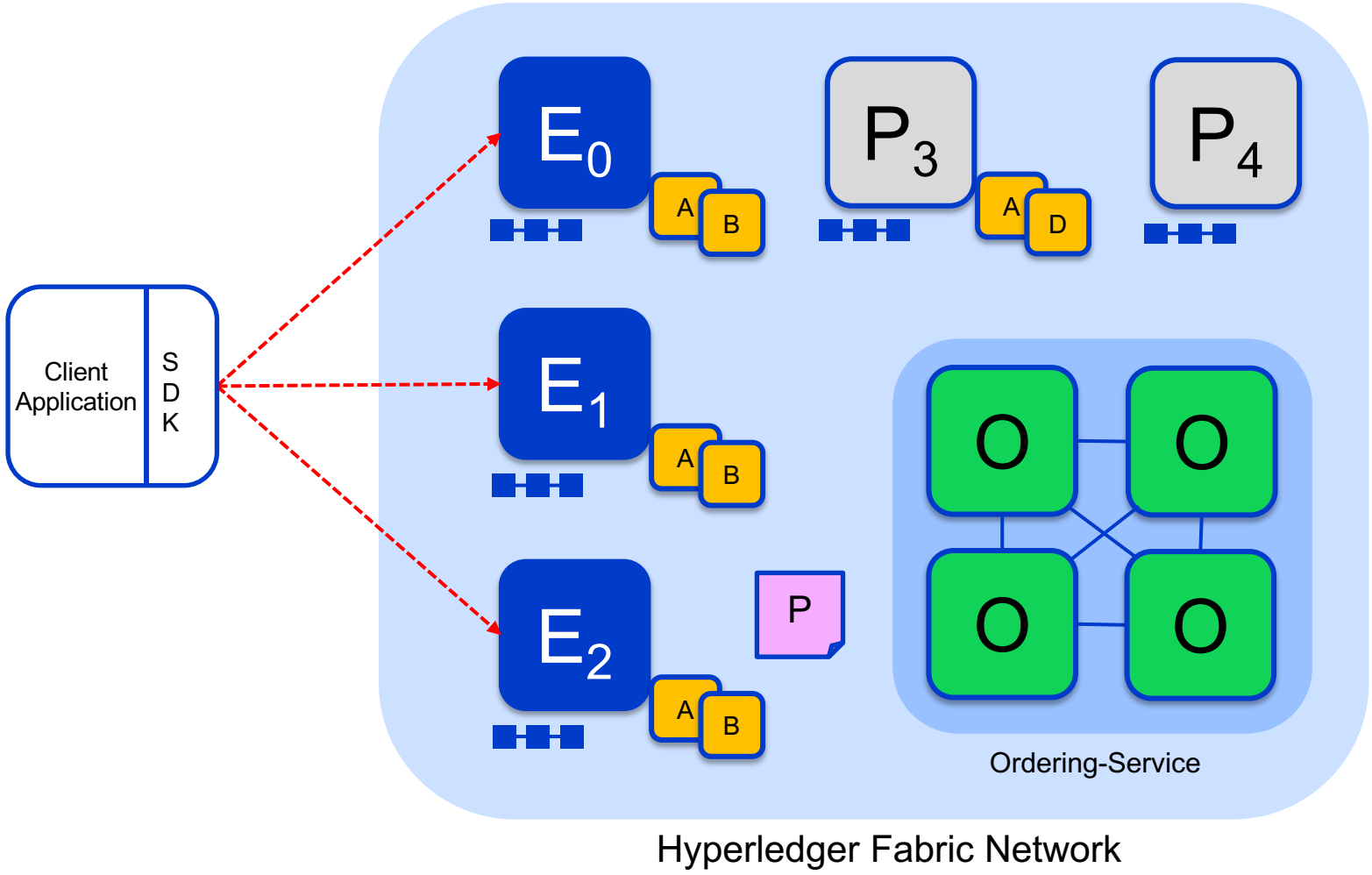
Nodes and roles

	**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



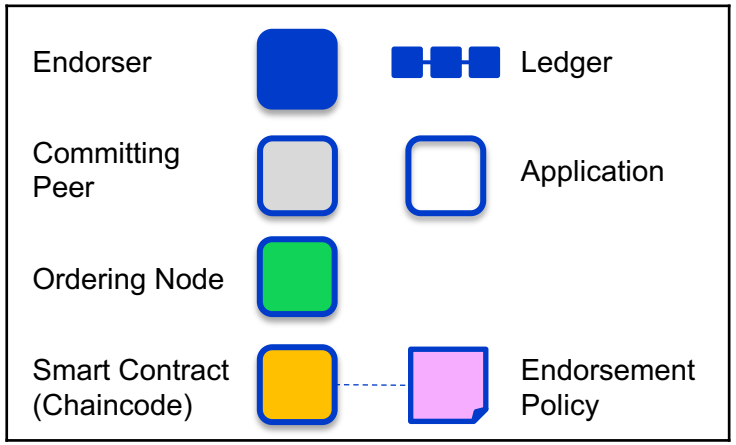
Hyperledger Fabric Network

Application proposes transaction

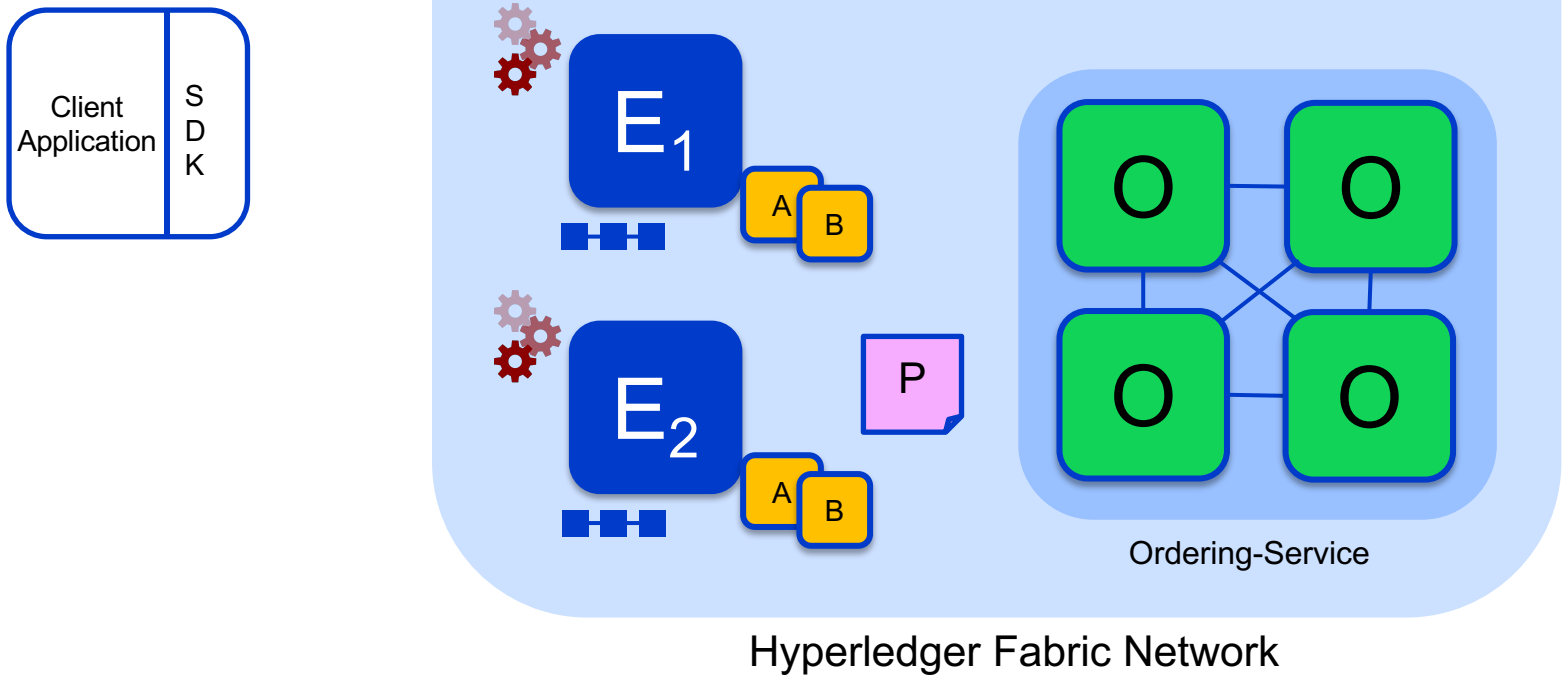
Endorsement policy:
- "E_0, E_1 and E_2 must sign"
- (P_3, P_4 are not part of the policy)

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

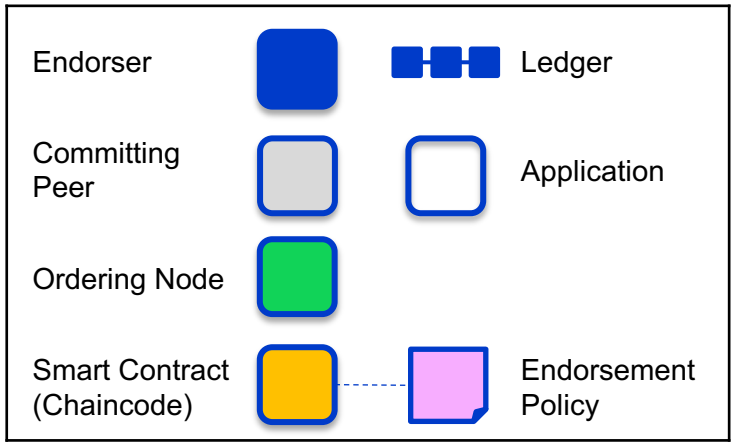
Endorsers Execute Proposals

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

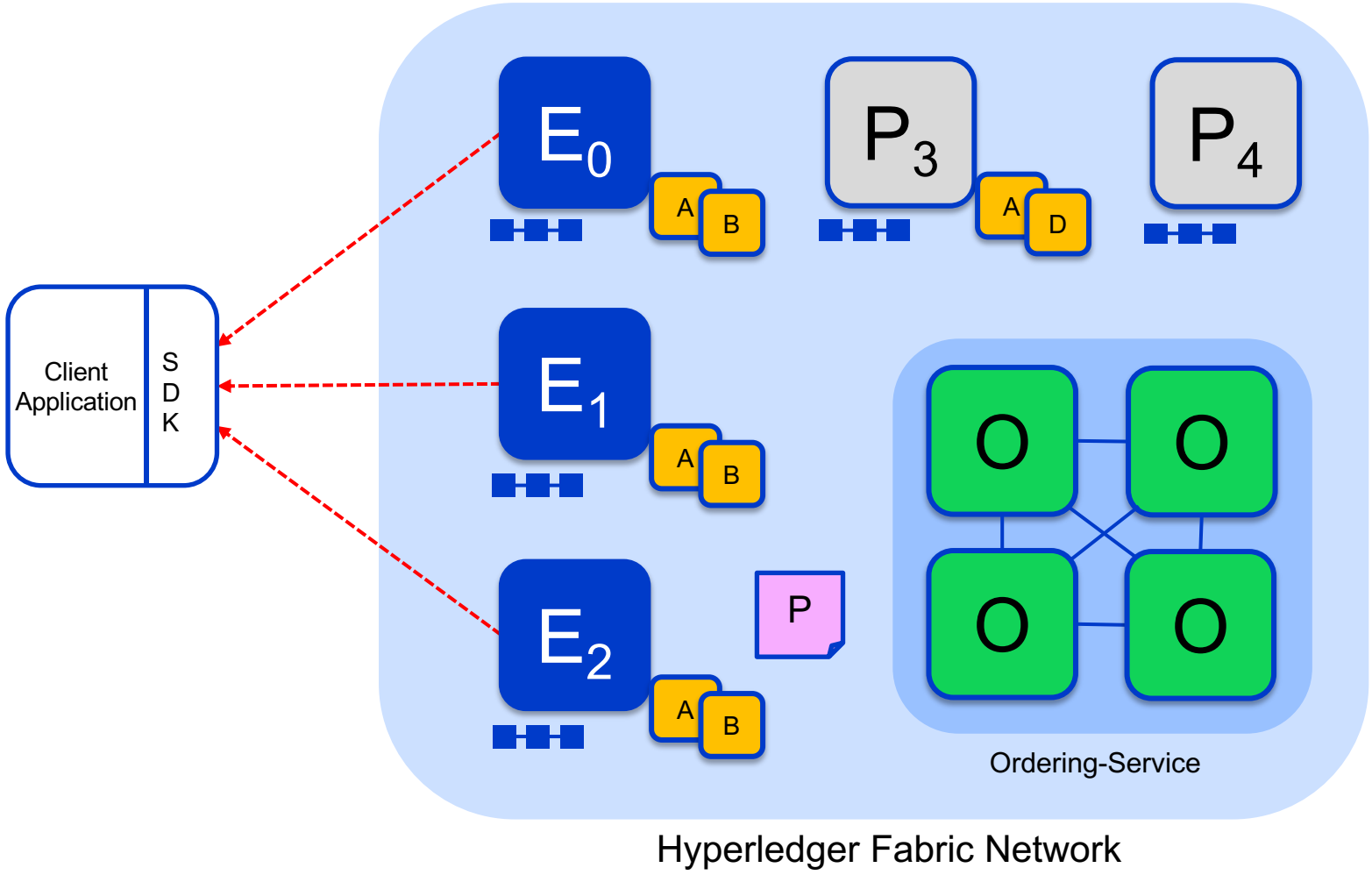
Each execution will capture the set of Read and Written data, called RW sets, which will now flow in the fabric.

Transactions can be signed & encrypted

Key:



Sample transaction: Step 3/7 – Proposal Response



Hyperledger Fabric Network

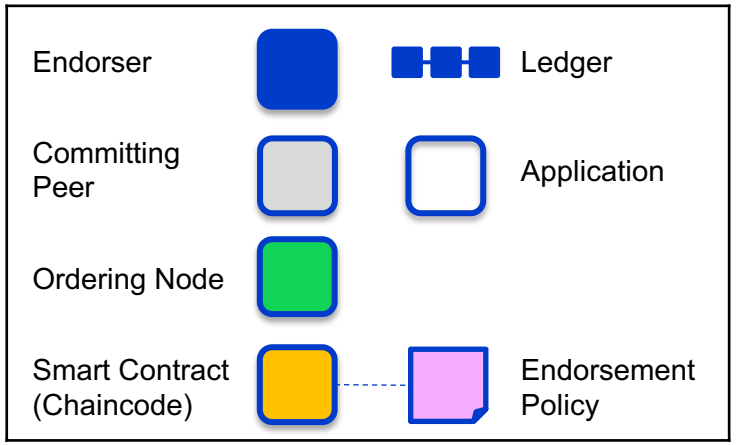
Application receives responses

RW sets are asynchronously returned to application

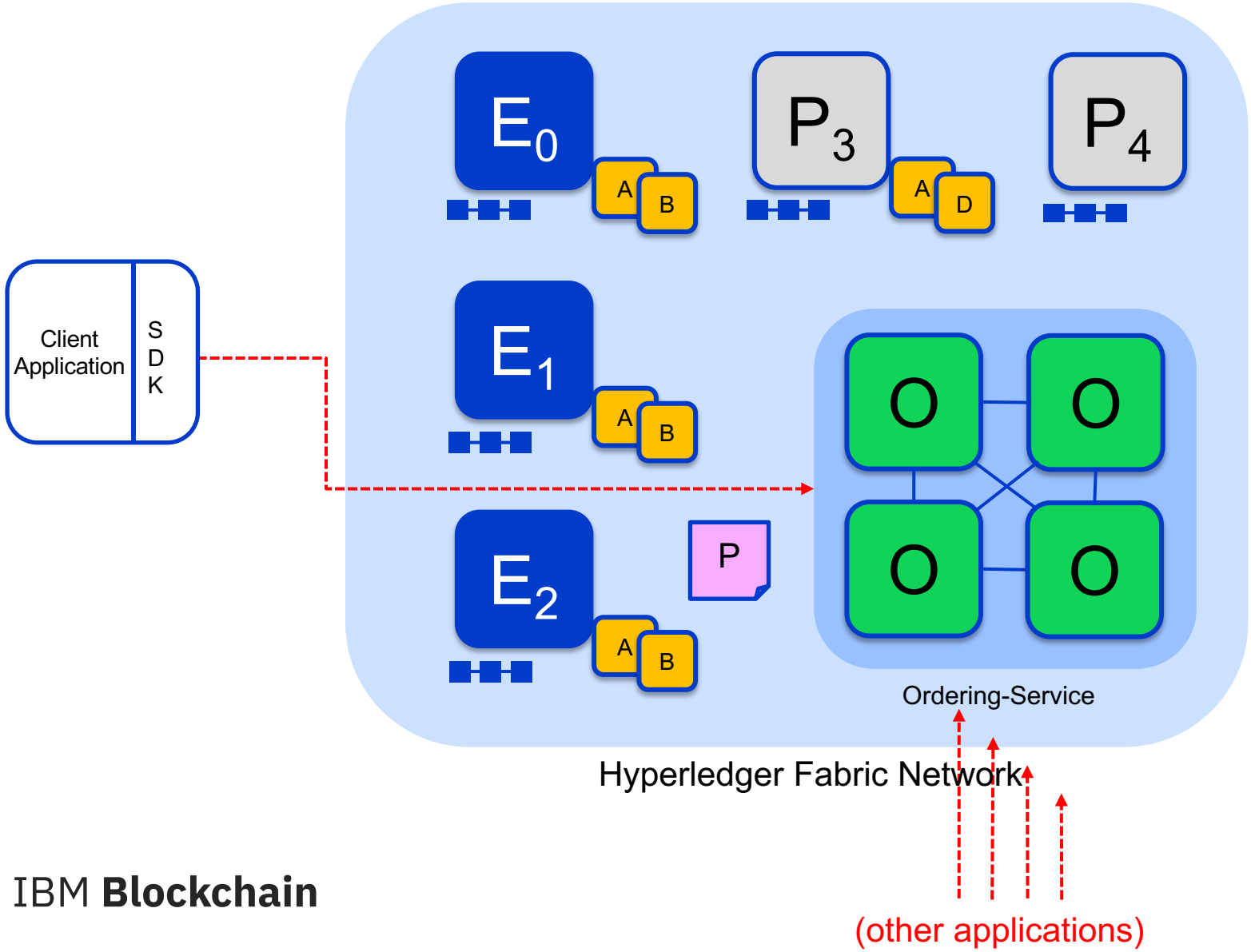
The RW sets are signed by each endorser, and also includes each record version number

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

Key:



Sample transaction: Step 4/7 – Order Transaction

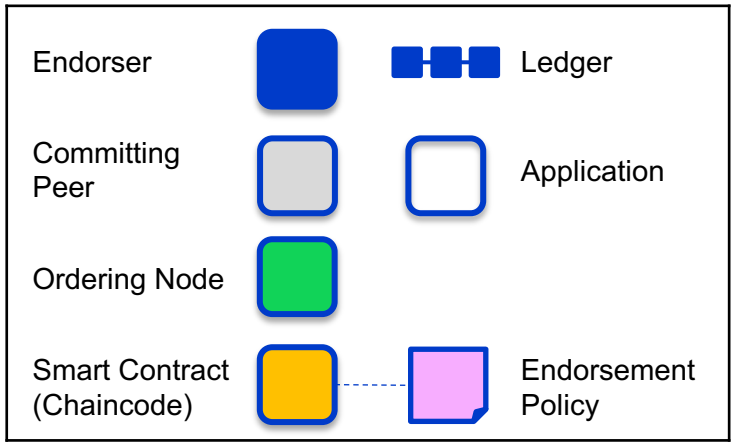


Responses submitted for ordering

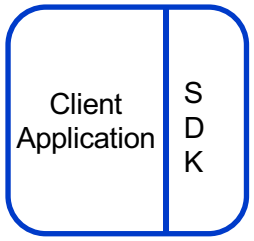
Application submits responses as a transaction to be ordered.

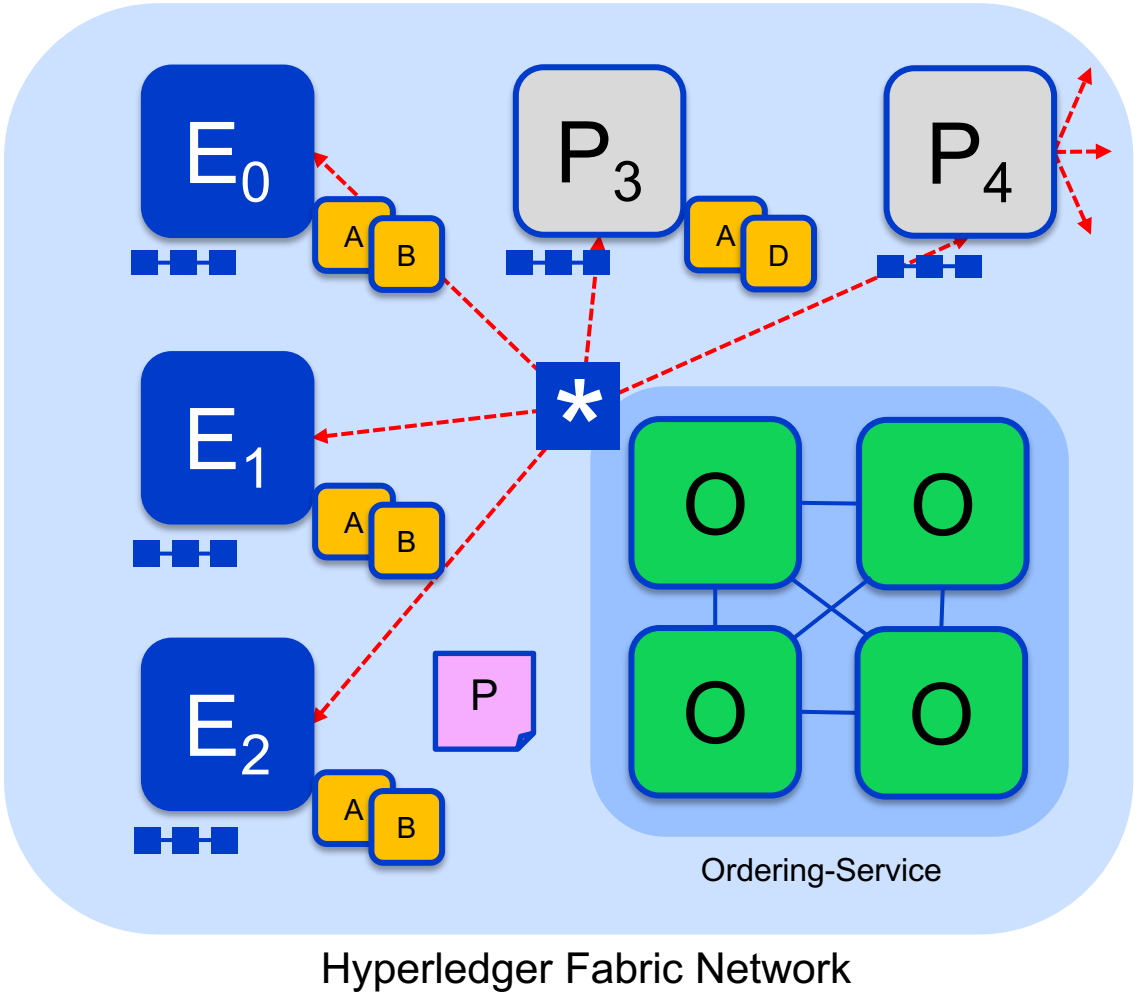
Ordering happens across the fabric in parallel with transactions submitted by other applications

Key:



Sample transaction: Step 5/7 – Deliver Transaction





Hyperledger Fabric Network

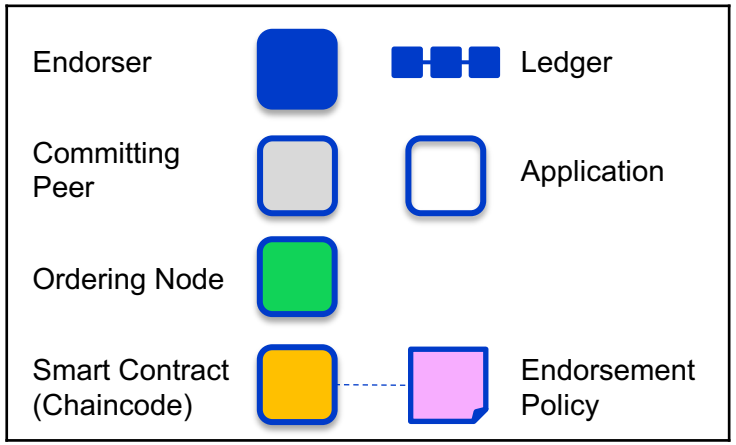
Orderer delivers to committing peers

Ordering service collects transactions into proposed blocks for distribution to committing peers. Peers can deliver to other peers in a hierarchy (not shown)

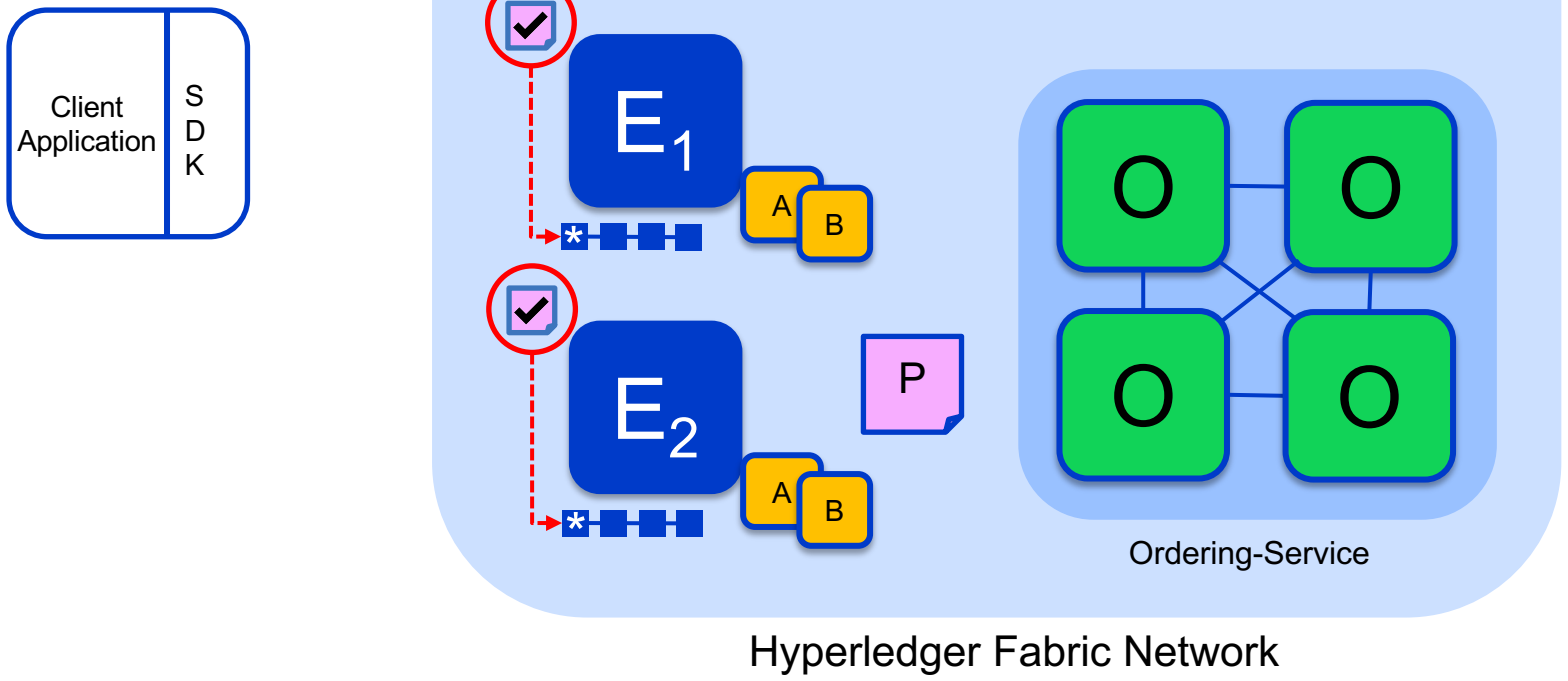
Different ordering algorithms available:

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

Key:



Sample transaction: Step 6/7 – Validate Transaction



Hyperledger Fabric Network

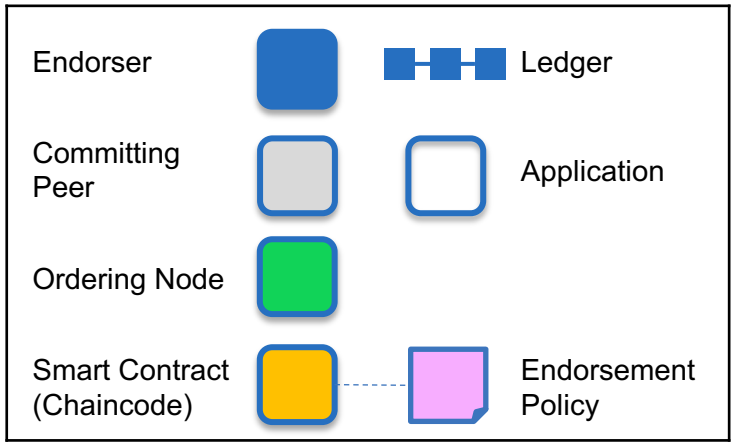
Committing peers validate transactions

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

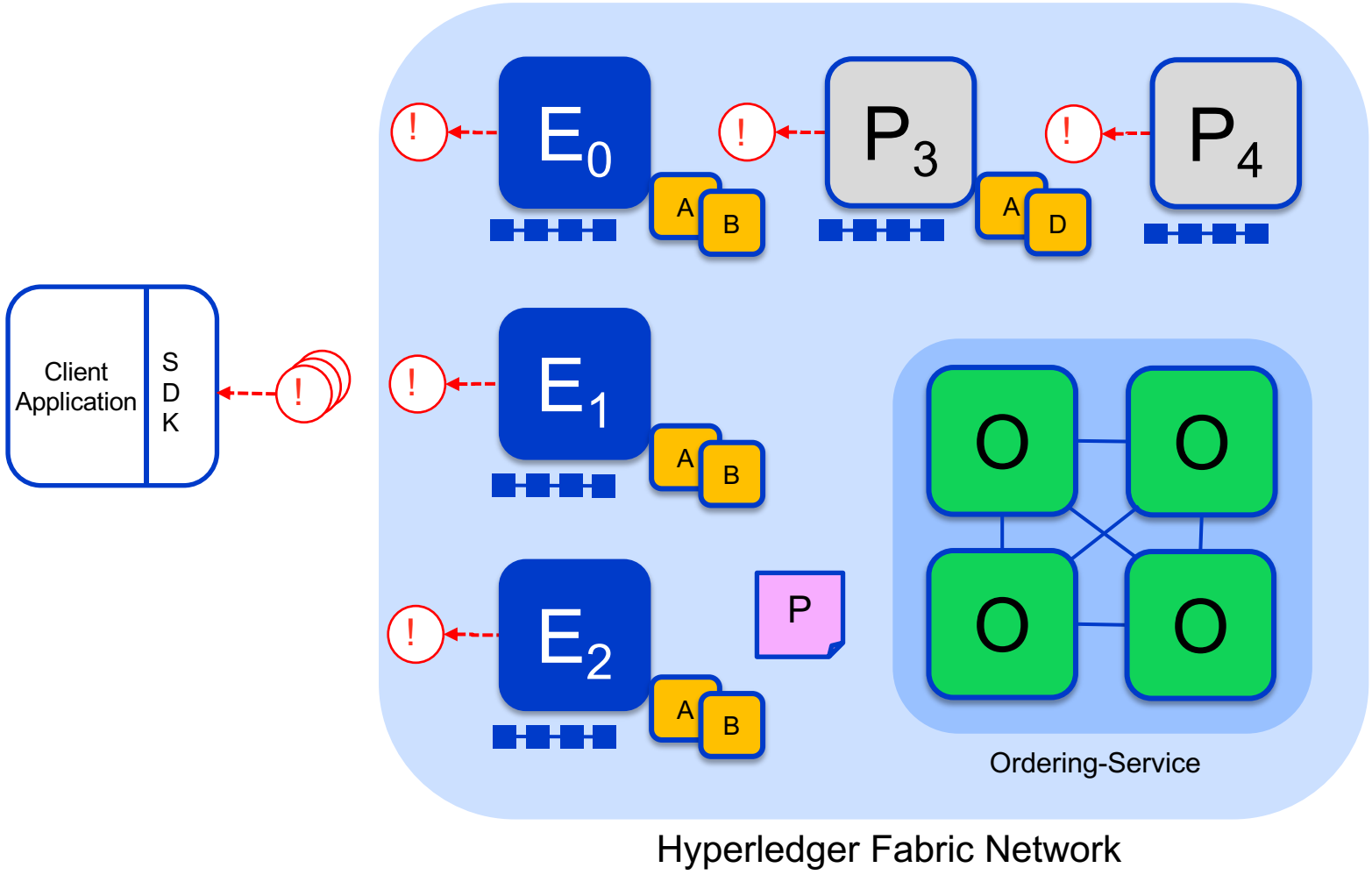
Validated transactions are applied to the world state and retained on the ledger

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

Key:



Sample transaction: Step 7/7 – Notify Transaction



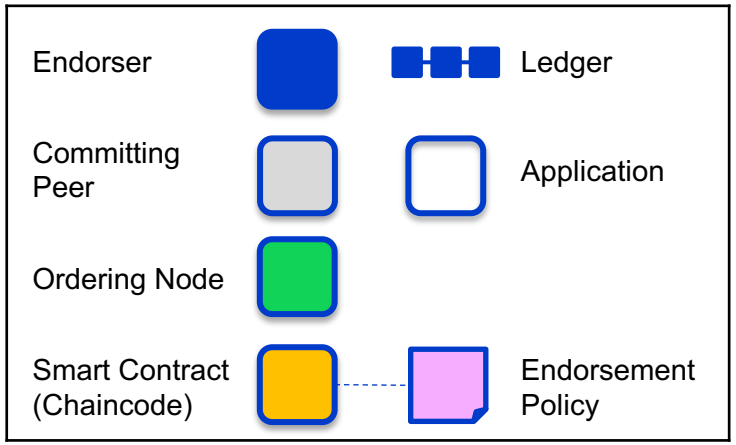
Hyperledger Fabric Network

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:

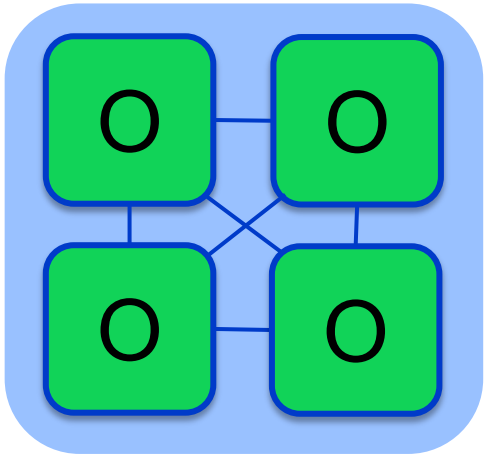


Technical Deep Dive

- Architectural Overview
- Network Consensus
- [Channels and Ordering Service]
- Components
- Network setup
- Endorsement Policies
- Membership Services
- Roadmap

Ordering Service

The ordering service packages transactions into blocks to be delivered to peers. Communication with the service is via channels.

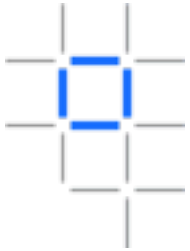


Ordering-Service

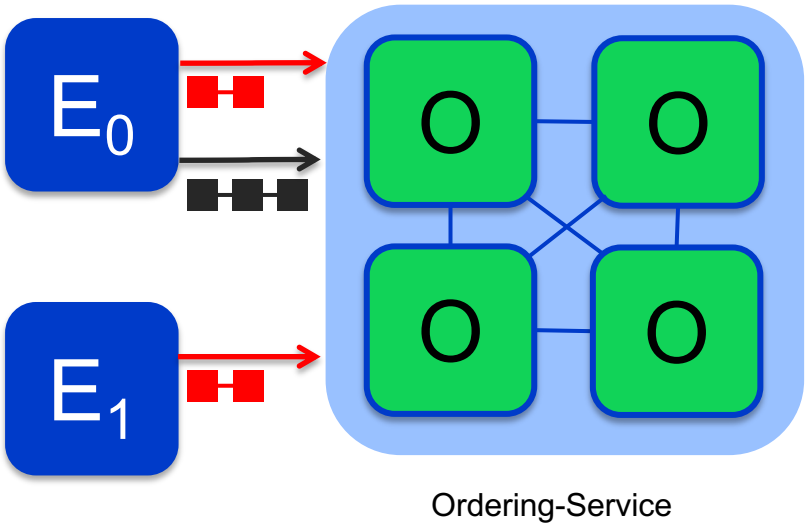
Different configuration options for the ordering service include:

- SOLO
  - Single node for development
- Kafka : Crash fault tolerant consensus
  - 3 nodes minimum
  - Odd number of nodes recommended
- Raft : Crash fault tolerant
  - In-process consensus, no extra dependency
  - Decentralized – multiple orderer orgs

Channels

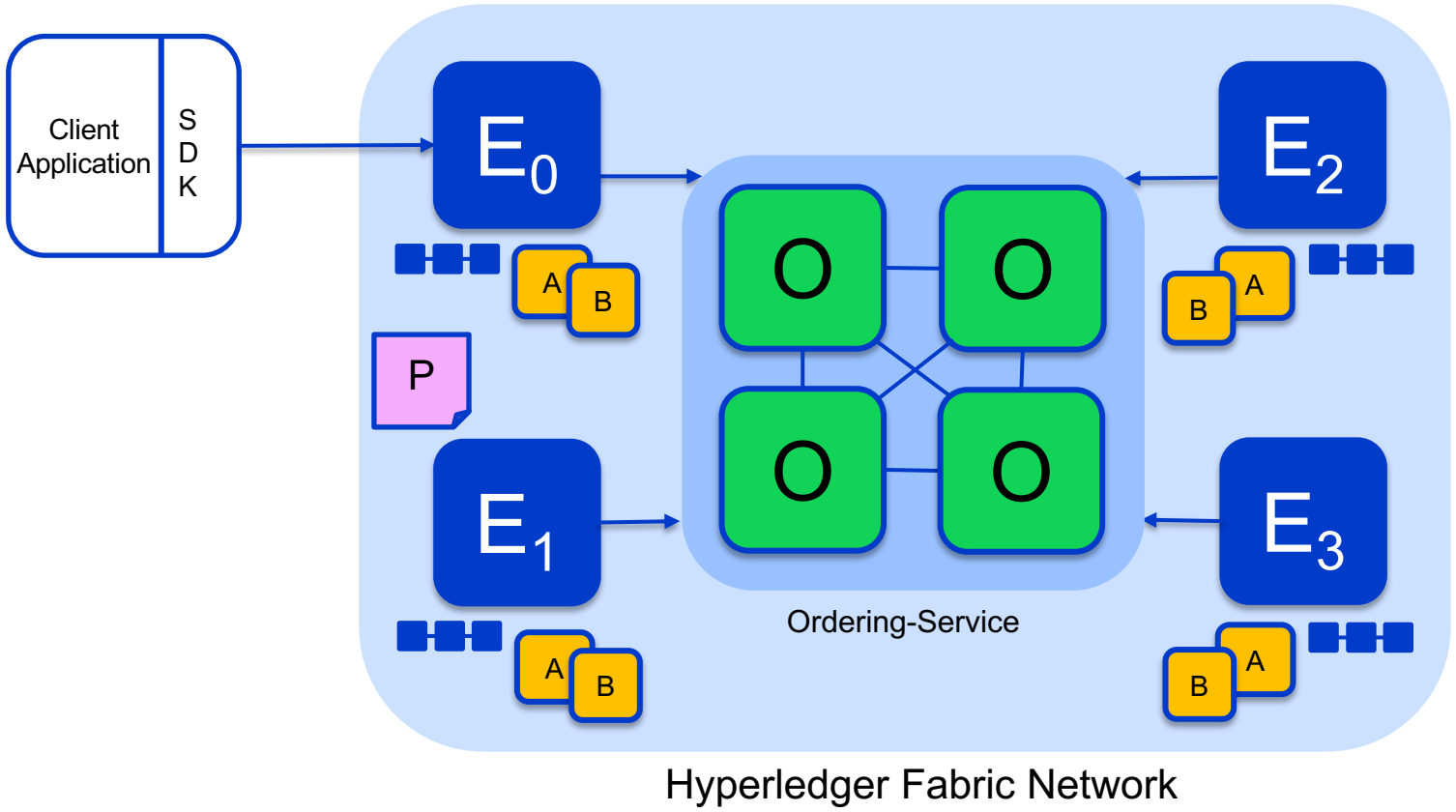


Channels provide privacy between different ledgers



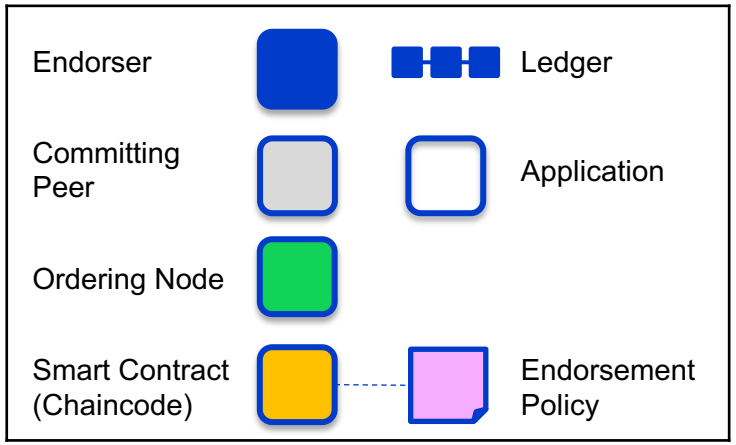
- Ledgers exist in the scope of a channel
  - Channels can be shared across an entire network of peers
  - Channels can be permissioned for a specific set of participants
- Chaincode is installed on peers to access the worldstate
- Chaincode is instantiated on specific channels
- Peers can participate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

Single Channel Network

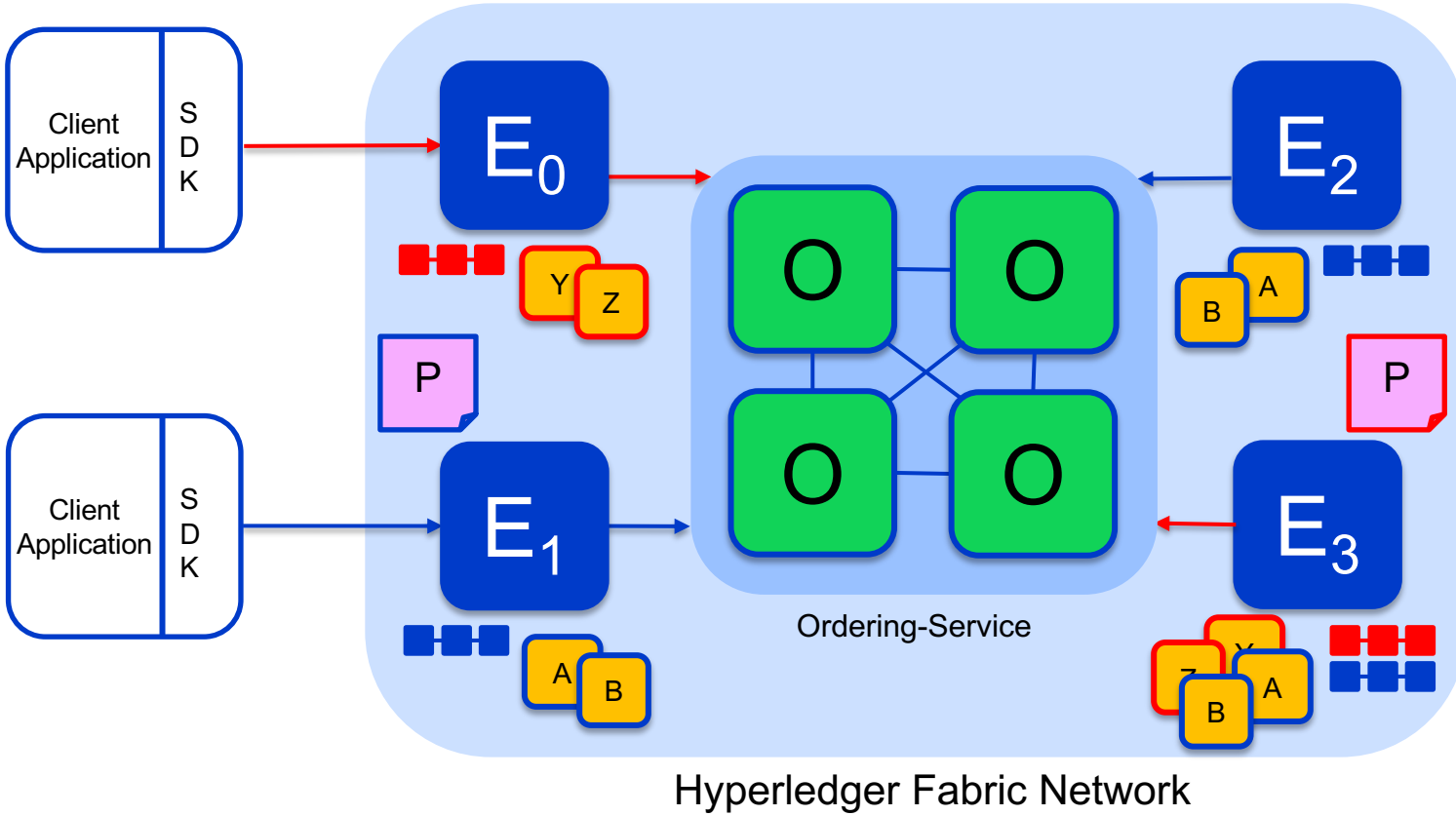


- All peers connect to the same system channel (blue).
- All peers have the same chaincode and maintain the same ledger
- Endorsement by peers E_0, E_1, E_2 and E_3

Key:

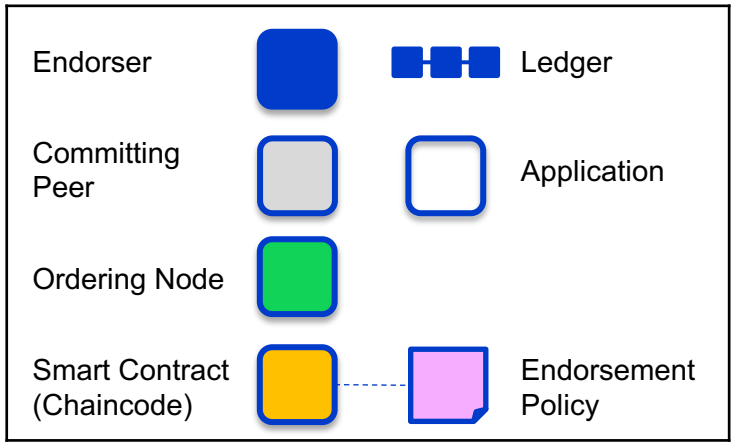


Multi Channel Network



- 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:



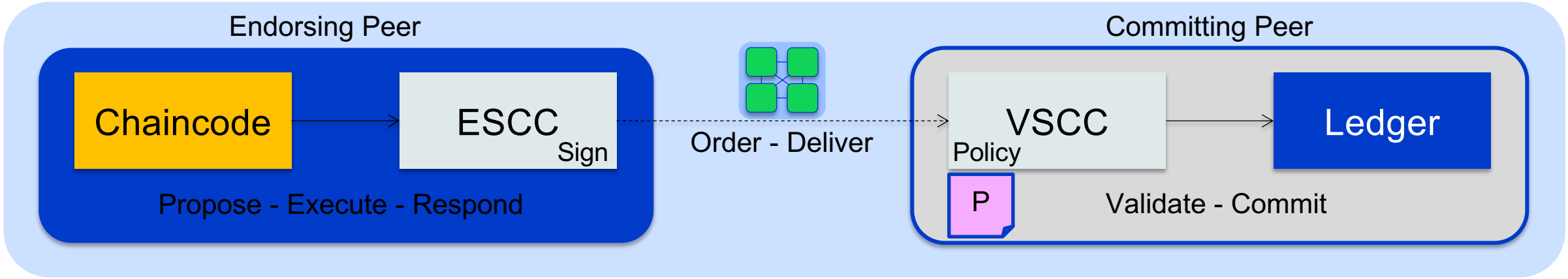
Technical Deep Dive

- Architectural Overview
- Network Consensus
- Channels and Ordering Service
- Components
- Network setup
- [Endorsement Policies]
- Membership Services
- Roadmap

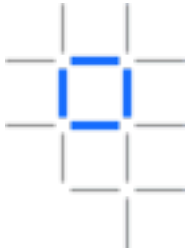
Endorsement Policies

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



Endorsement Policy Syntax



```
$ peer chaincode instantiate
-C mychannel
-n mycc
-v 1.0
-p chaincode_example02
-c '{"Args":["init","a", "100", "b","200"]}'
-P "AND('Org1MSP.member')"
```

Instantiate the chaincode mycc on channel mychannel with the policy AND('Org1MSP.member')

Policy Syntax: EXPR(E[, E...])

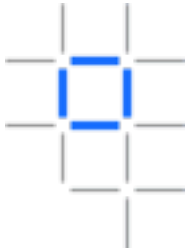
Where EXPR is either AND or OR and E is either a principal or nested EXPR

Principal Syntax: MSP.ROLE

Supported roles are: member and admin

Where MSP is the MSP ID, and ROLE is either “member” or “admin”

Endorsement Policy Examples



Examples of policies:

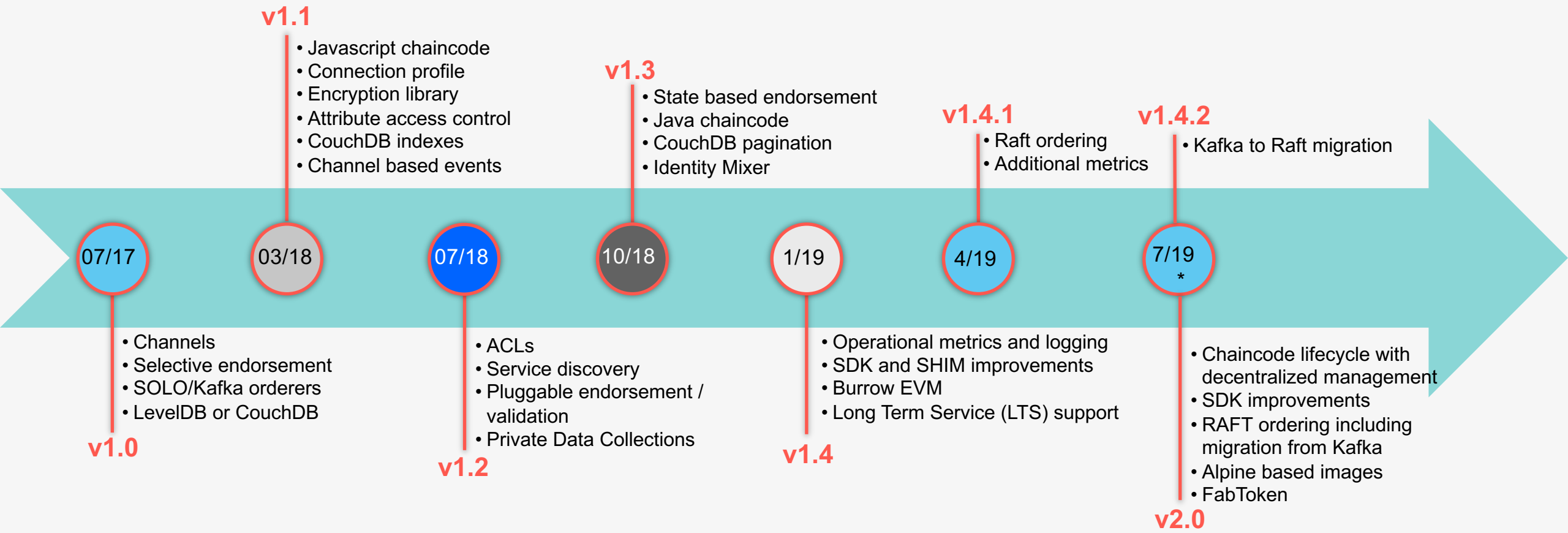
- Request 1 signature from all three principals
  - AND('Org1.member', 'Org2.member', 'Org3.member')
- Request 1 signature from either one of the two principals
  - OR('Org1.member', 'Org2.member')
- Request either one signature from a member of the Org1 MSP or (1 signature from a member of the Org2 MSP and 1 signature from a member of the Org3 MSP)
  - OR('Org1.member', AND('Org2.member', 'Org3.member'))

Technical Deep Dive

- Architectural Overview
- Network Consensus
- Channels and Ordering Service
- Components
- Network setup
- Endorsement Policies
- Membership Services
- [Roadmap]

HYPERLEDGER FABRIC

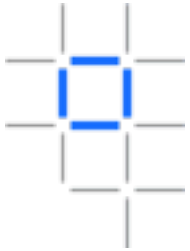
Roadmap



HYPERLEDGER
BLOCKCHAIN TECHNOLOGIES FOR BUSINESS

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

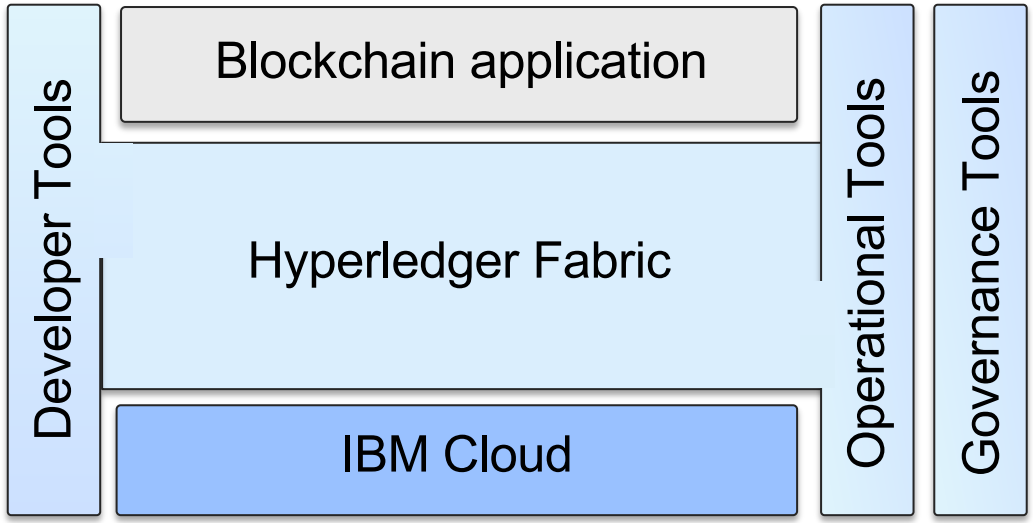


- Build Your First Network (BYFN) – Network administrator
  - A simple network with 2 organizations running 2 peers, with one channel, a simple chaincode
  - Dockerhub images
  - Uses predefined enrollment certificates and « Solo » Ordering Service
- Extend Your First Network – Network administrator
  - 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)

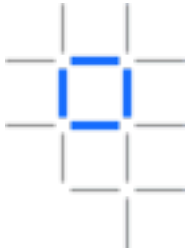
IBM Blockchain Platform

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



Resources



- **Hyperledger** http://hyperledger-fabric.readthedocs.io/en/release/
  - **Docs + tutorials**
- **IBM Code:** https://developer.ibm.com/code/technologies/blockchain/
  - **Code patterns, lectures, howtos, lab, etc**
- **IBM Blockchain Dev Center:** https://developer.ibm.com/blockchain/
  - **Blockchain 101**
- **IBM Blockchain Platform:**
  - **Starter Plan:** https://www.ibm.com/blockchain/getting-started.html

Get Involved!

Ensure the strength and longevity of a core technology to your business.

Publicly proclaim your leadership in the blockchain space.

Work with other blockchain leaders to develop and promote Hyperledger blockchain for business technologies.

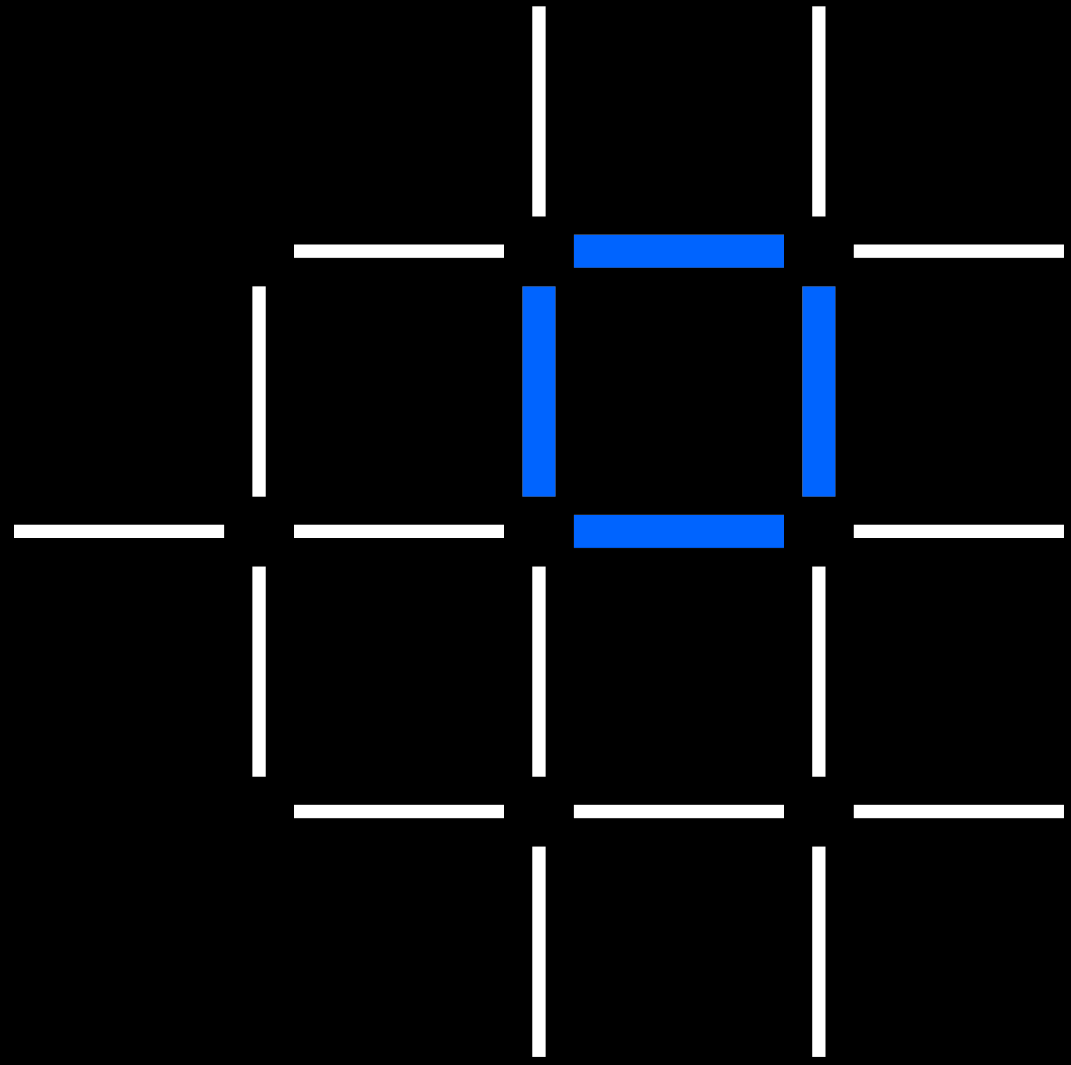
Visit hyperledger.org/about/join
or email info@hyperledger.org.

Thank you

Swetha Repakula
srepaku@us.ibm.com
@swayr93

Morgan Bauer
mbauer@us.ibm.com
@ibmhb

IBM **Blockchain**



www.ibm.com/blockchain

developer.ibm.com/blockchain

www.hyperledger.org

IBM