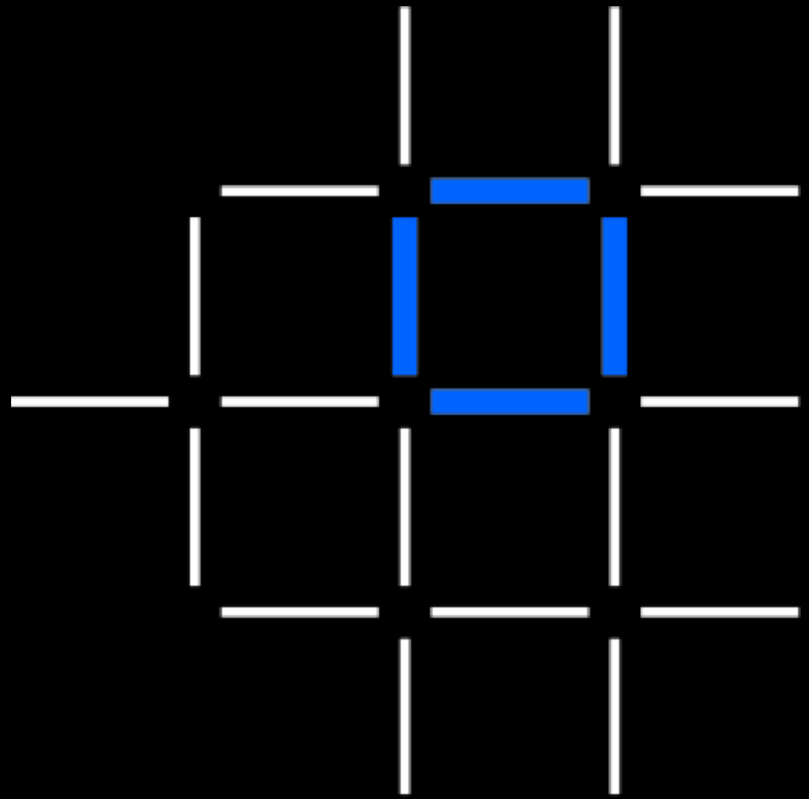
Blockchain Basics, Hyperledger Fabric, IBM Blockchain

Raghavendra Deshpande
Twitter - @ragdeshp
Email – ragdeshp@in.ibm.com



V4.07, 1 March 2019

IBM **Blockchain**

IBM

Introduction

Fabric Architecture Concepts

IBM Blockchain Platform

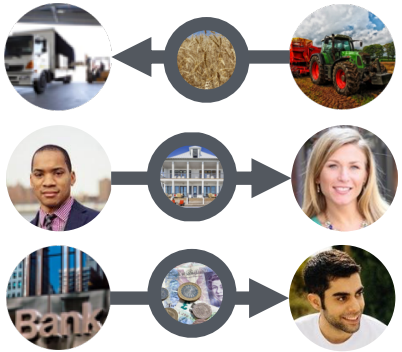
Business Networks

Business Network
Ecosystem the business exists in.
Suppliers, Banks, Regulators…

Assets
Anything that can be
manipulated to produce value

Ledgers

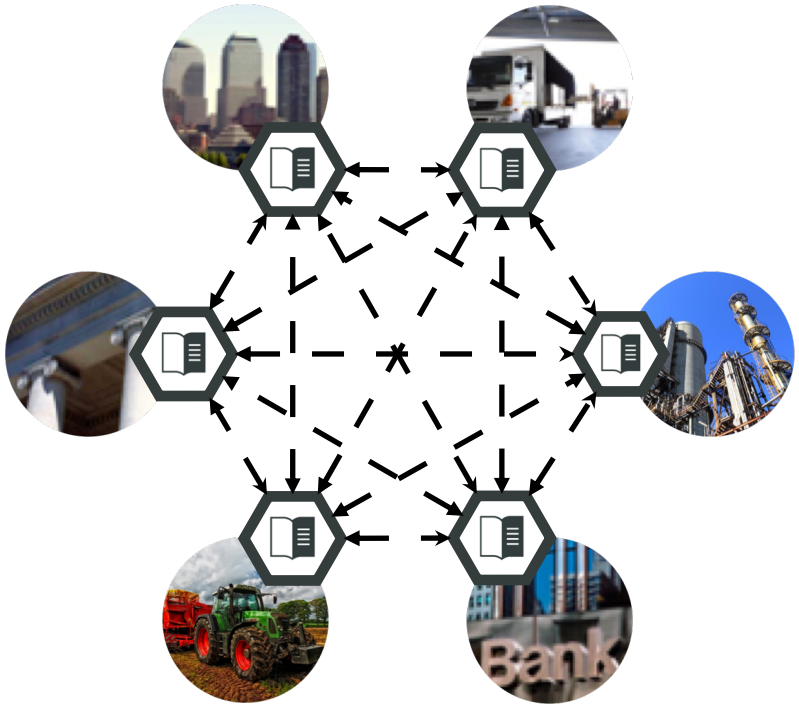
Ledger
A record of all your incoming and outgoing assets.



Transactions
Transfer of ownership of an asset.

Contracts
Conditions under which assets can be transferred.

Separate Ledgers (Status Quo)



Inefficient

- Separate ledgers record information multiple times.

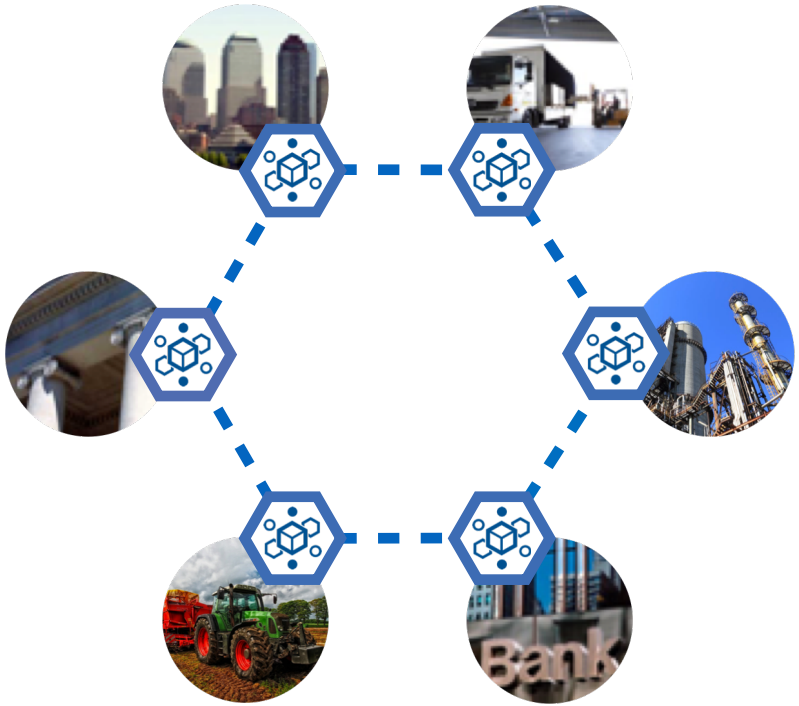
Expensive

- Time & money consumed on maintaining the same data.
- Point-to-point exchange of data is slow.
- Exchange costs money if middle-men are involved.

Vulnerable

- One mistake on one ledger will cause an issue.
- Disputes are hard to reconcile because data is siloed.
- Data is also often centralised.

Shared Ledgers (Using Blockchain)



Consensus

- Transactions must be approved collectively before being written.

Immutability

- Once a transaction is written to the chain it cannot be removed or edited.

Provenance

- The chain contains the history of every asset.

Finality

- The chain is the single source of truth, there is no disparity between replicas.

Hyperledger: A Linux Foundation project

- IBM Blockchain Platform is underpinned by technology from the Hyperledger project
- Hyperledger is a collaborative effort created to advance cross-industry blockchain technologies for business
- Founded February 2016; now more than **260 member organizations**
- Open source
  Open standards
  Open governance model

Source: https://www.hyperledger.org/members
Updated: 8 January 2019

Premier

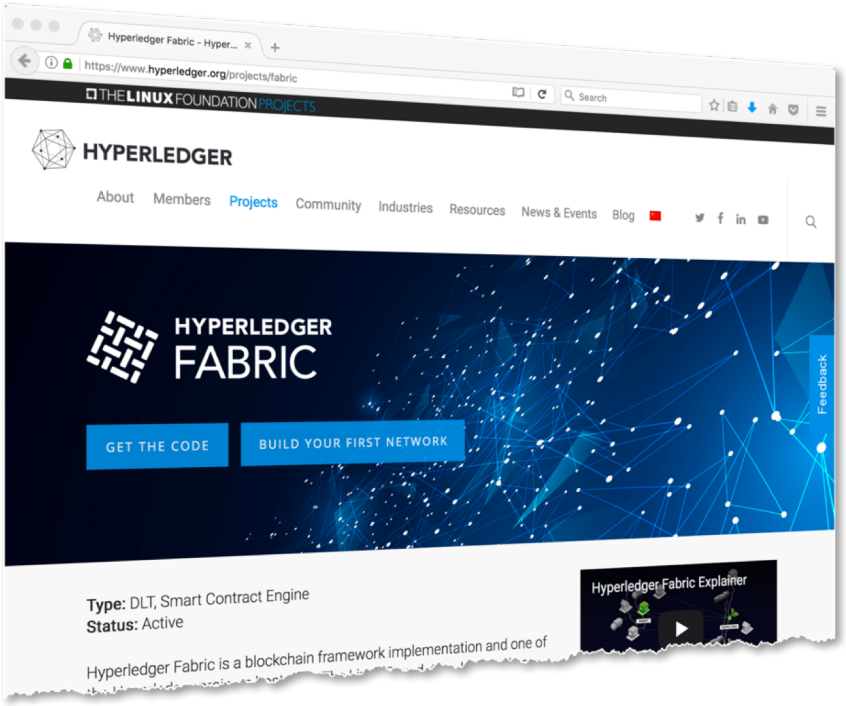
Associate

General

Academia Associate

General

Distributed ledger

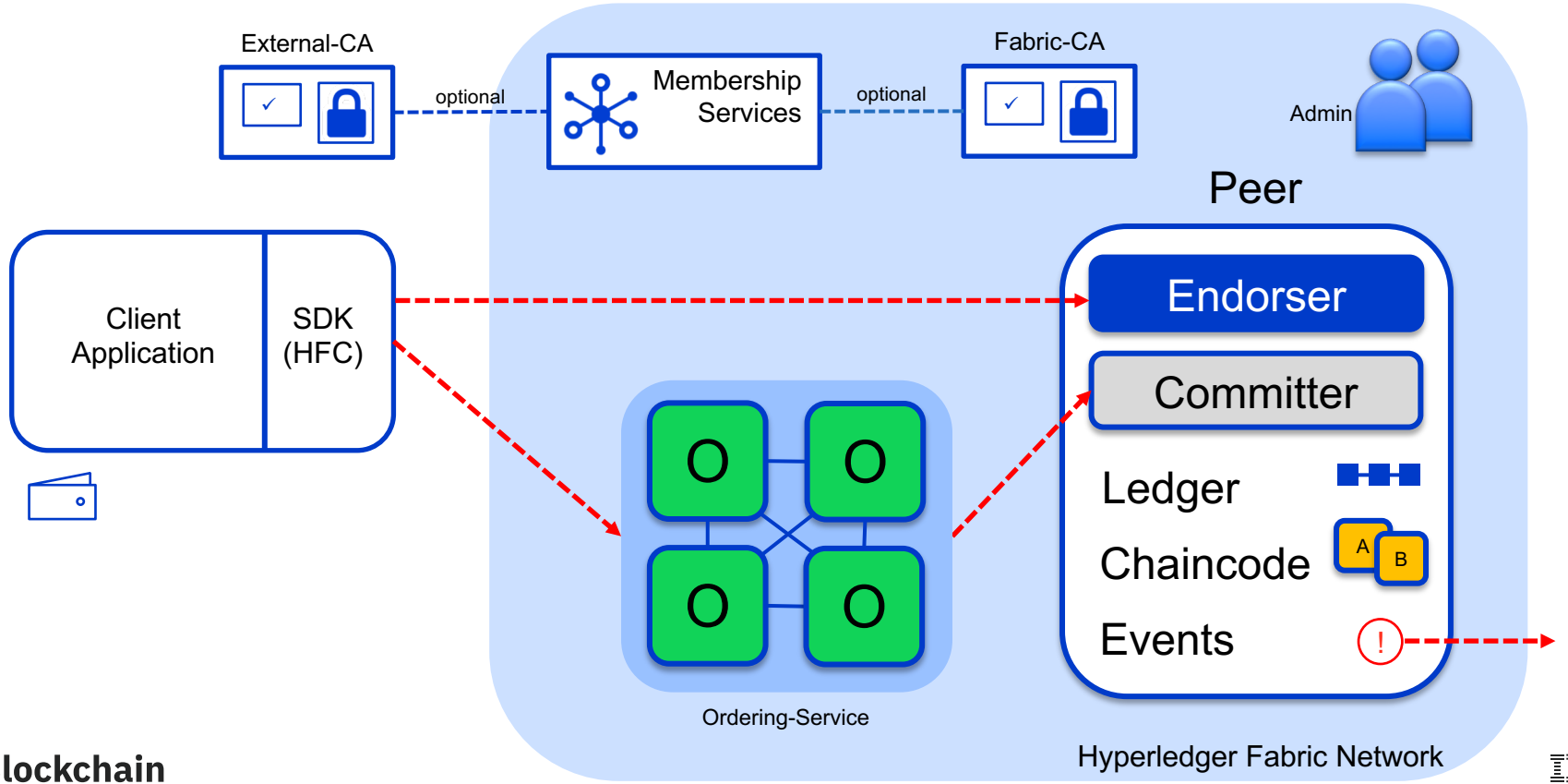


- An implementation of blockchain technology that is a foundation for developing blockchain applications
- Emphasis on ledger, smart contracts, consensus, confidentiality, resiliency and scalability.
- V1.4 released January 2019
  - Long Term Service release with emphasis on production operational and serviceability enhancements
  - New programming model abstractions for ease of development
- IBM is one of the many contributing organizations

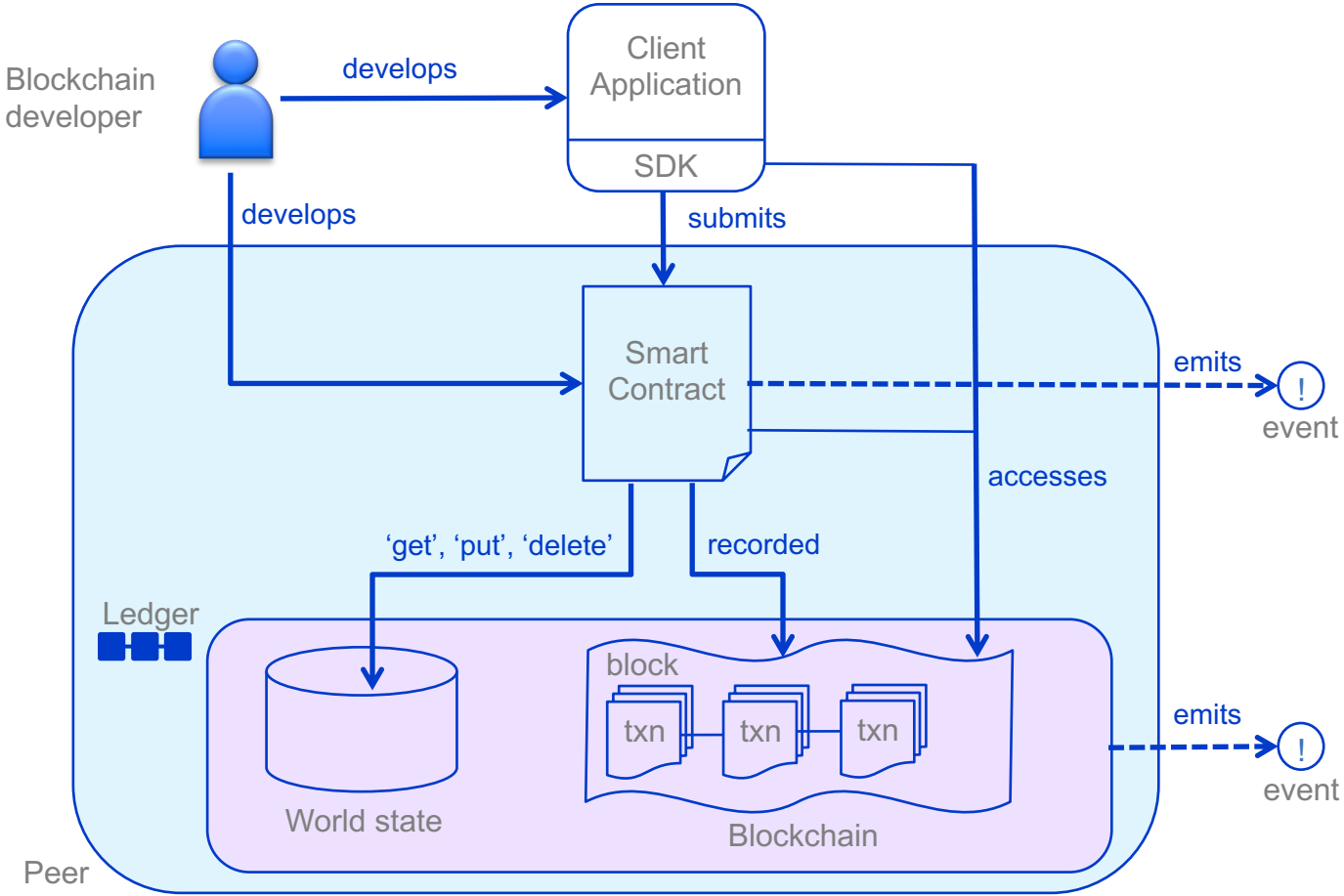
Technical Concepts

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

Hyperledger Fabric V1.x Architecture



How applications interact with the ledger

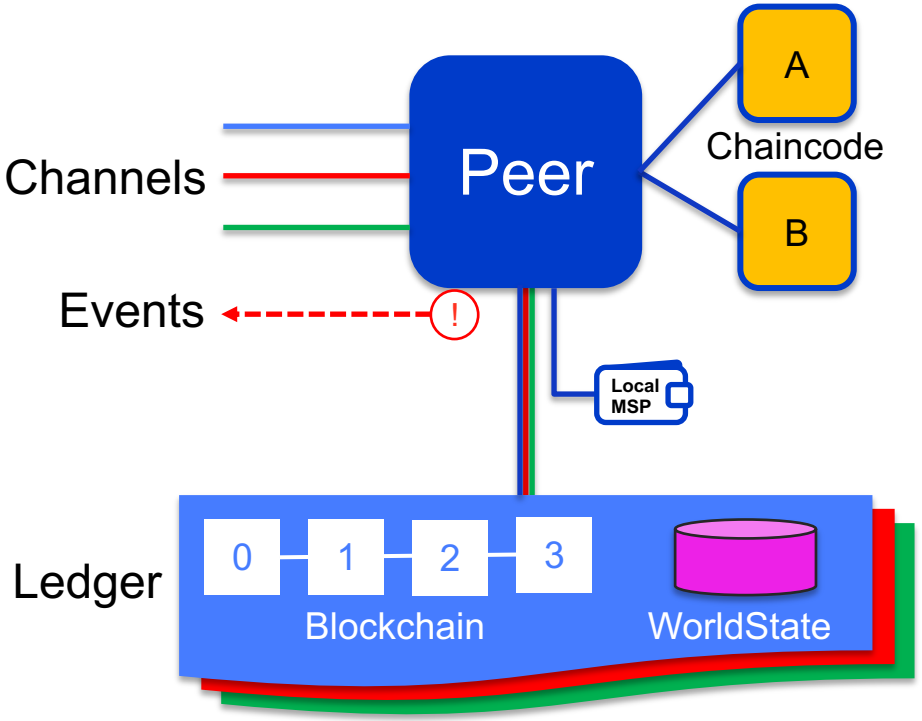


Technical Concepts

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

Fabric Peer

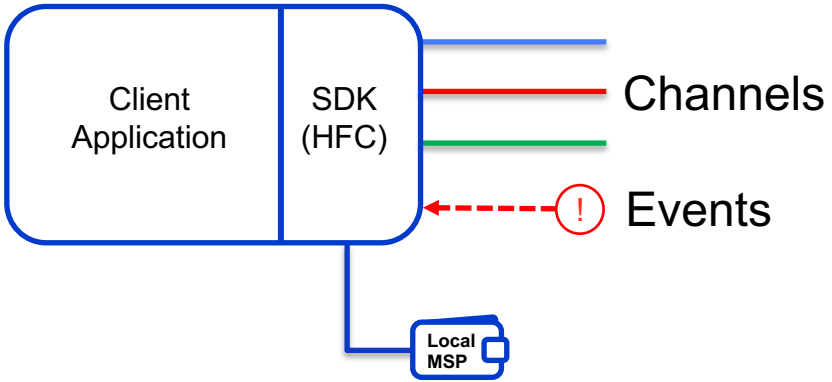
- Each peer:
  - Connects to one or more channels
  - Maintains one or more ledgers per channel
  - Maintains installed chaincode
  - Manages runtime docker containers for instantiated chaincode
    - Chaincode is instantiated on a channel
    - Runtime docker container shared by channels with same chaincode instantiated (no state stored in container)
  - Has a local MSP (Membership Services Provider) that provides crypto material
  - Emits events to the client application



Client Application

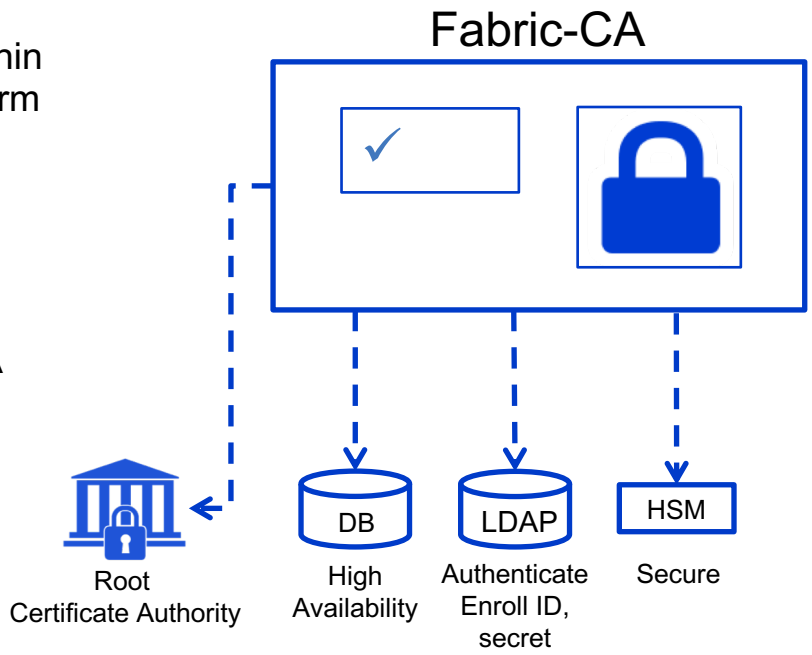
- Each client application uses Fabric SDK to:
  - Connects over channels to one or more peers
  - Connects over channels to one or more orderer nodes
  - Receives events from peers
  - Local MSP provides client crypto material

  - Client can be written in different languages (Node.js, Go, Java, Python?)



Fabric-CA

- Default (optional) Certificate Authority within Fabric network for issuing Ecerts (long-term identity)
- Supports clustering for HA characteristics
- Supports LDAP for user authentication
- Supports HSM for security
- Can be configured as an intermediate CA



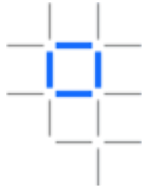
Technical Concepts

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

Nodes and roles

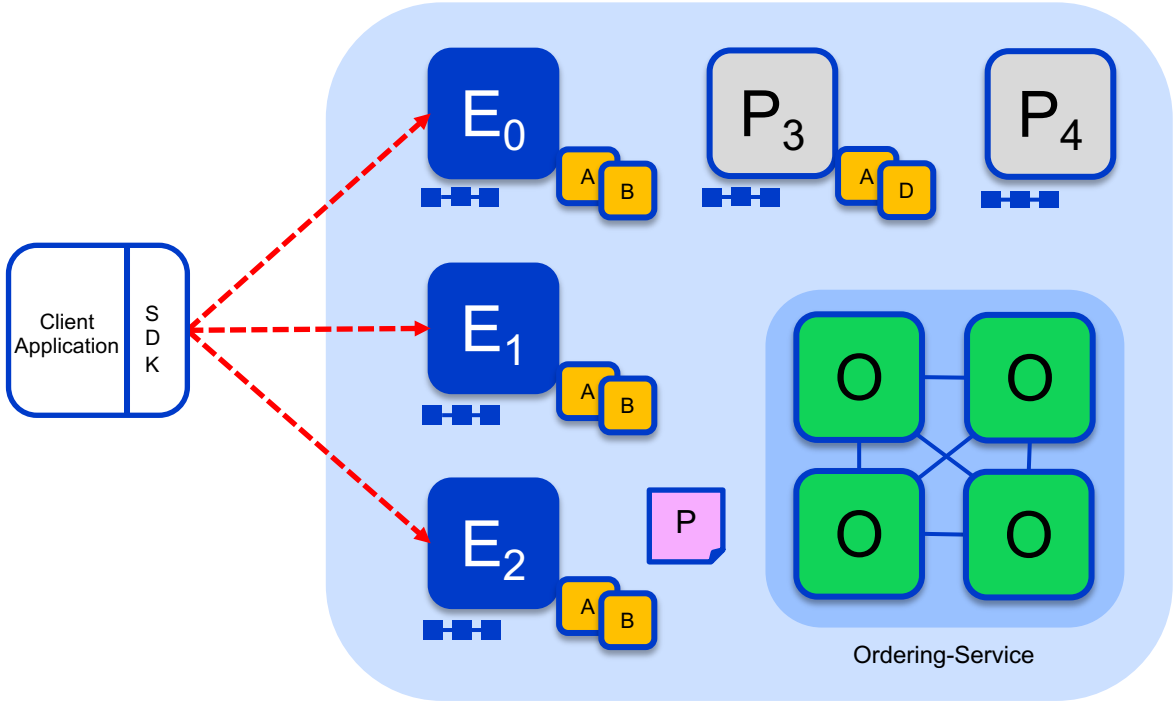
	**Peer:** Maintains ledger and state. Commits transactions. May hold smart contract (chaincode).
	**Endorsing Peer:** Specialized peer also endorses transactions by receiving a transaction proposal and responds by granting or denying endorsement. Must hold smart contract.
	**Ordering Node:** Approves 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

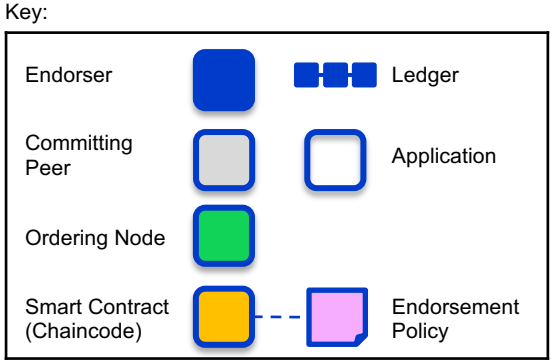


Hyperledger Fabric Network

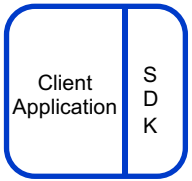
Application proposes transaction

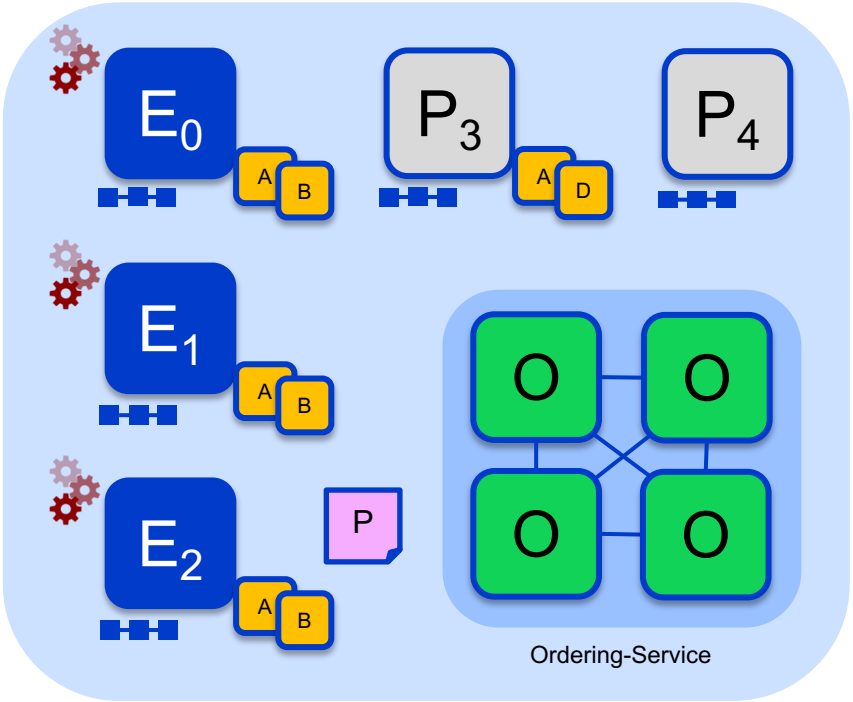
Endorsement policy:
- "E_0, E_1 and E_2 must sign"
- (P_3, P_4 are not part of the policy)

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



Sample transaction: Step 2/7 – Execute proposal





Hyperledger Fabric Network

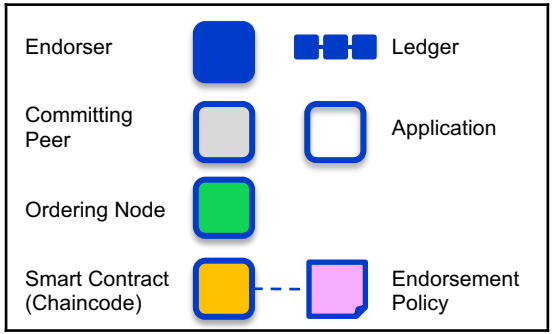
Endorsers Execute Proposals

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

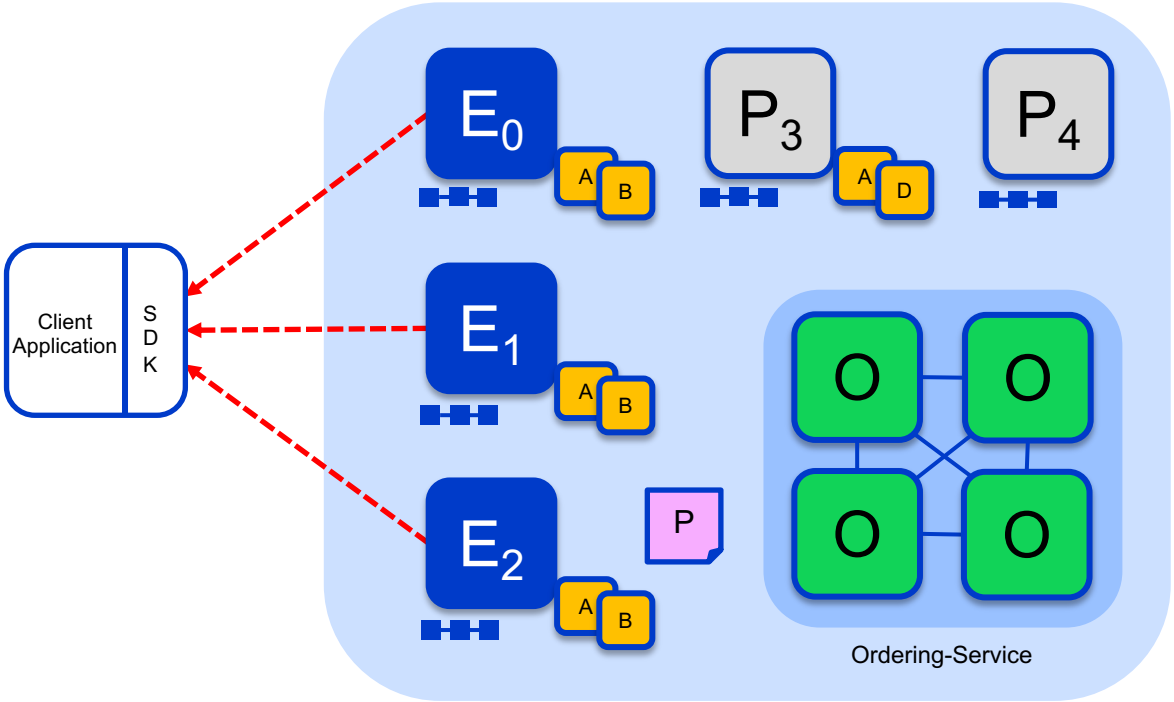
Each execution will capture the set of Read and Written data, called RW sets, which will now flow in the fabric.

Transactions can be signed & encrypted

Key:



Sample transaction: Step 3/7 – Proposal Response



Hyperledger Fabric Network

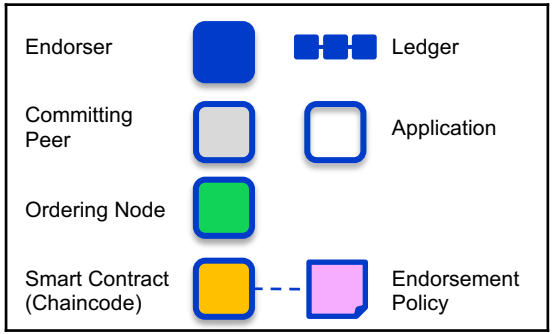
Application receives responses

RW sets are asynchronously returned to application

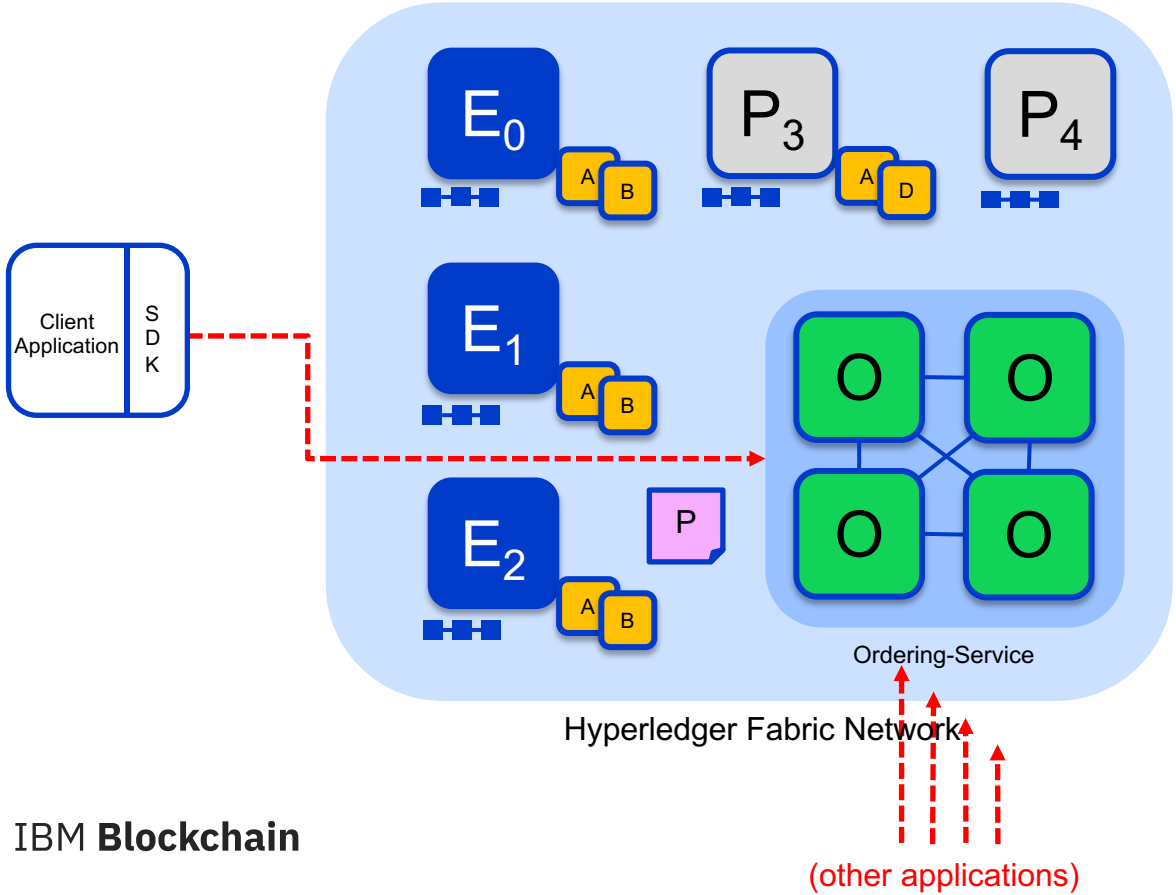
The RW sets are signed by each endorser, and also includes each record version number

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

Key:



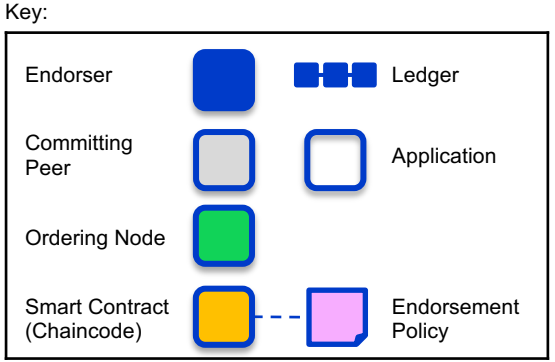
Sample transaction: Step 4/7 – Order Transaction



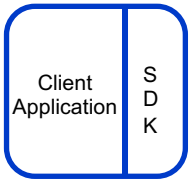
Responses submitted for ordering

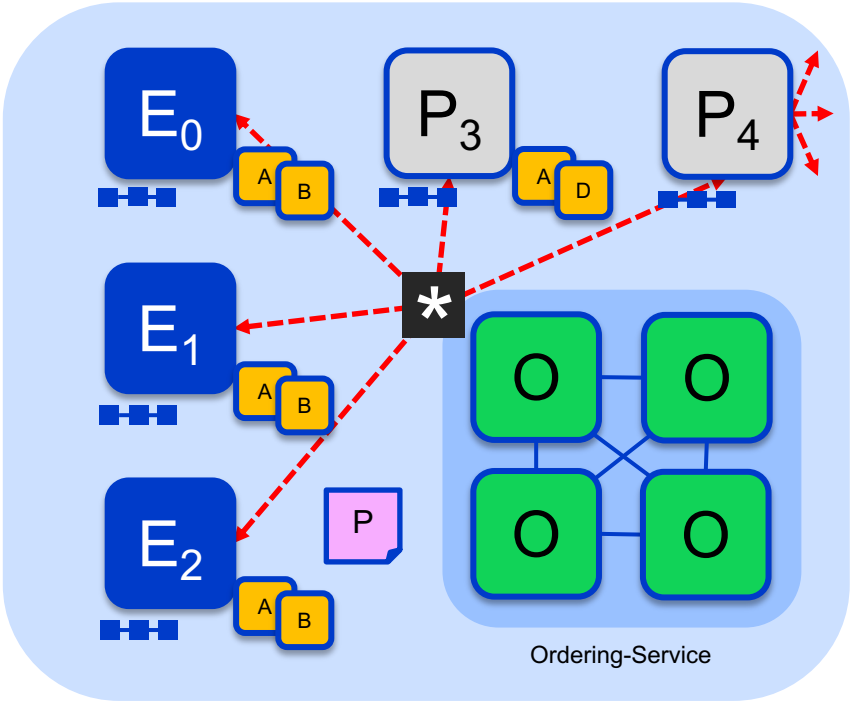
Application submits responses as a transaction to be ordered.

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



Sample transaction: Step 5/7 – Deliver Transaction





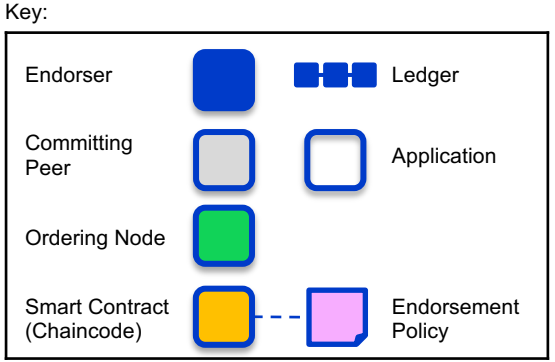
Hyperledger Fabric Network

Orderer delivers to committing peers

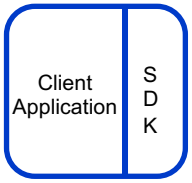
Ordering service collects transactions into proposed blocks for distribution to committing peers. Peers can deliver to other peers in a hierarchy (not shown)

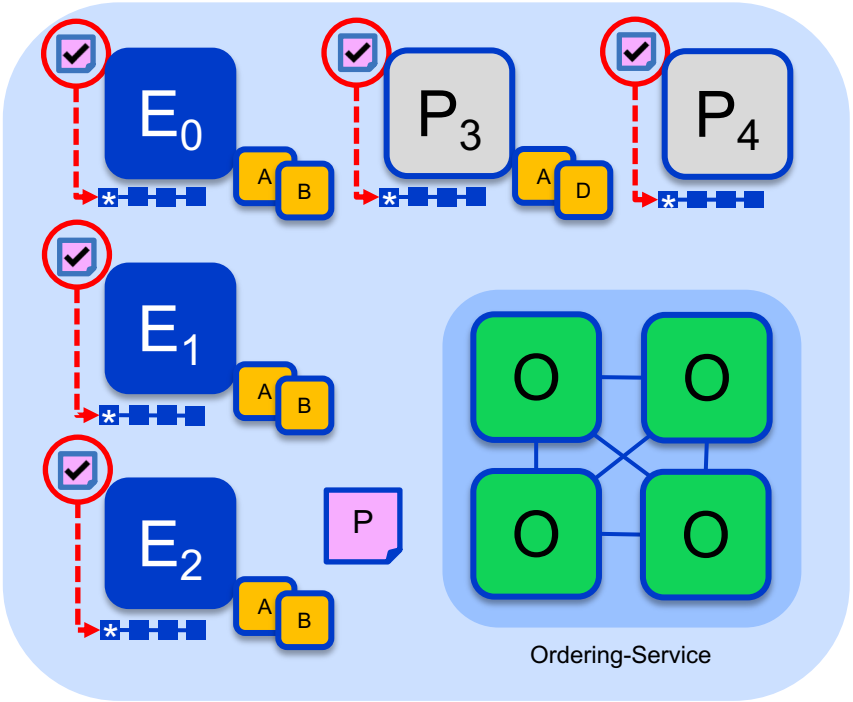
Different ordering algorithms available:

- SOLO (Single node, development)
- Kafka (Crash fault tolerance)



Sample transaction: Step 6/7 – Validate Transaction





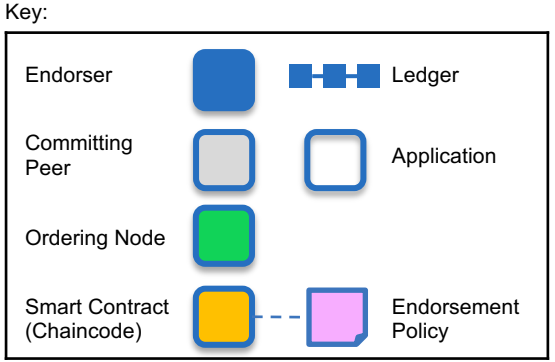
Hyperledger Fabric Network

Committing peers validate transactions

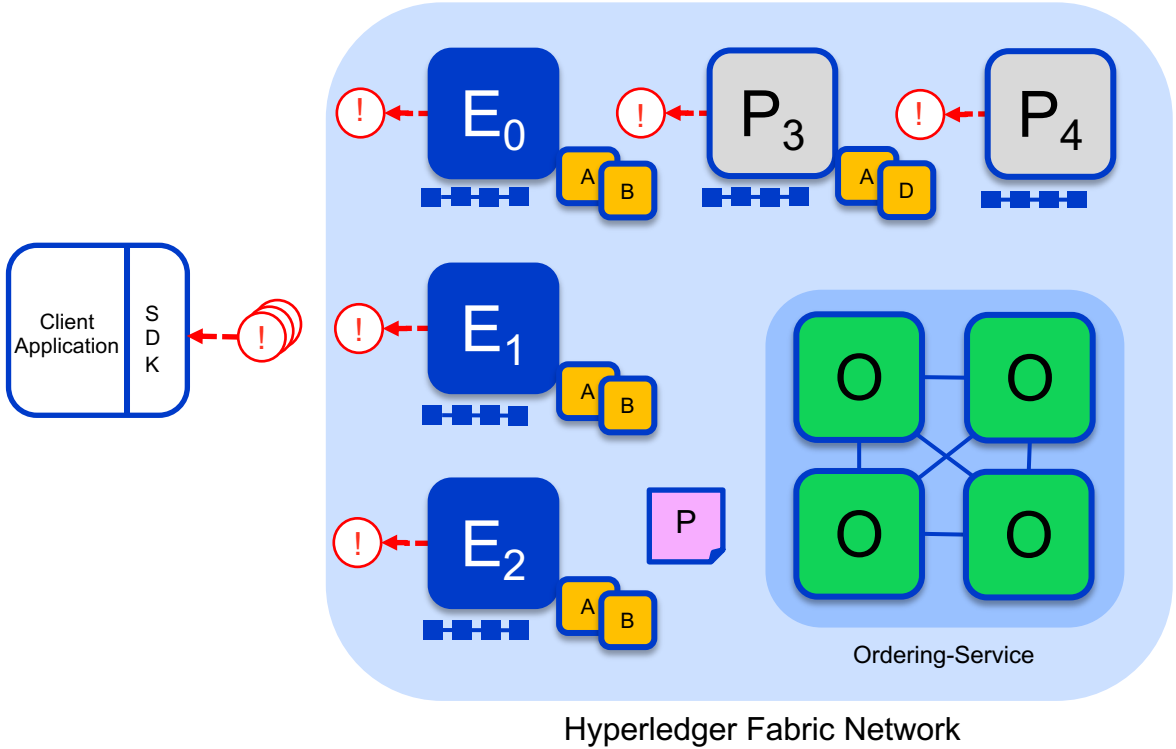
Every committing peer validates against the endorsement policy. Also check RW sets are still valid for current world state

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



Sample transaction: Step 7/7 – Notify Transaction

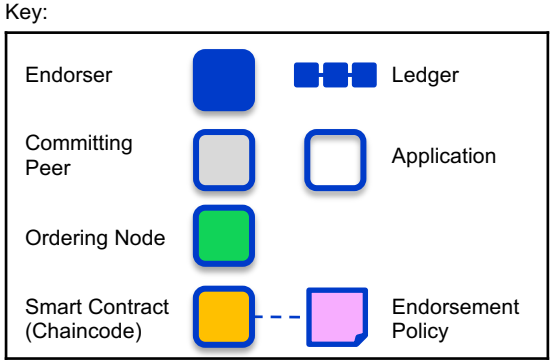


Hyperledger Fabric Network

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

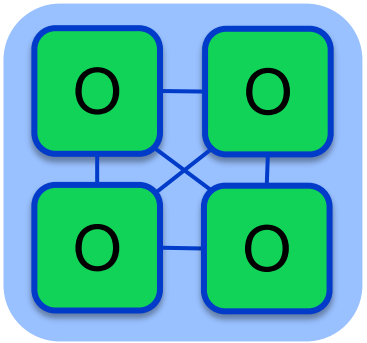


Technical Concepts

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

Ordering Service

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



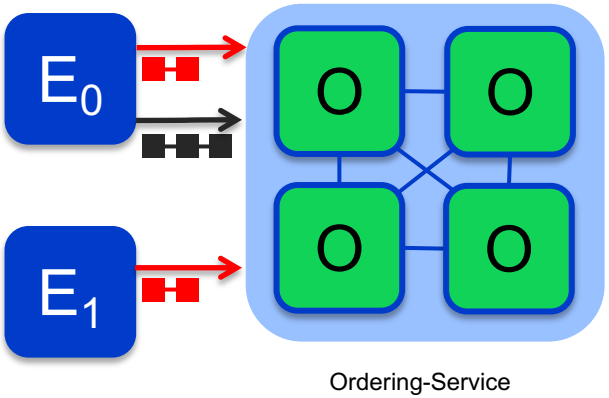
Ordering-Service

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

Channels

Channels provide privacy between different ledgers

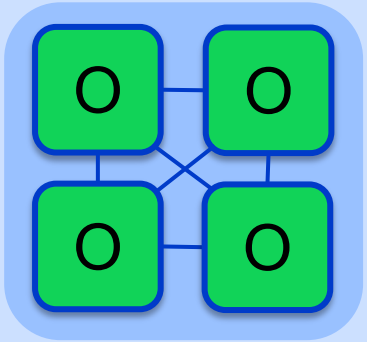


- Ledgers exist in the scope of a channel
  - Channels can be shared across an entire network of peers
  - Channels can be permissioned for a specific set of participants
- Chaincode is installed on peers to access the worldstate
- Chaincode is instantiated on specific channels
- Peers can participate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

Technical Concepts

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

Bootstrap Network (1/6) - Configure & Start Ordering Service



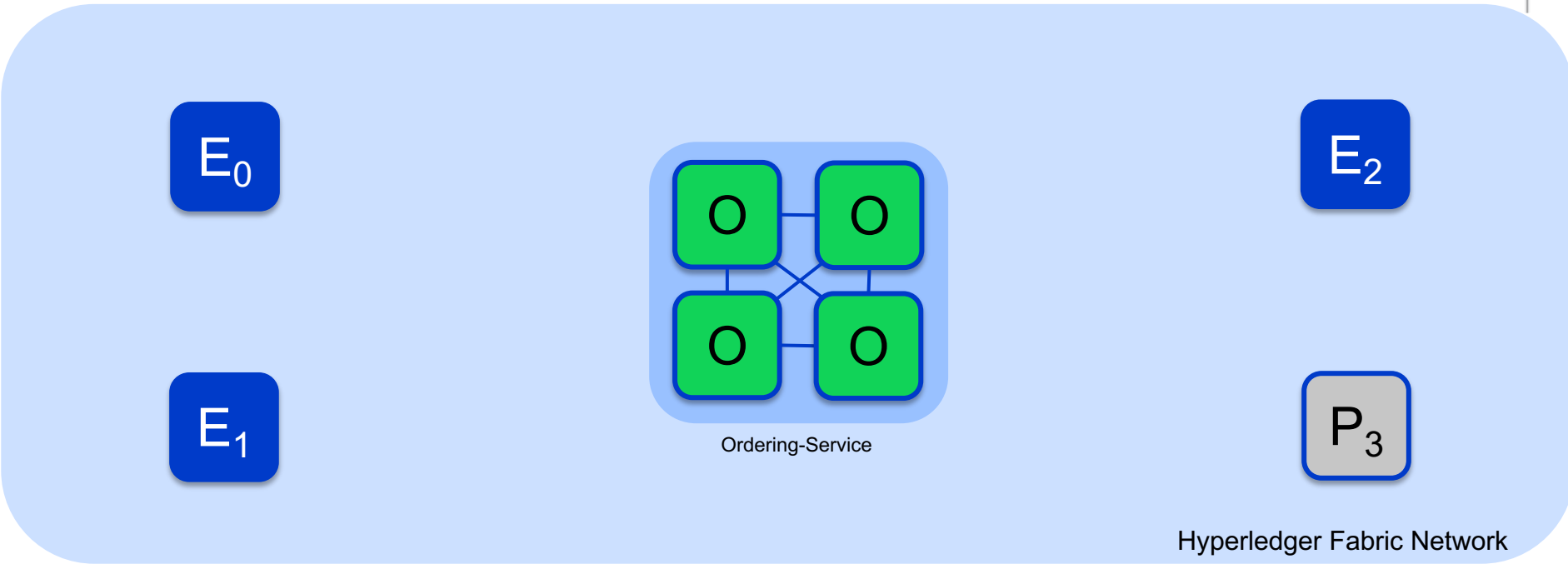
Ordering-Service

Hyperledger Fabric Network

An Ordering Service is configured and started for the network:

$ docker-compose [-f orderer.yml] ...

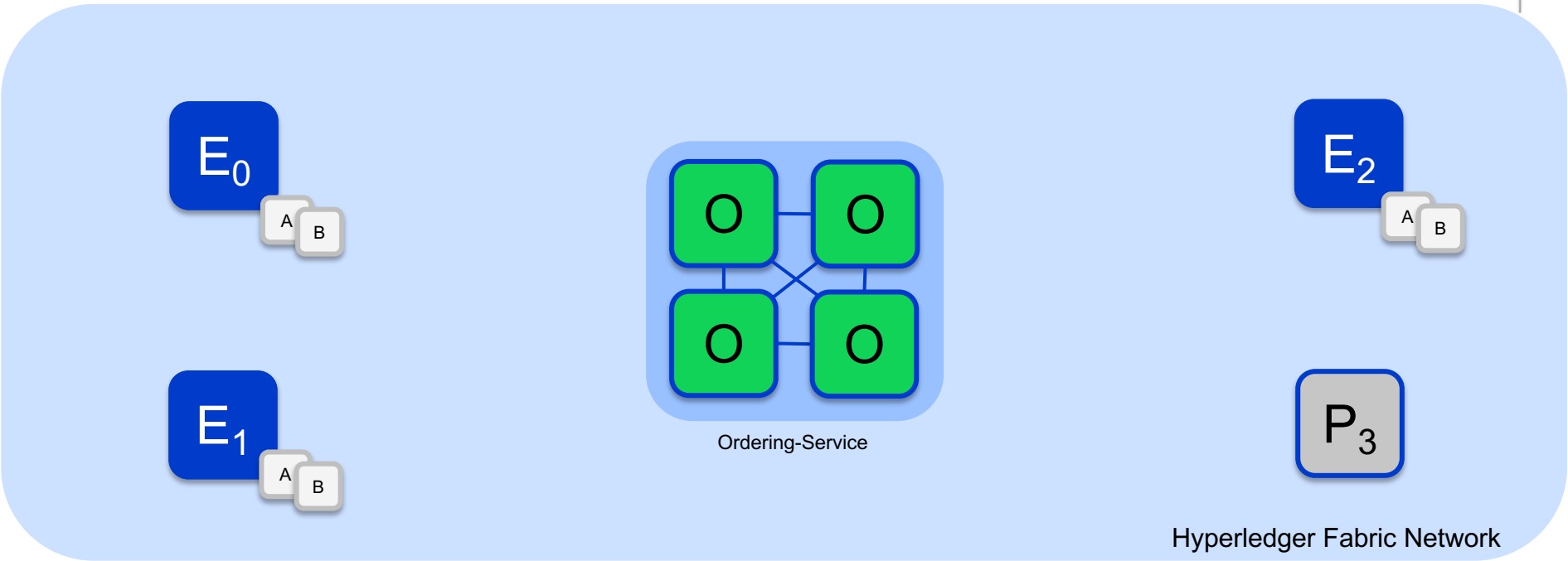
Bootstrap Network (2/6) - Configure and Start Peer Nodes



A peer is configured and started for each Endorser or Committer in the network:

$ peer node start ...

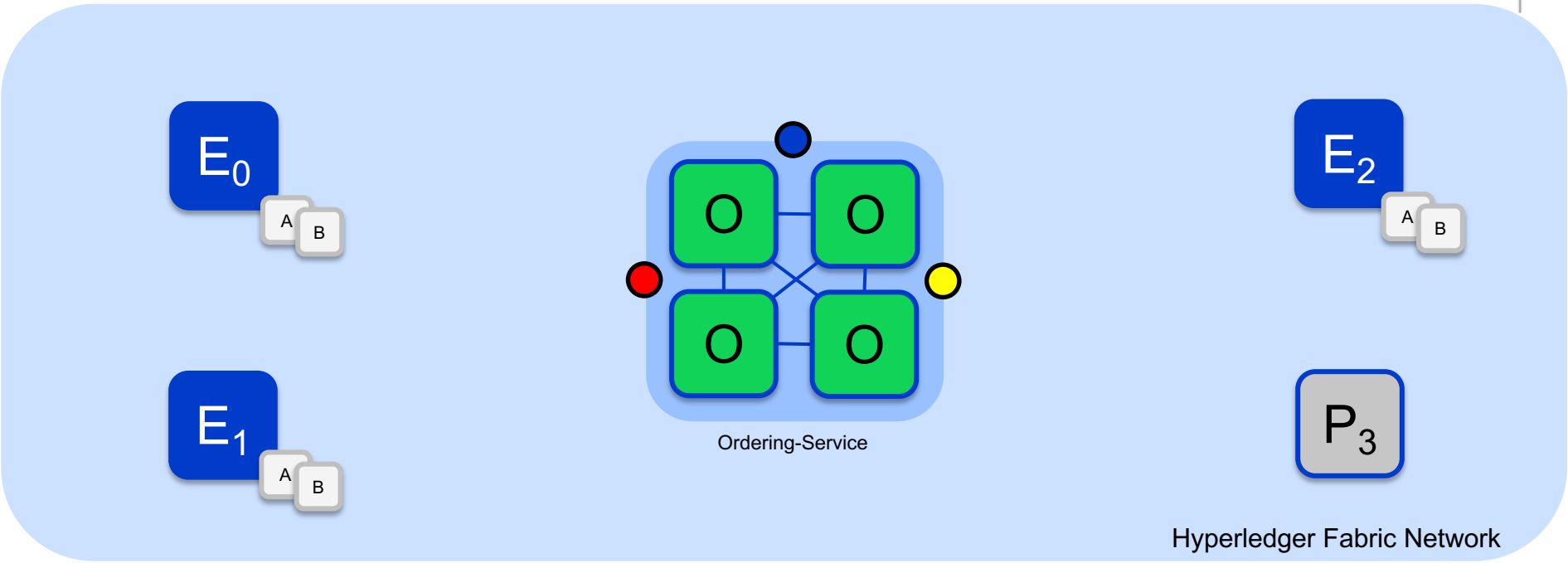
Bootstrap Network (3/6) - Install Chaincode



Chaincode is installed onto each Endorsing Peer that needs to execute it:

$ peer chaincode install ...

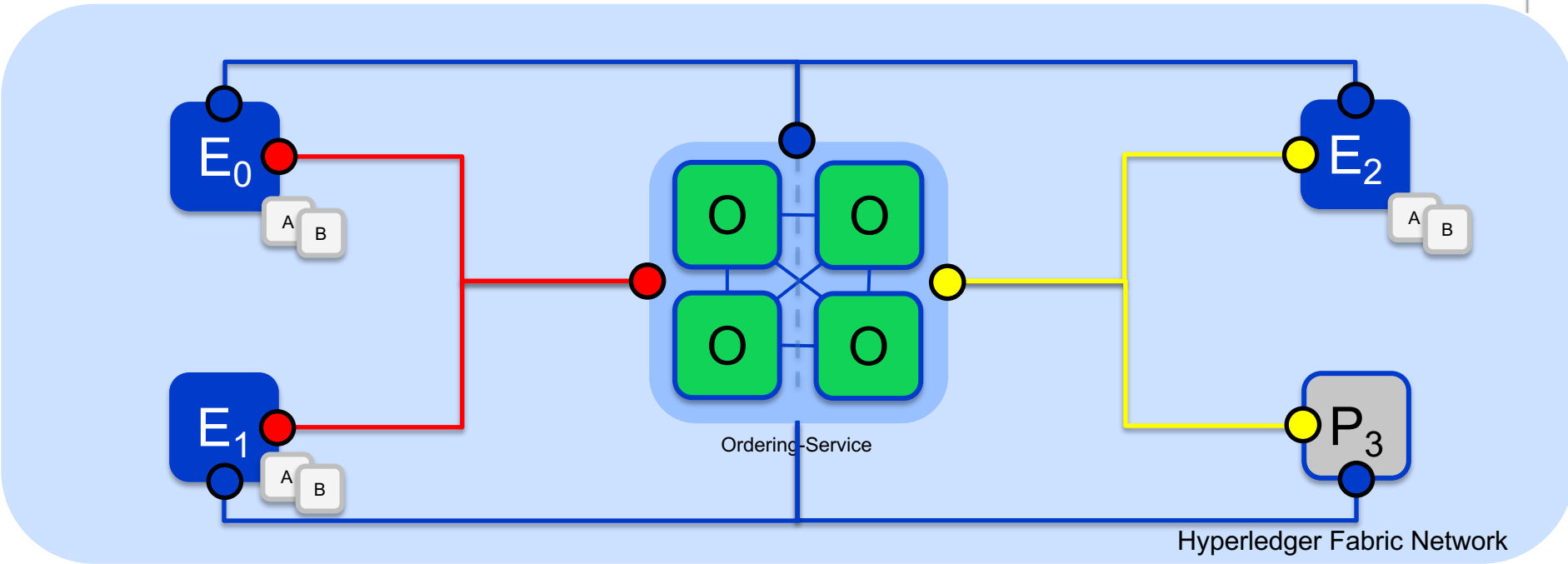
Bootstrap Network (4/6) – Create Channels



Channels are created on the ordering service:

$ peer channel create –o [orderer] ...

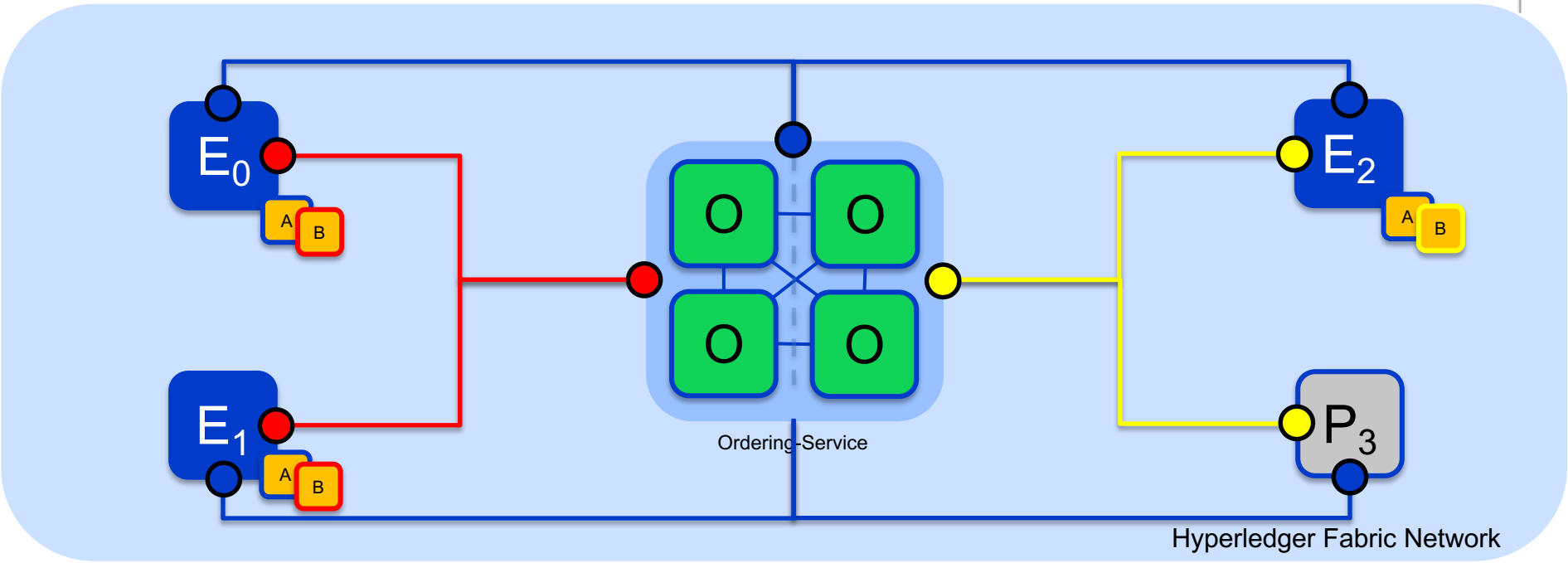
Bootstrap Network (5/6) – Join Channels



Peers that are permissioned can then join the channels they want to transact on:

$ peer channel join ...

Bootstrap Network (6/6) – Instantiate Chaincode



Peers finally instantiate the Chaincode on the channels they want to transact on:

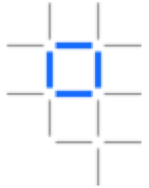
$ peer chaincode instantiate ... –P ‘policy’

An Endorsement Policy is specified and once instantiated chaincode can process transactions.

Technical Concepts

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

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...])

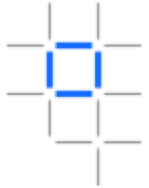
Where EXPR is either AND or OR and E is either a principal or nested EXPR

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'))

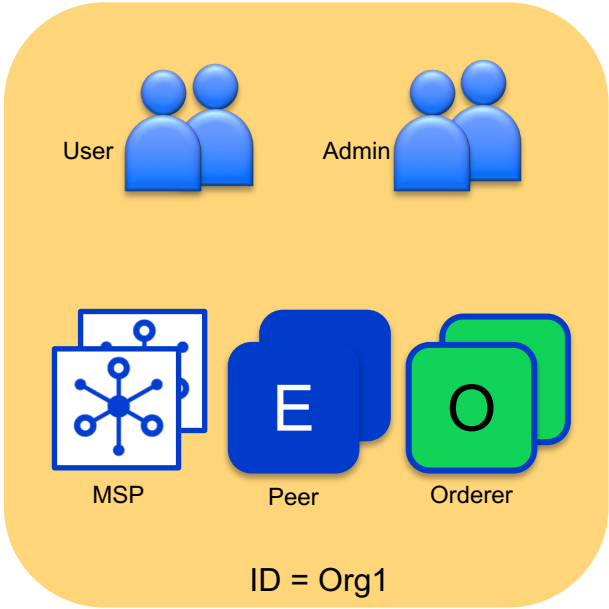
Technical Concepts

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

Organisations

Organisations define boundaries within a Fabric Blockchain Network

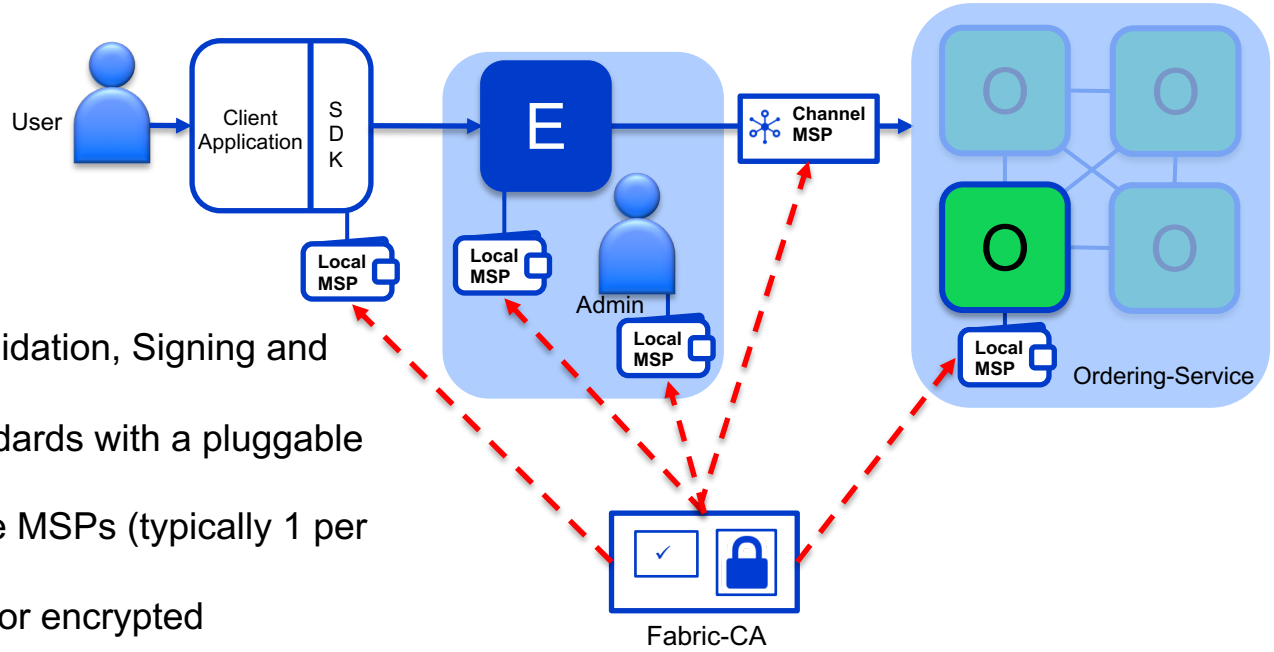
- Each organisation defines:
  - Membership Services Provider (MSP) for identities
  - Administrator(s)
  - Users
  - Peers
  - Orderers (optional)
- A network can include many organisations representing a consortium
- Each organisation has an ID



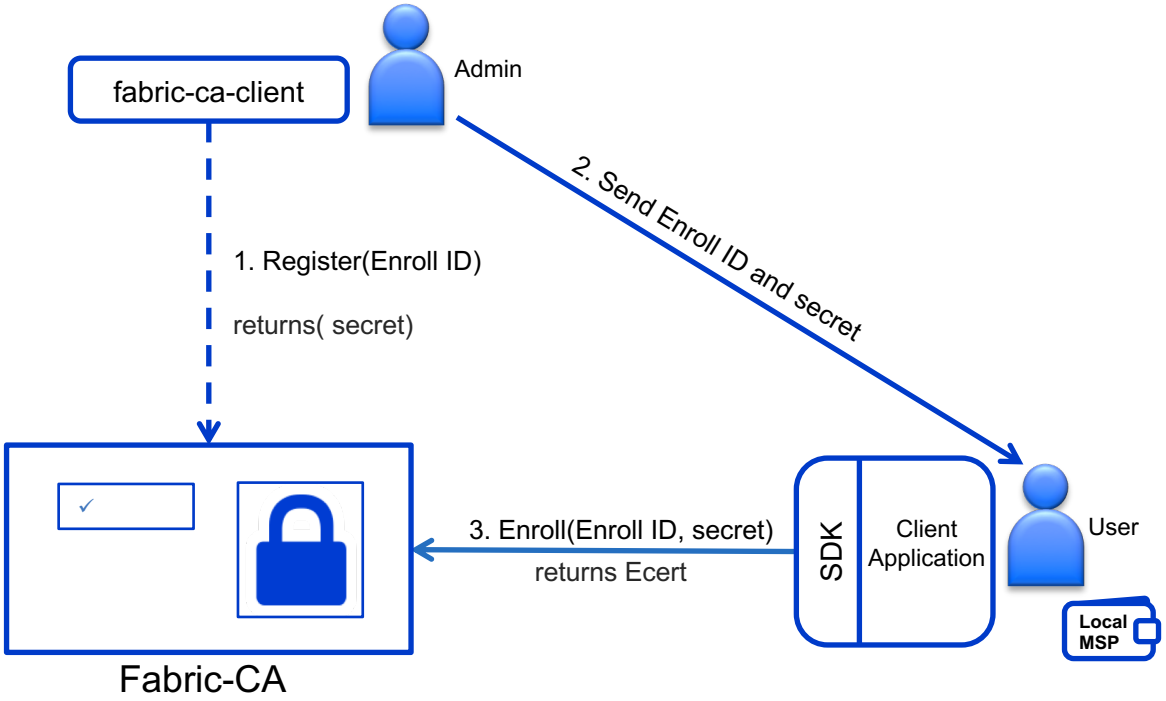
Membership Services Provider - Overview

A MSP manages a set of identities within a distributed Fabric network

- Provides identity for:
  - Peers and Orderers
  - Client Applications
  - Administrators
- Identities can be issued by:
  - Fabric-CA
  - An external CA
- Provides: Authentication, Validation, Signing and Issuance
- Supports different crypto standards with a pluggable interface
- A network can include multiple MSPs (typically 1 per org)
- Includes TLS crypto material for encrypted communications



New User Registration and Enrollment



Registration and Enrollment

- Admin registers new user with Enroll ID
- User enrolls and receives credentials
- Additional offline registration and enrollment options available

Introduction

Fabric Architecture Concepts

IBM Blockchain Platform

IBM Blockchain Strategy

Drive the development of **applications** for specific business use-cases, to be deployed to active **hybrid multi-cloud blockchain networks**

Services

Collaborate with services teams from ideation all the way to production

Ecosystem

Tap into our diverse ecosystem to develop strategic partnerships and create your competitive advantage

Solutions

Solve critical industry challenges by building and joining new business networks and applications

Platform

Develop, govern and operate hybrid multi-cloud enterprise blockchain networks with speed and security

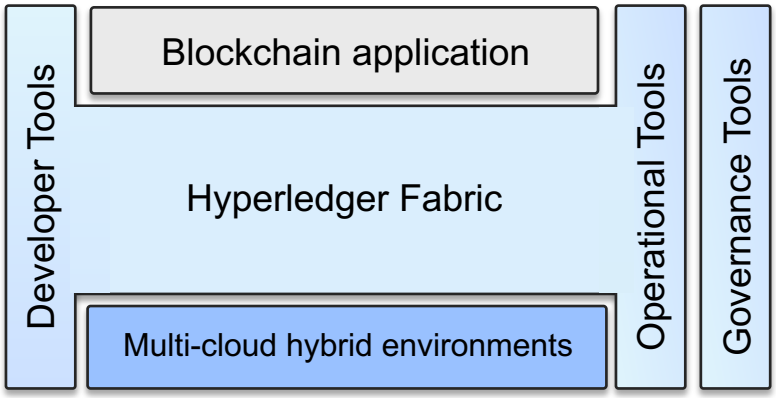
HYPERLEDGER

A founding, premier member of Hyperledger, IBM is committed to open source, standards & governance

Introducing the 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 hybrid and multi-cloud business network

- **Developer tools** that enable you 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 IBM Cloud Private (ICP), AWS and a highly secure and performant **IBM Cloud** environment



http://ibm.biz/Platform_Demo

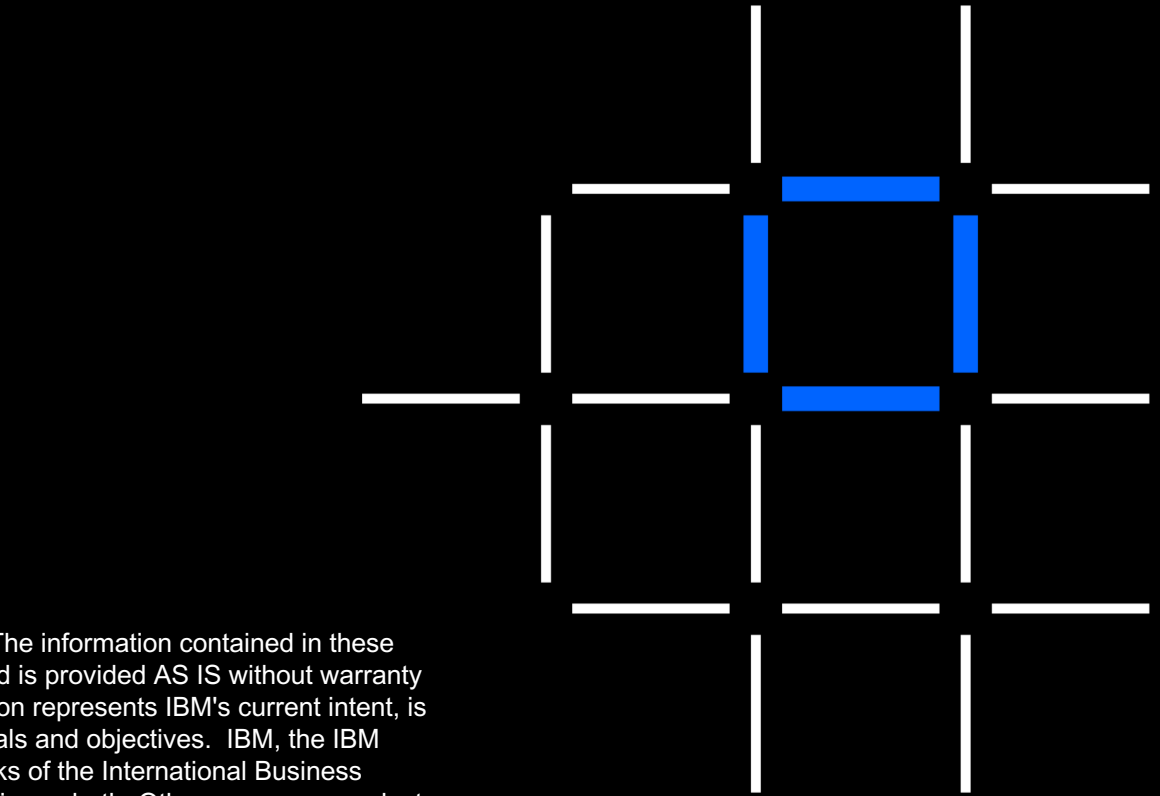
Thank you

IBM **Blockchain**

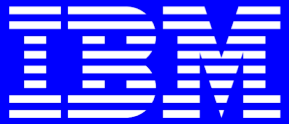
www.ibm.com/blockchain

developer.ibm.com/blockchain

www.hyperledger.org



IBM


IBM