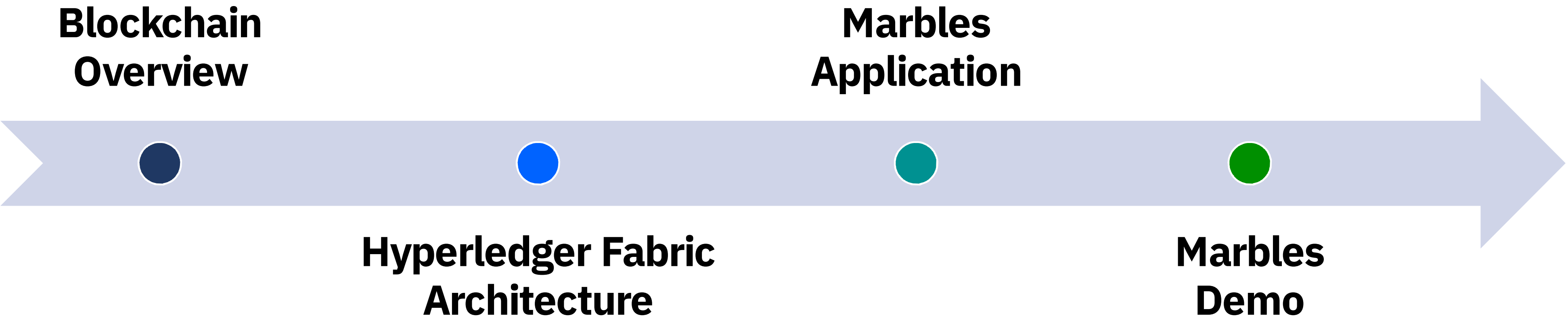
Deploy an asset-transfer app using Blockchain

Presenters: Raheel Zubairy, Mihir Shah, Dhyey Shah

https://github.com/IBM-Blockchain/marbles

Agenda



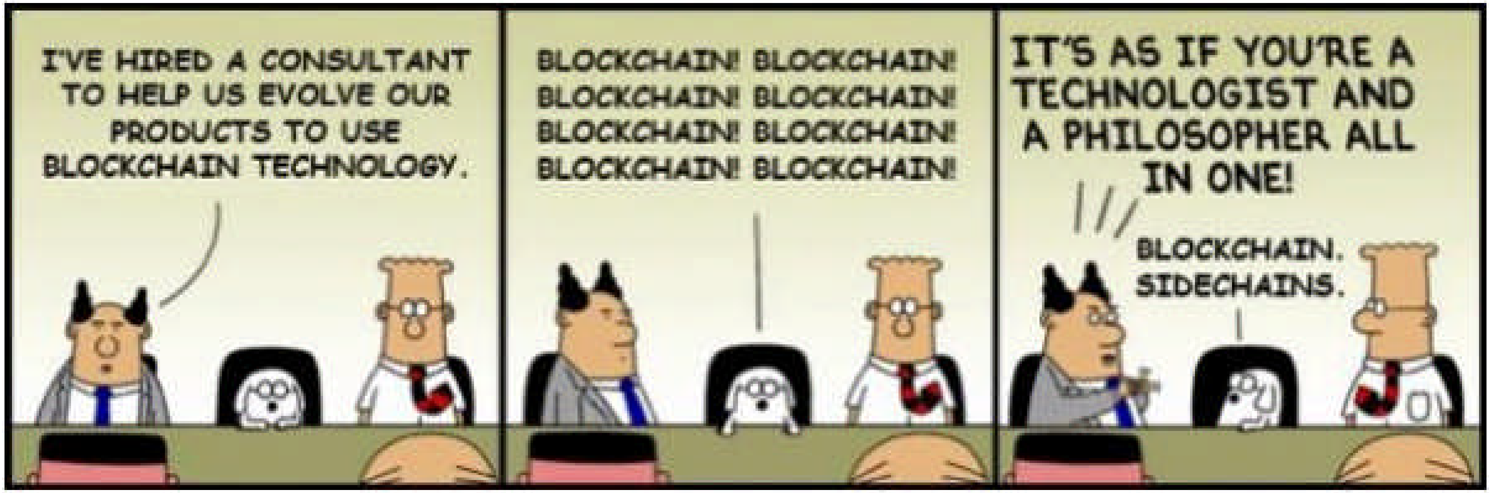
Blockchain Overview

Hyperledger Fabric Architecture

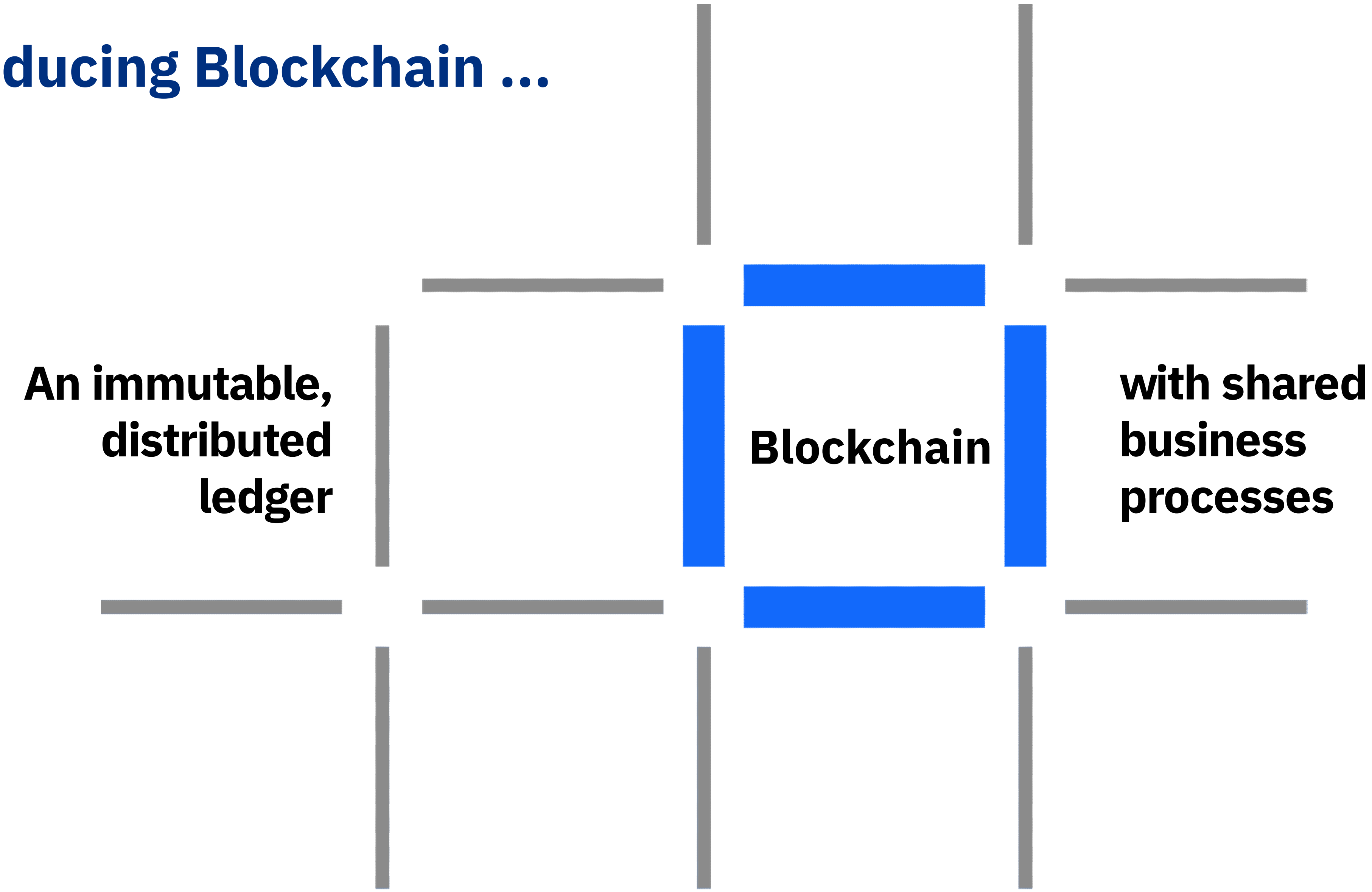
Marbles Application

Marbles Demo

I THINK WE SHOULD BUILD A BLOCKCHAIN
UH-OH
S.Adams
E-mail: SCOTTADAMS@AOL.COM
DOES HE UNDERSTAND WHAT HE SAID OR IS IT SOMETHING HE SAW IN A TRADE MAGAZINE AD?
© 1998 United Feature Syndicate, Inc.(NYC)
WHAT COLOR DO YOU WANT THAT BLOCKCHAIN?
I THINK MAUVE HAS THE MOST RAM.


I'VE HIRED A CONSULTANT TO HELP US EVOLVE OUR PRODUCTS TO USE BLOCKCHAIN TECHNOLOGY.
BLOCKCHAIN! BLOCKCHAIN! BLOCKCHAIN! BLOCKCHAIN! BLOCKCHAIN! BLOCKCHAIN! BLOCKCHAIN! BLOCKCHAIN!
IT'S AS IF YOU'RE A TECHNOLOGIST AND A PHILOSOPHER ALL IN ONE!
BLOCKCHAIN. SIDECHAINS.

Introducing Blockchain ...



An immutable,
distributed
ledger

Blockchain

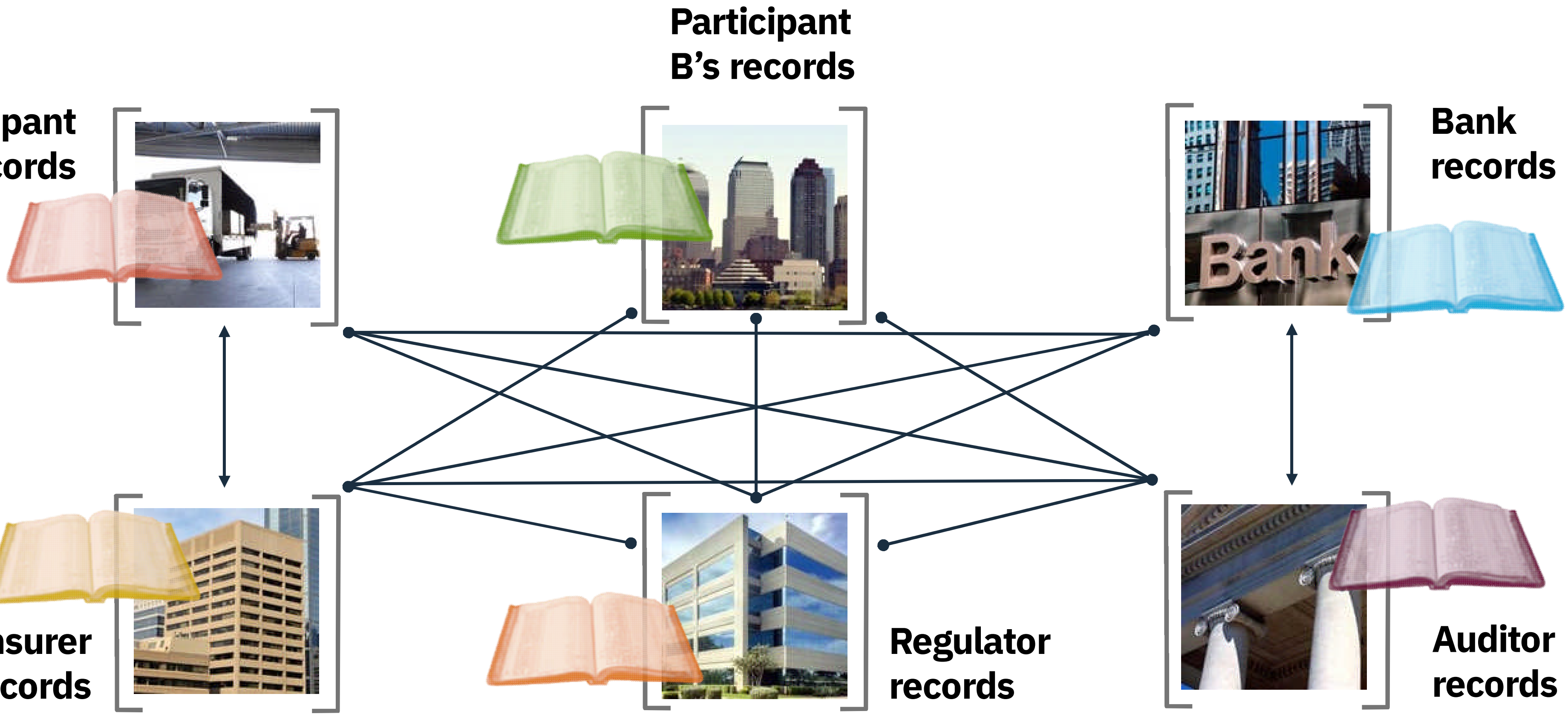
with shared
business
processes

Ledgers are key

Ledger is THE system of record for a business. Business will have multiple ledgers for multiple business networks in which they participate.

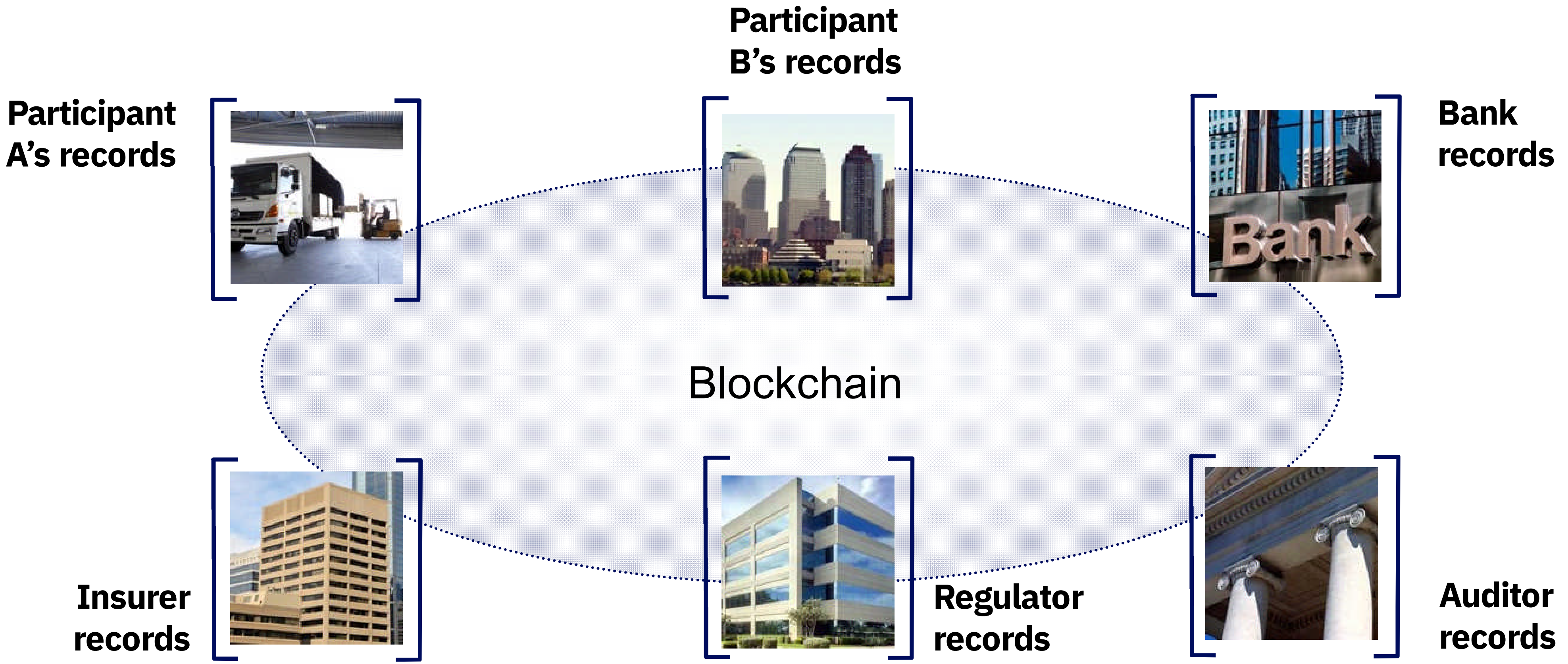
- Transaction – an asset transfer onto or off the ledger
  - John gives a car to Anthony (simple)
- Contract – conditions for transaction to occur
  - If Anthony pays John money, then car passes from John to Anthony (simple)
  - If car won't start, funds do not pass to John (as decided by third party arbitrator) (more complex)

Problem …



… inefficient, expensive, vulnerable

A shared, replicated, permissioned ledger …



Participant A’s records

Participant B’s records

Bank records

Blockchain

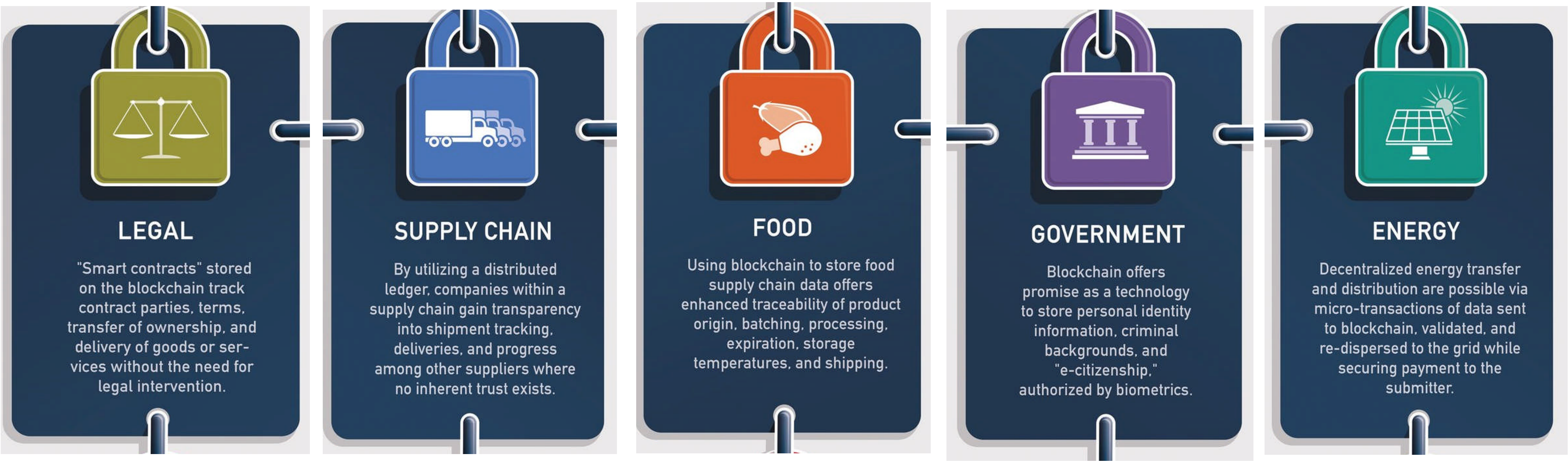
Insurer records

Regulator records

Auditor records

… with consensus, provenance, immutability and finality

Blockchain for Every Industry



LEGAL

"Smart contracts" stored on the blockchain track contract parties, terms, transfer of ownership, and delivery of goods or services without the need for legal intervention.

SUPPLY CHAIN

By utilizing a distributed ledger, companies within a supply chain gain transparency into shipment tracking, deliveries, and progress among other suppliers where no inherent trust exists.

FOOD

Using blockchain to store food supply chain data offers enhanced traceability of product origin, batching, processing, expiration, storage temperatures, and shipping.

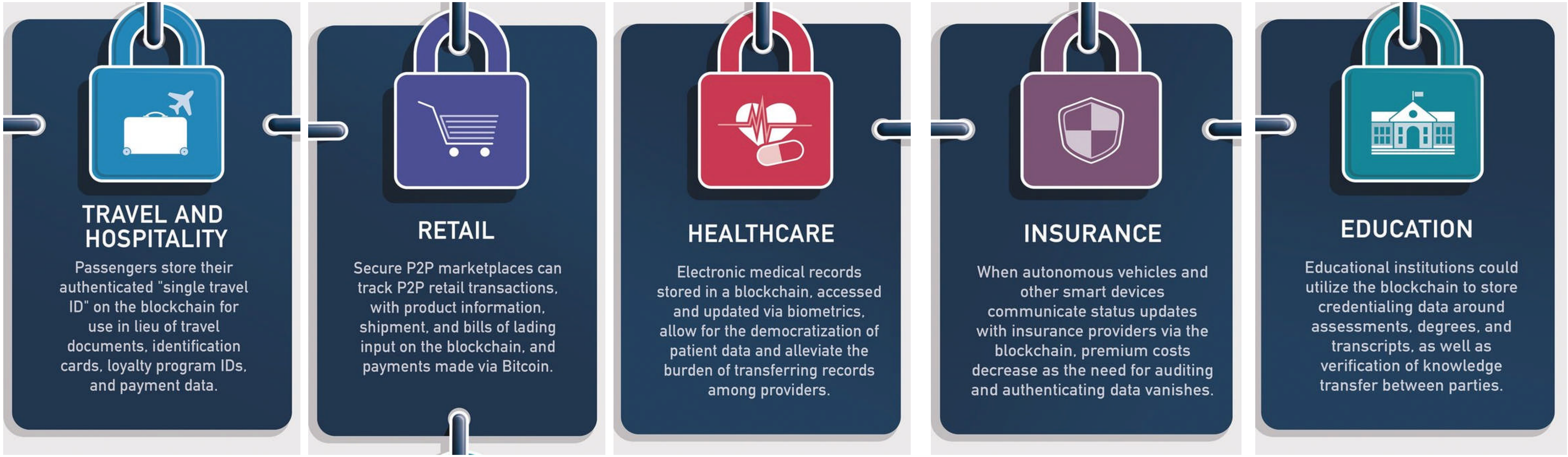
GOVERNMENT

Blockchain offers promise as a technology to store personal identity information, criminal backgrounds, and "e-citizenship," authorized by biometrics.

ENERGY

Decentralized energy transfer and distribution are possible via micro-transactions of data sent to blockchain, validated, and re-dispersed to the grid while securing payment to the submitter.

Blockchain for Every Industry



TRAVEL AND HOSPITALITY

Passengers store their authenticated "single travel ID" on the blockchain for use in lieu of travel documents, identification cards, loyalty program IDs, and payment data.

RETAIL

Secure P2P marketplaces can track P2P retail transactions, with product information, shipment, and bills of lading input on the blockchain, and payments made via Bitcoin.

HEALTHCARE

Electronic medical records stored in a blockchain, accessed and updated via biometrics, allow for the democratization of patient data and alleviate the burden of transferring records among providers.

INSURANCE

When autonomous vehicles and other smart devices communicate status updates with insurance providers via the blockchain, premium costs decrease as the need for auditing and authenticating data vanishes.

EDUCATION

Educational institutions could utilize the blockchain to store credentialing data around assessments, degrees, and transcripts, as well as verification of knowledge transfer between parties.

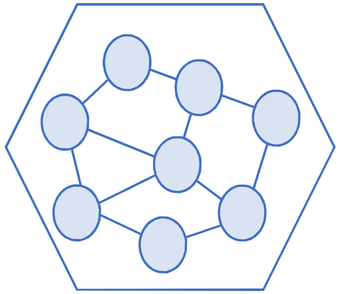
What is Blockchain?

Wikipedia definition:
-A continuously growing list of records, called blocks, which are linked and secured using cryptography. Once recorded, the data in any given block cannot be altered retroactively without the alteration of all subsequent blocks

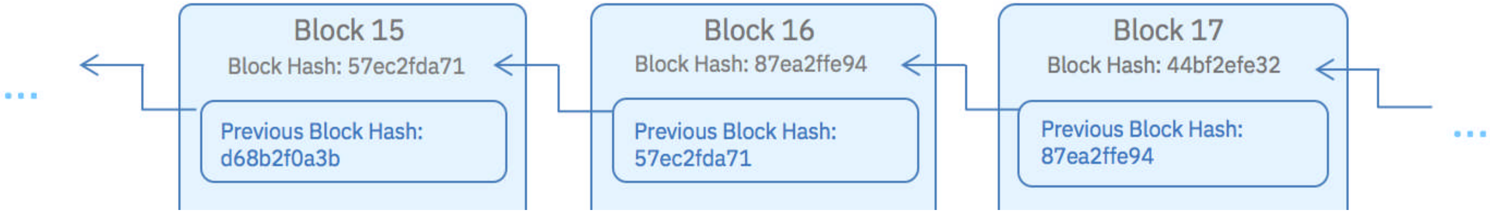
-A distributed ledger that can record transactions in a verifiable and permanent way, typically managed by a peer-to-peer network collectively adhering to a protocol for validating new blocks

Key Concepts

- **Peers / Nodes**
  - Members of the blockchain network

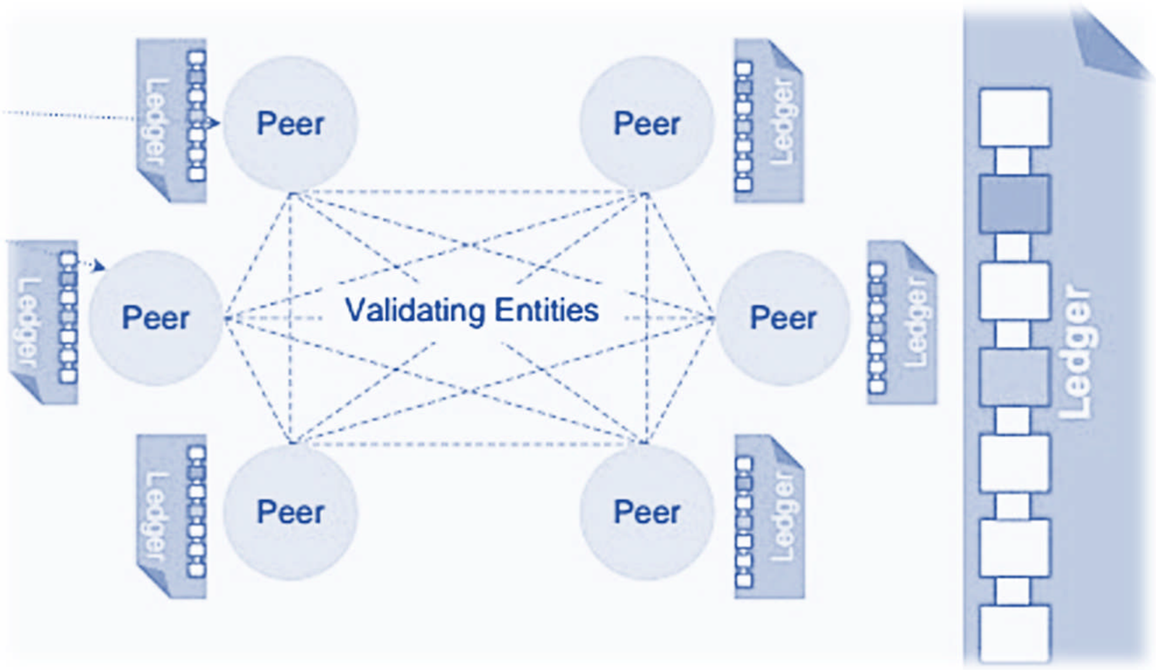


- **Cryptography**
  - Each block contains record-state and a hash to verify integrity



Key Concepts

- **Consensus**
  - The process by which peers agree to the addition of next block



Bitcoin and Blockchain

bitcoin

- Cryptocurrency
- First blockchain application
- Permissionless – Open to anyone
- Consensus achieved through ‘Proof of Work’
- Requires mining - resource intensive

BLOCKCHAIN

Blockchain for business differs in key areas:

- Identity over anonymity
- Creating private secure networks
- Smart contracts to enable valid transactions
- Designed for business use cases

Blockchain providers

HYPERLEDGER

- Network privacy: Channels, eCerts
- Access: Permissioned
- Consensus (Ordering): Solo, Kafka

ETHEREUM

- Network privacy: None
- Access: Permissionless
- Consensus: Proof of Work

corda

- Network privacy: Semi-private
- Access: Doorman service
- Consensus: Validity, Uniqueness

ripple

- Network privacy: None
- Access: Permissionless
- Consensus: Distributed Agreement Protocol

Requirements of blockchain for business

Append-only distributed system of record shared across business network

Shared ledger

Smart contract

Business terms embedded in transaction database & executed with transactions

Ensuring appropriate visibility; transactions are secure, authenticated & verifiable

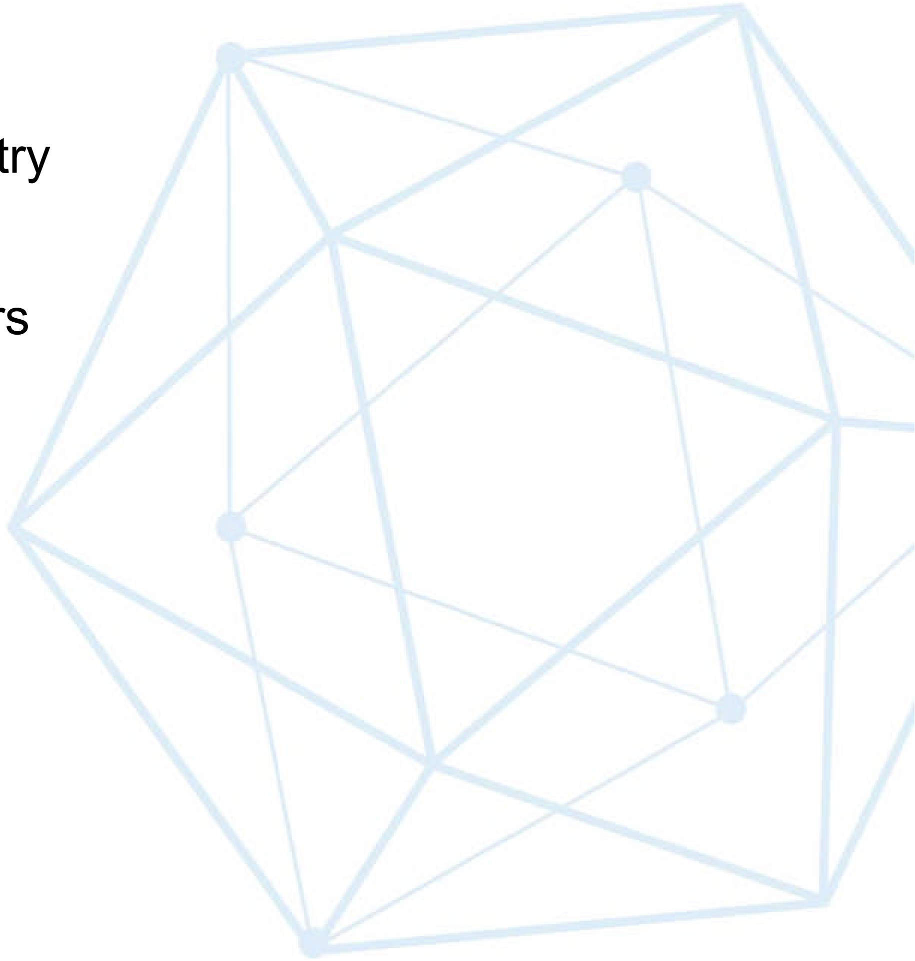
Privacy

Trust

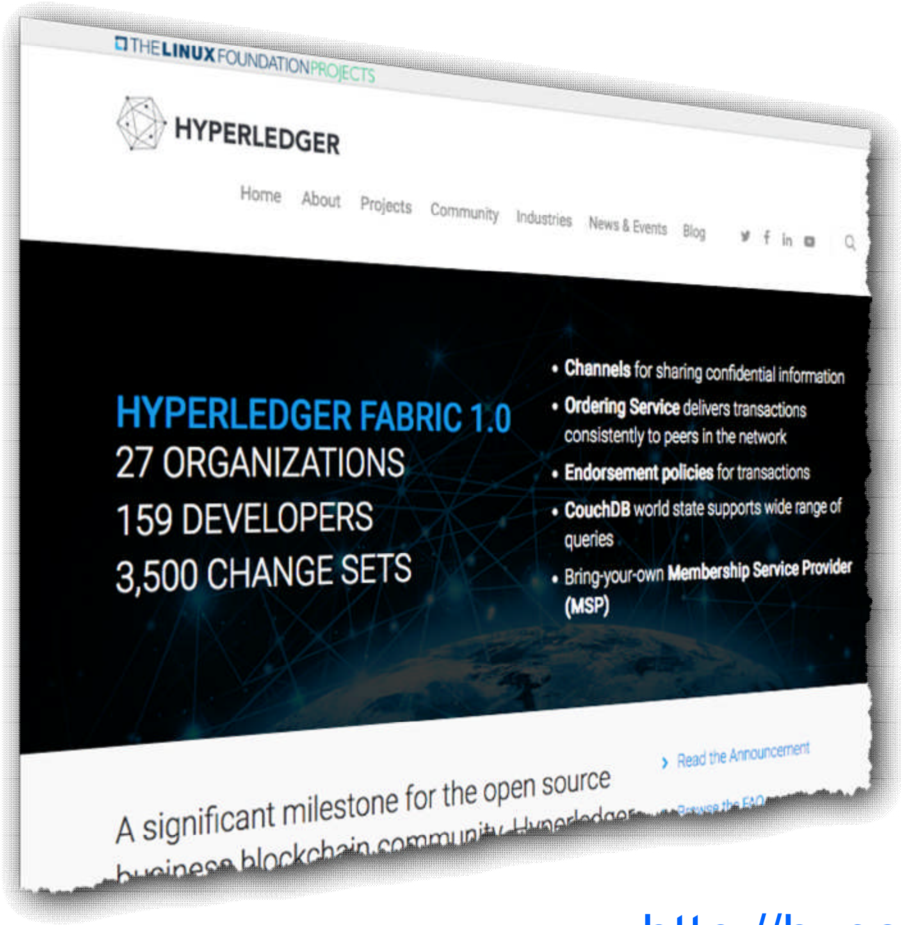
Transactions are endorsed by relevant participants

Hyperledger: A Linux Foundation project

- A collaborative effort created to advance cross-industry blockchain technologies for business
- Announced December 2015, more than 150 members
- Open source, open standards, open governance
- Five frameworks and three tools projects
- IBM is a premier member of Hyperledger



Hyperledger Fabric: Distributed ledger platform



THE LINUX FOUNDATION PROJECTS

HYPERLEDGER

HYPERLEDGER FABRIC 1.0
27 ORGANIZATIONS
159 DEVELOPERS
3,500 CHANGE SETS

- **Channels** for sharing confidential information
- **Ordering Service** delivers transactions consistently to peers in the network
- **Endorsement policies** for transactions
- **CouchDB** world state supports wide range of queries
- Bring-your-own **Membership Service Provider (MSP)**

A significant milestone for the open source business blockchain community. Hyperledger

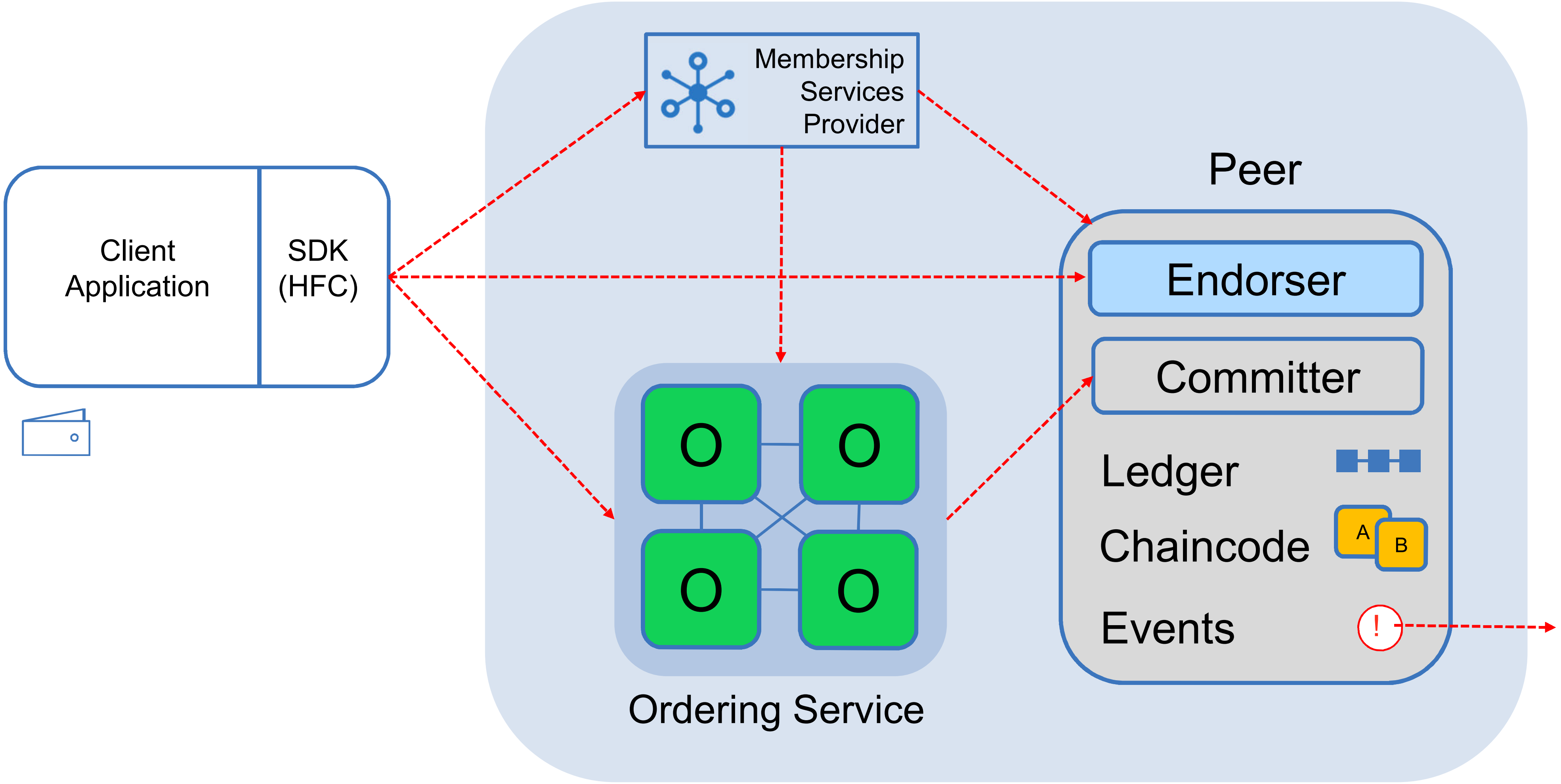
- An implementation of blockchain technology that is a foundation for developing blockchain applications
- Emphasis on ledger, smart contracts, consensus, confidentiality, resiliency and scalability.
- V1.0 released July 2017
  - 159 developers from 27 organizations
  - IBM is one contributor of code, IP and development effort to Hyperledger Fabric

http://hyperledger-fabric.readthedocs.io/

What is Hyperledger Fabric

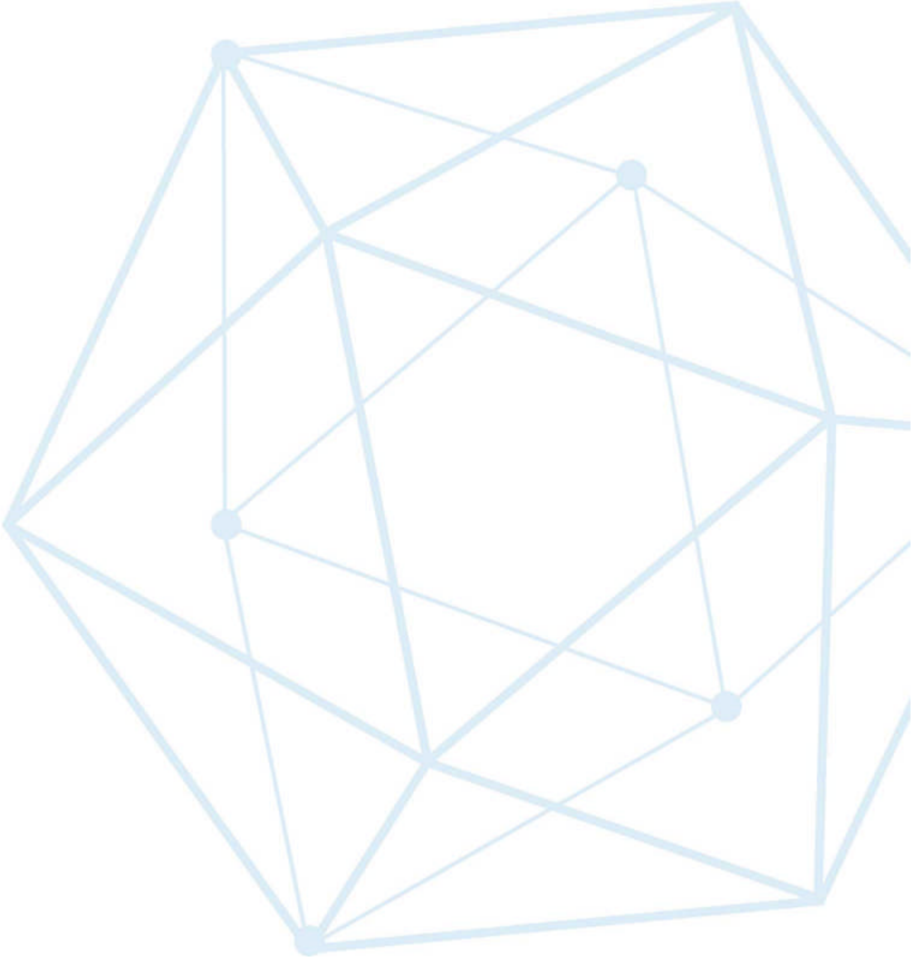
- Hyperledger Fabric
  - An implementation of blockchain technology that is intended as a foundation for developing blockchain applications
  - Key technical features:
    - A shared ledger and smart contracts implemented as “chaincode”
    - Privacy and permissioning through membership services
    - Modular architecture and flexible hosting options

Hyperledger Fabric V1 Architecture



Hyperledger Fabric V1 - Deep Dive Topics

- Network setup
- Channels and Ordering Service
- Network Consensus
- Endorsement Policies
- Permissioned ledger access
- Pluggable world-state



Network Setup

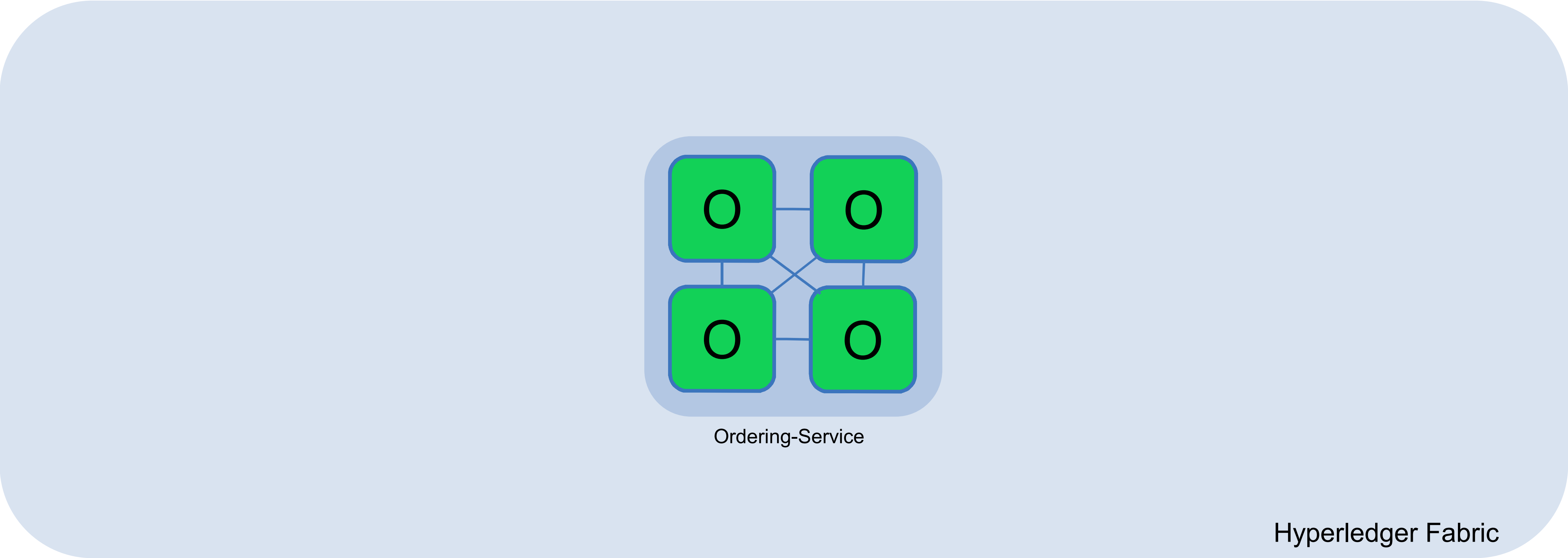
Nodes and roles

🟦 **Endorsing Peer**: Specialized committing peer that receives a transaction proposal for endorsement, responds granting or denying endorsement. Must hold smart contract (chaincode)

🟩 **Ordering Nodes (service):** Approves the inclusion of transaction blocks into the ledger and communicates with committing and endorsing peer nodes. Does not hold smart contract (chaincode). Does not hold ledger.

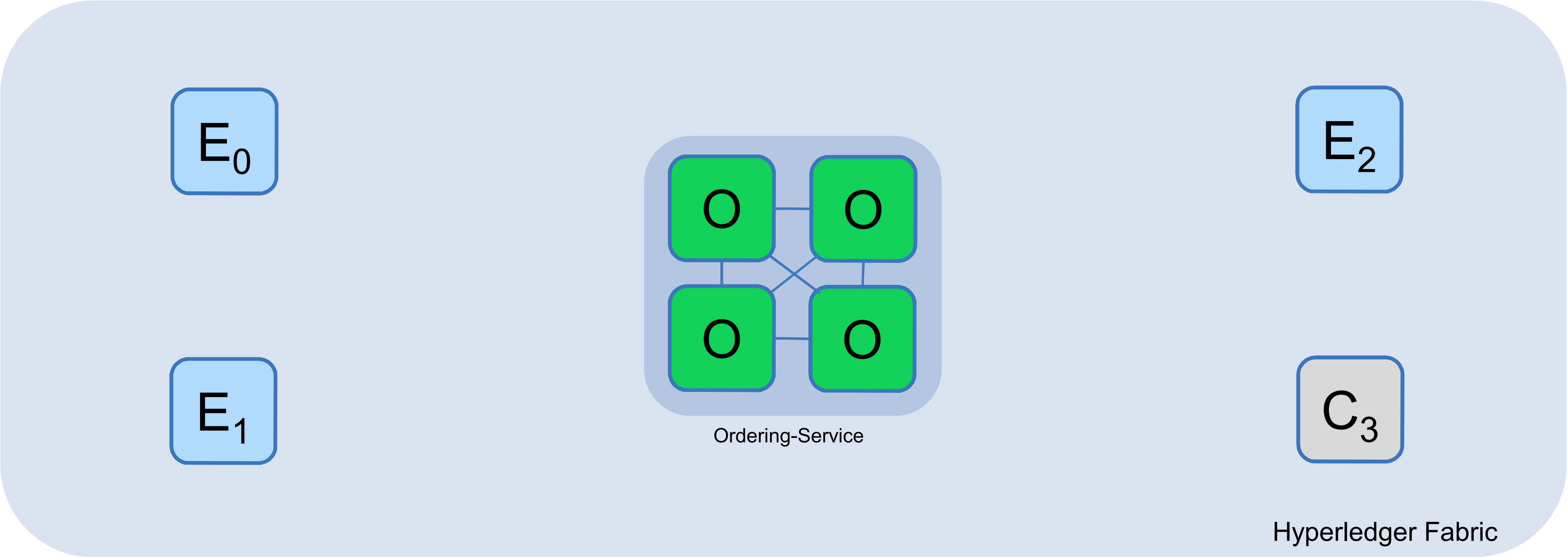
⬜ **Committing Peer**: Maintains ledger and state. Commits transactions.

Bootstrapping the Network (1/6) – Configure & start Ordering Service



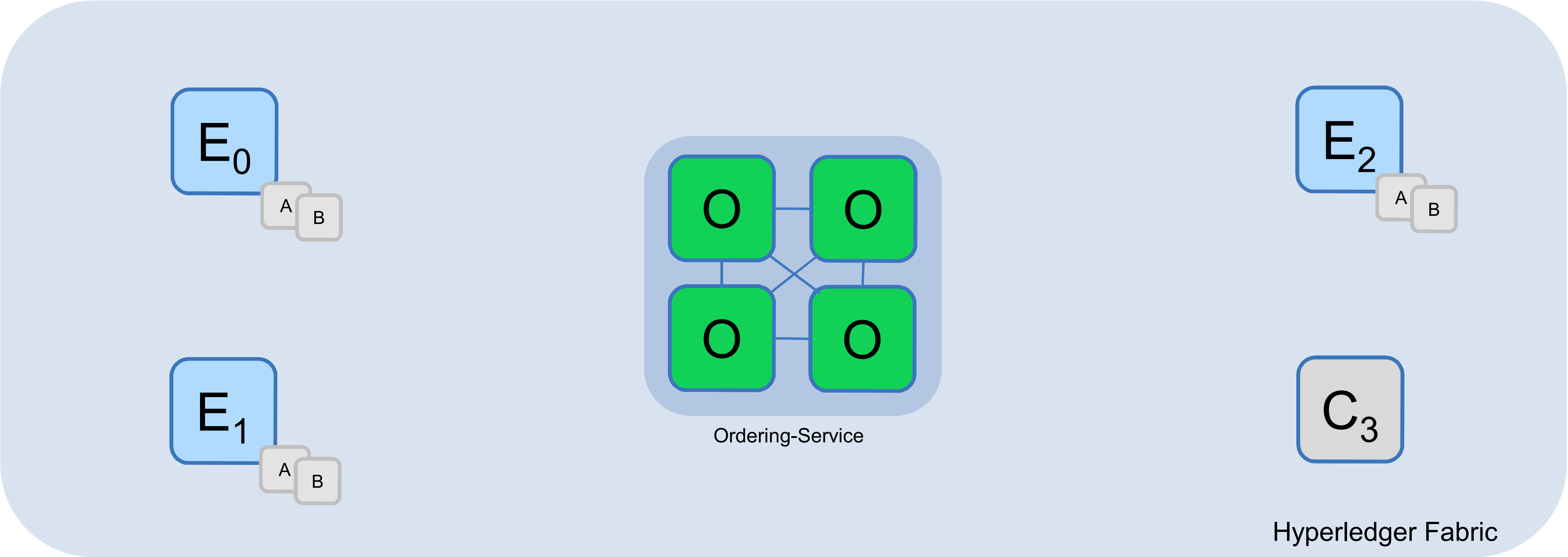
- An Ordering Service is configured and started for other network peers to use

Bootstrapping the Network (2/6) – Configure and Start Peer Nodes



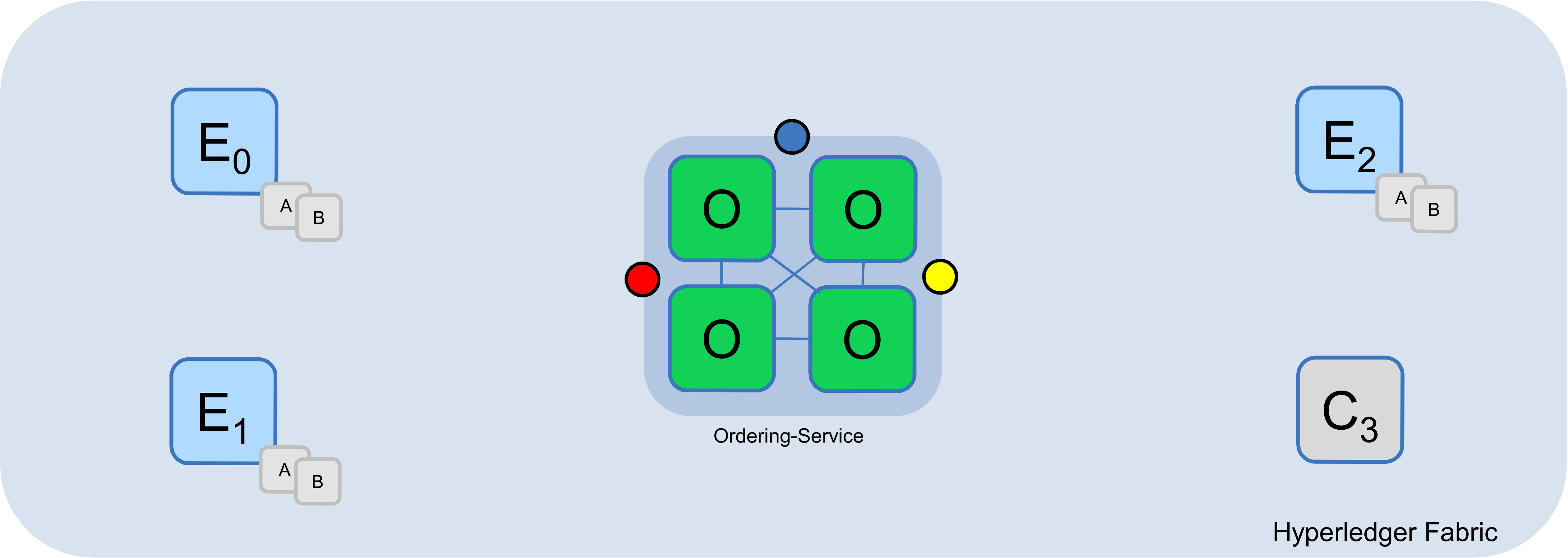
- A peer is configured and started for each Endorser or Committer in the network

Bootstrapping the Network (3/6) – Install Chaincode



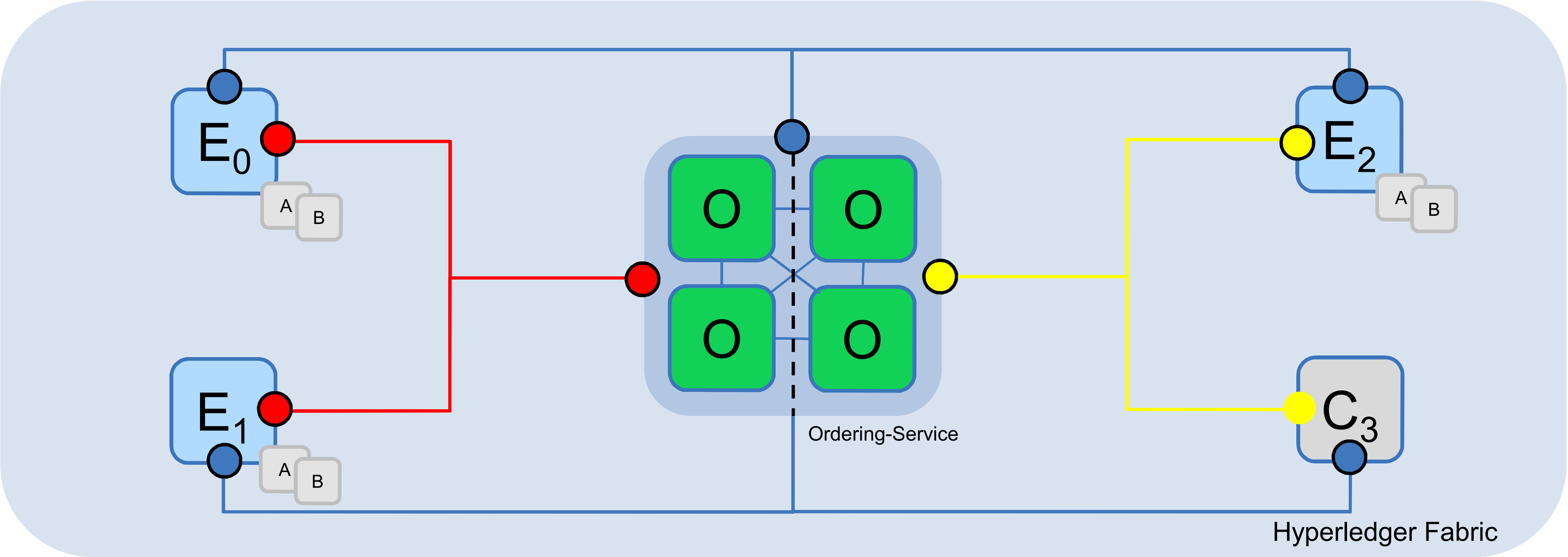
- Chaincode is installed onto each Endorsing Peer that needs to execute it

Bootstrapping the Network (4/6) – Create Channels



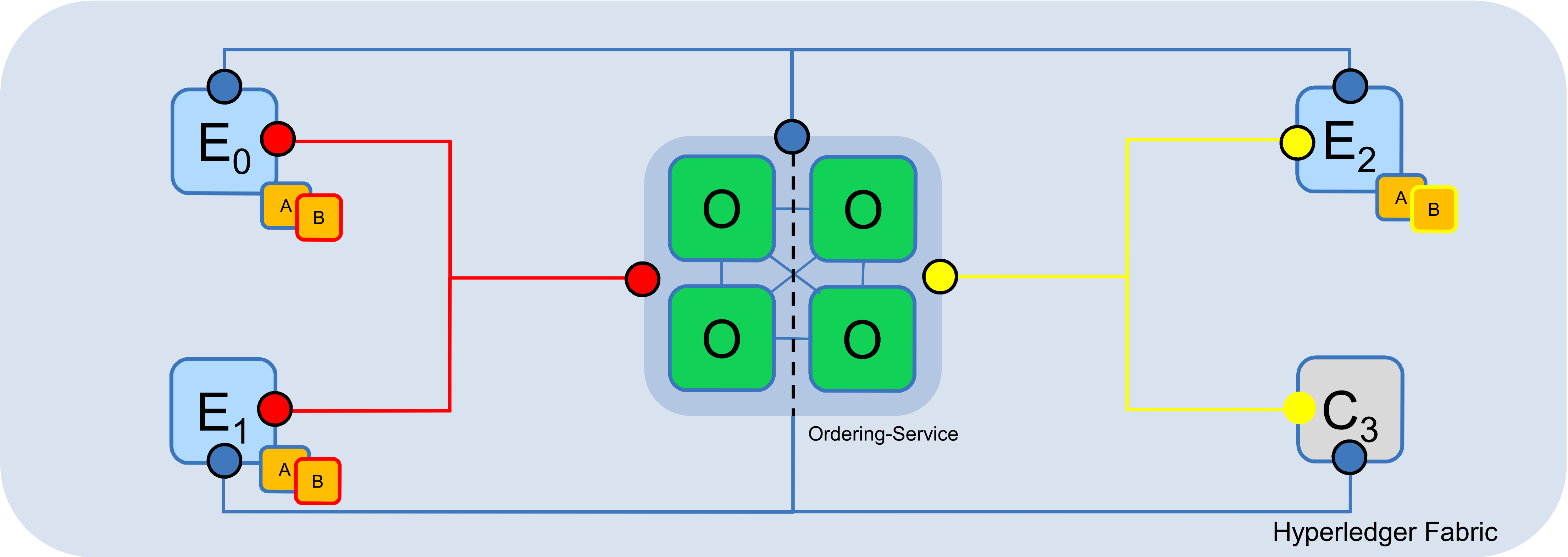
- Channels are created on the ordering service

Bootstrapping the Network (5/6) – Join Channels



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

Bootstrapping the Network (6/6) – Instantiate Chaincode

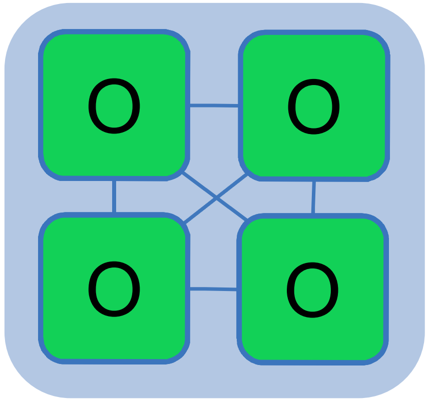


- Peers finally instantiate the Chaincode on the channels they want to transact on
- Once instantiated a Chaincode is live and can process transaction requests
- Endorsement Policy must be specified at instantiation time

Channels and Ordering Service

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

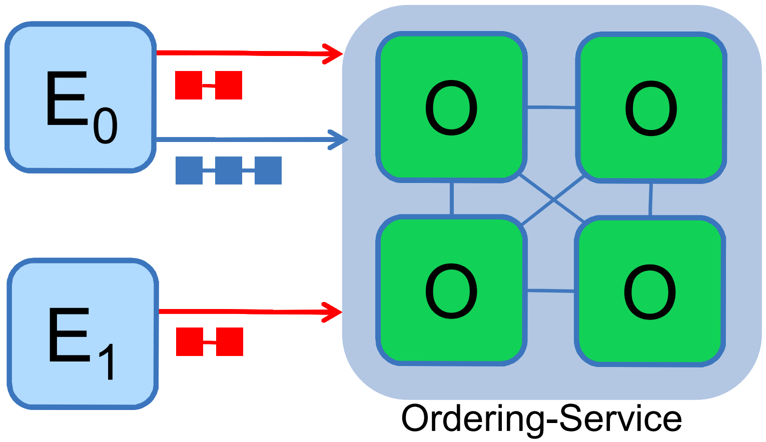
- Single node for development

–Kafka : Crash fault tolerant consensus

- 3 nodes minimum
- Odd number of nodes recommended

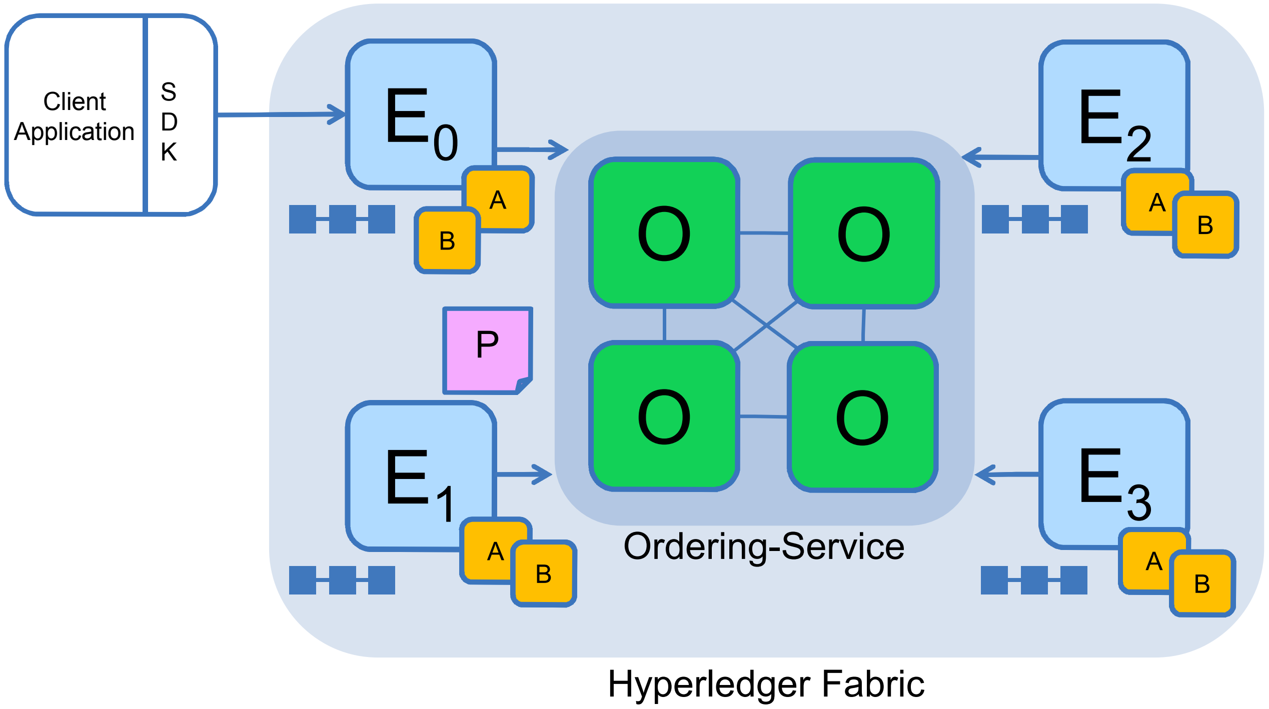
Channels

Separate channels isolate transactions on different ledgers

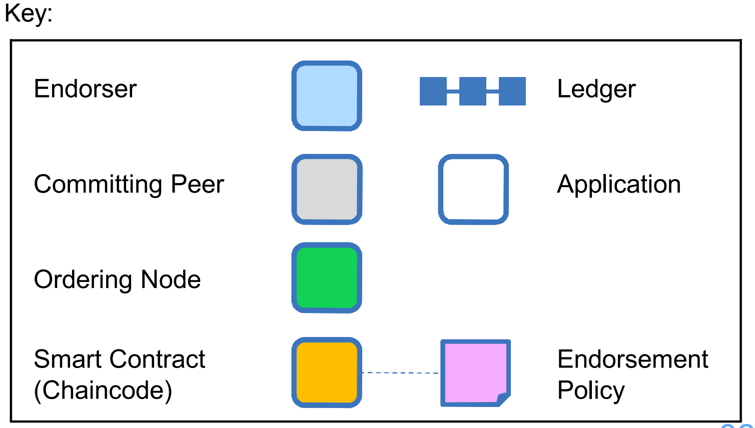


- Chaincode is installed on peers that need to access the worldstate
- Chaincode is instantiated on specific channels for specific peers
- Ledgers exist in the scope of a channel
  - Ledgers can be shared across an entire network of peers
  - Ledgers can be included only on a specific set of participants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

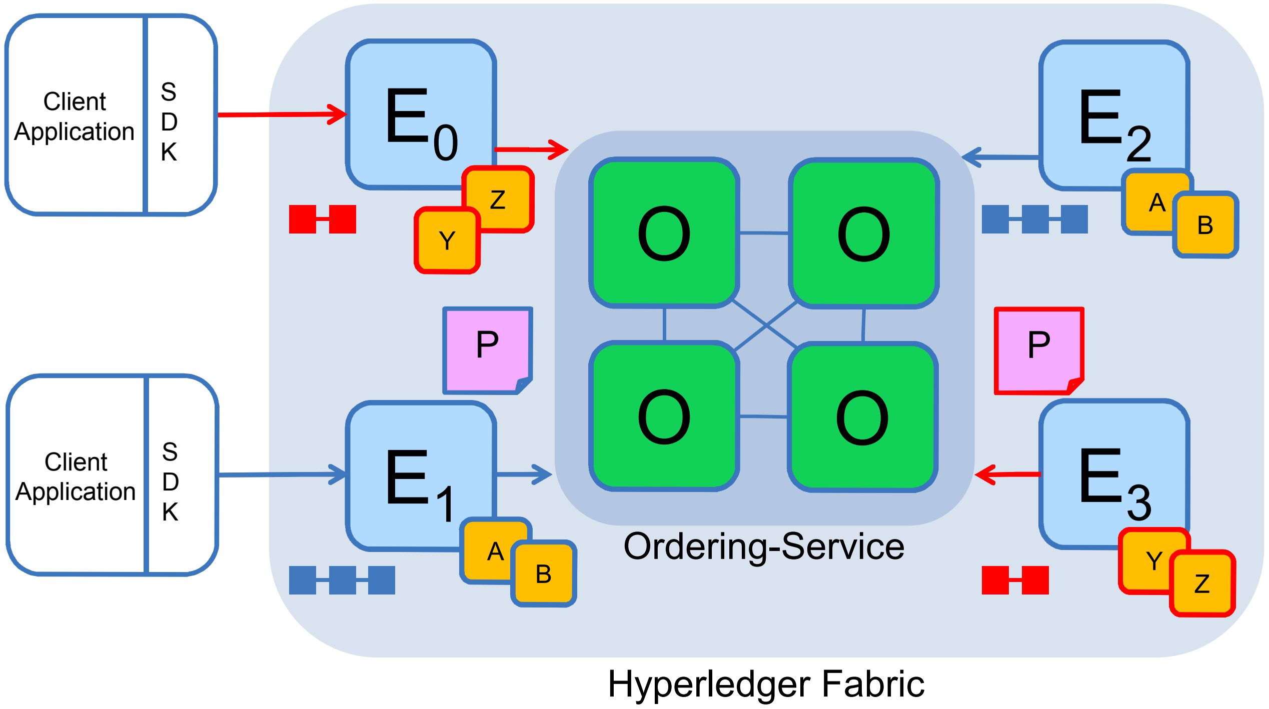
Single Channel Network



- 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

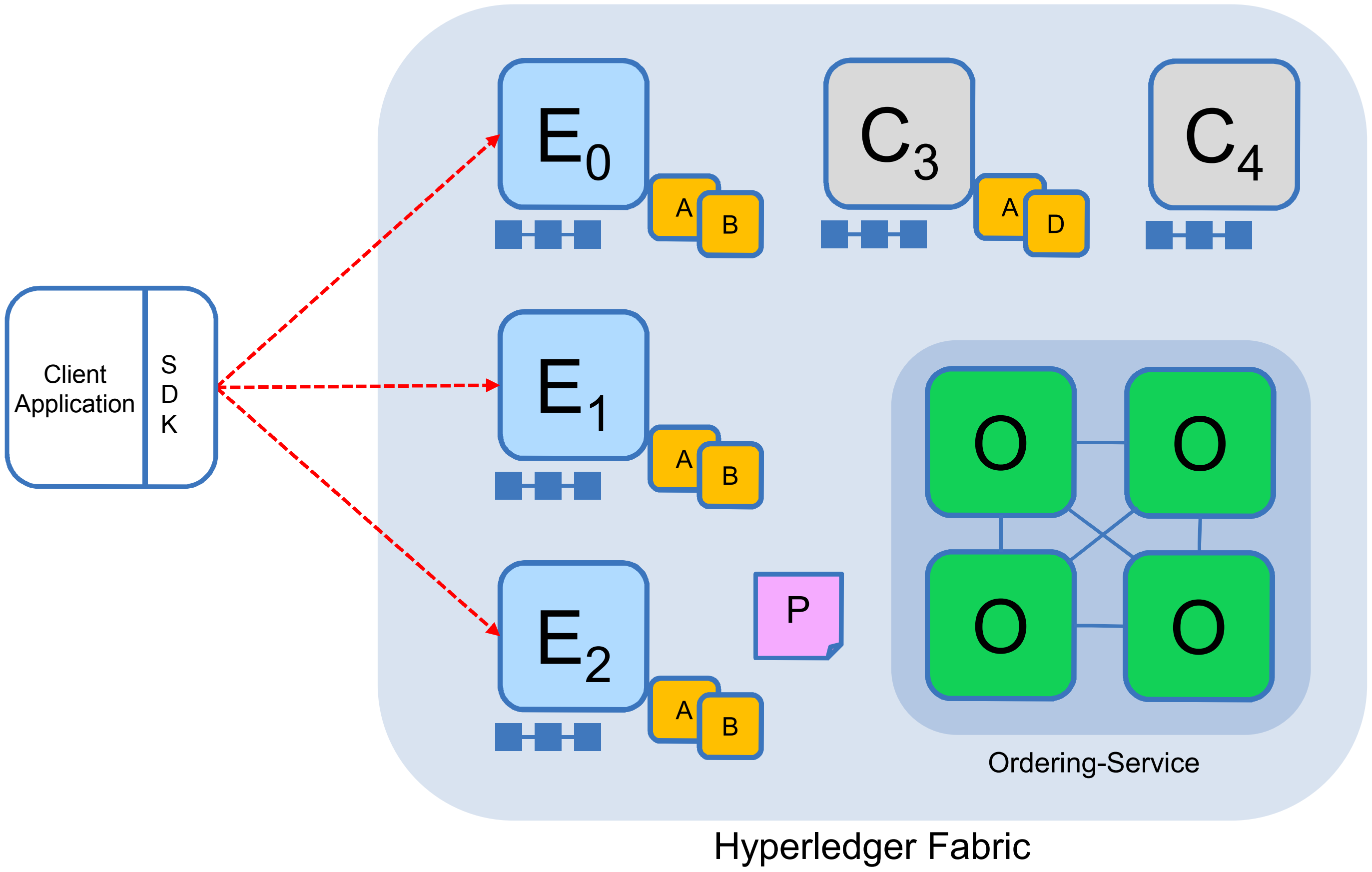
- Peers E_1 and E_2 connect to the blue channel for chaincodes A and B

Key:

Endorser	🟦	■–■–■	Ledger
Committing Peer	⬜	□	Application
Ordering Node	🟩		
Smart Contract (Chaincode)	🟧	🟪	Endorsement Policy

Network Consensus

Sample transaction: Step 1/7 – Propose transaction



Application proposes transaction

Endorsement policy:

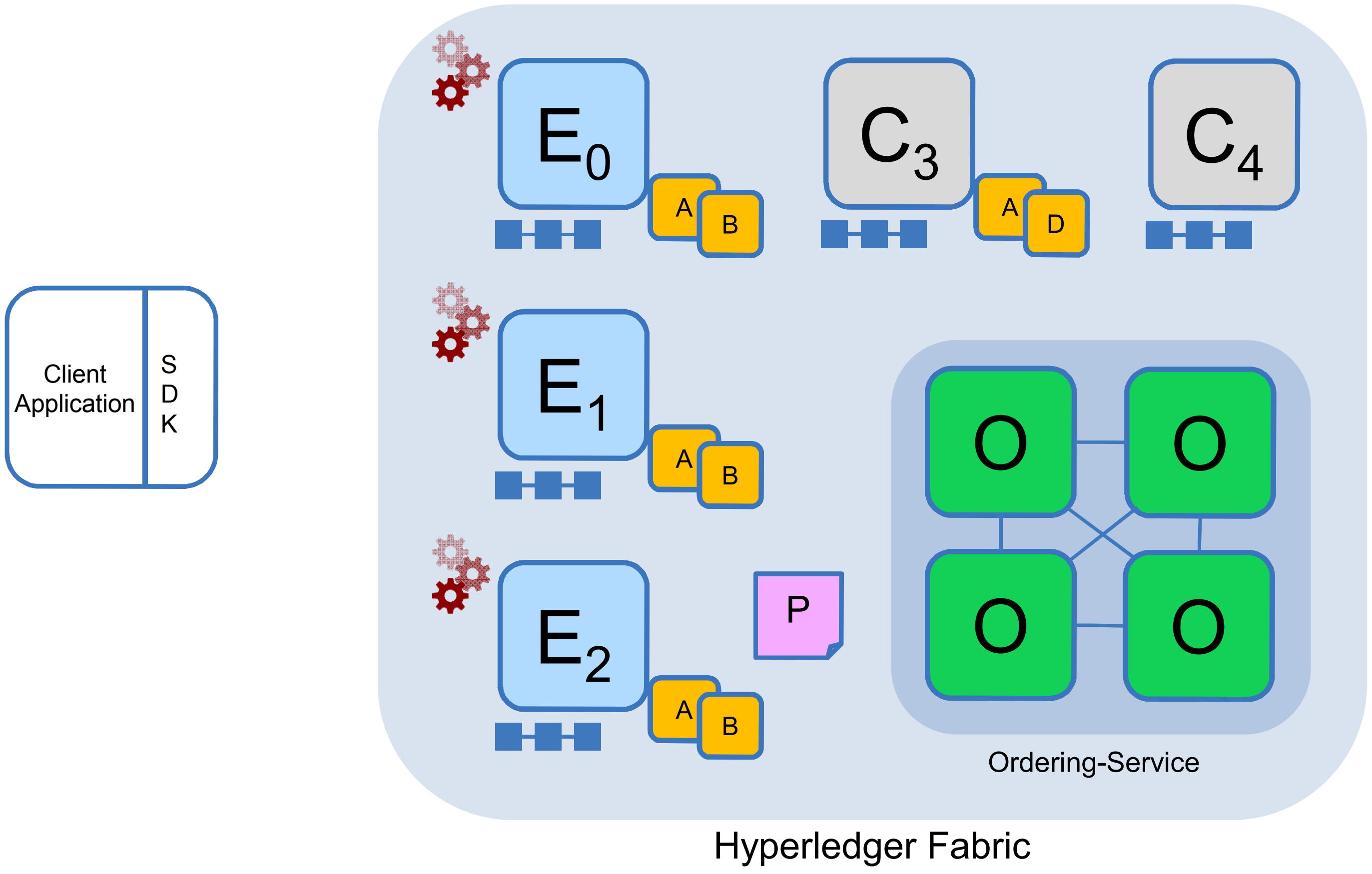
- “$E_{0,}$ E_1 and E_2 must sign”
- (C_3, C_4 are not part of the policy)

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

Key:

Endorser	🟦	■-■-■	Ledger
Committing Peer	⬜	□	Application
Ordering Node	🟩		
Smart Contract (Chaincode)	🟨	🟪	Endorsement Policy

Sample transaction: Step 2/7 – Execute proposal



Endorsers Execute Proposals

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

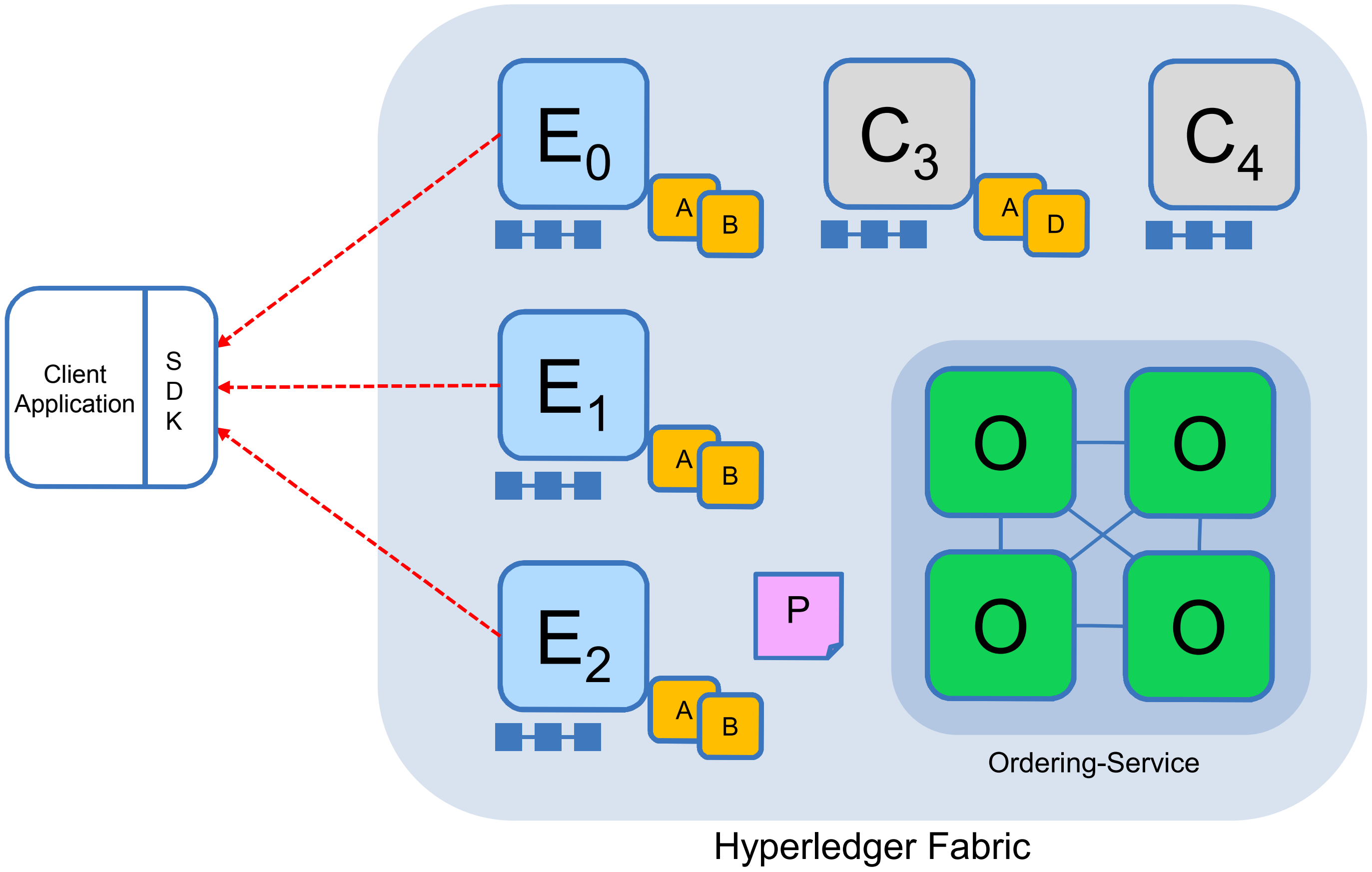
Each execution will capture the set of Read and Written data, called RW sets, which will now flow in the fabric.

Transactions can be signed & encrypted

Key:

Endorser	🟦	■-■-■	Ledger
Committing Peer	⬜	□	Application
Ordering Node	🟩		
Smart Contract (Chaincode)	🟨	🟪	Endorsement Policy

Sample transaction: Step 3/7 – Proposal Response



Application receives responses

RW sets are asynchronously returned to application

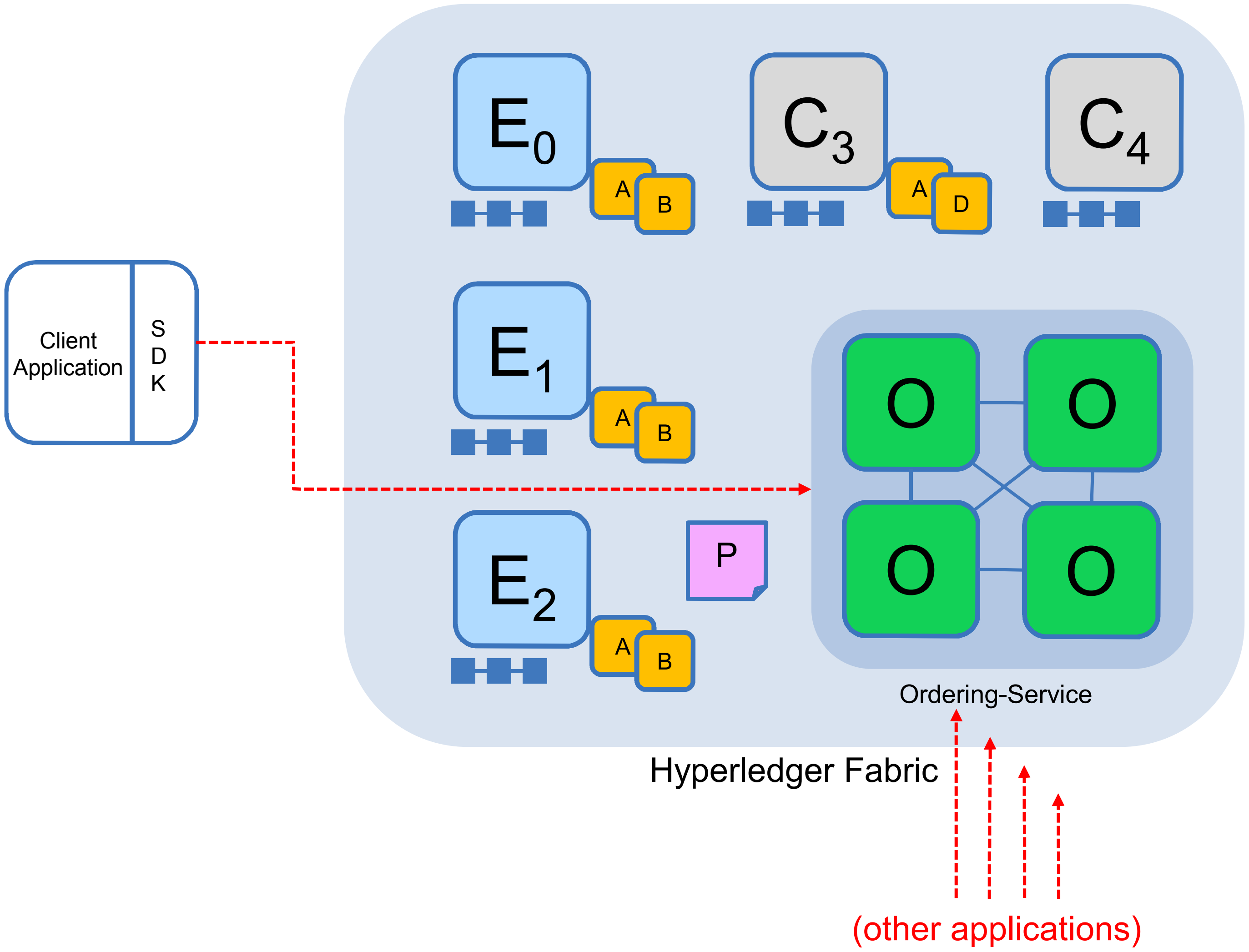
The RW sets are signed by each endorser, and also includes each record version number

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

Key:

Endorser	🟦	■-■-■	Ledger
Committing Peer	⬜	□	Application
Ordering Node	🟩		
Smart Contract (Chaincode)	🟨	🟪	Endorsement Policy

Sample transaction: Step 4/7 – Order Transaction



Application submits responses for ordering

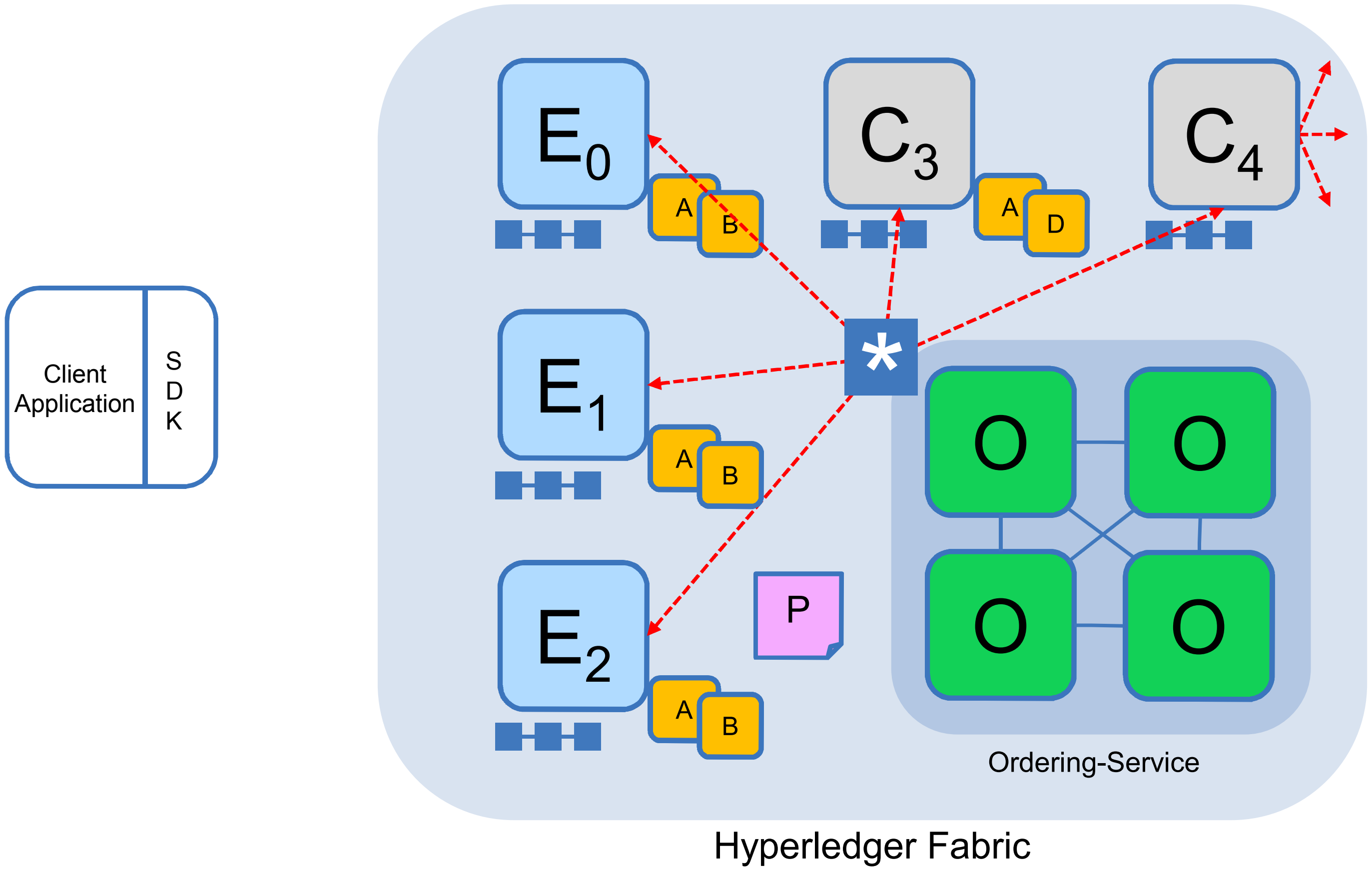
Application submits responses as a transaction to be ordered.

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

Key:

Endorser	🟦	■━■━■	Ledger
Committing Peer	⬜	□	Application
Ordering Node	🟩		
Smart Contract (Chaincode)	🟧	🟪	Endorsement Policy

Sample transaction: Step 5/7 – Deliver Transaction



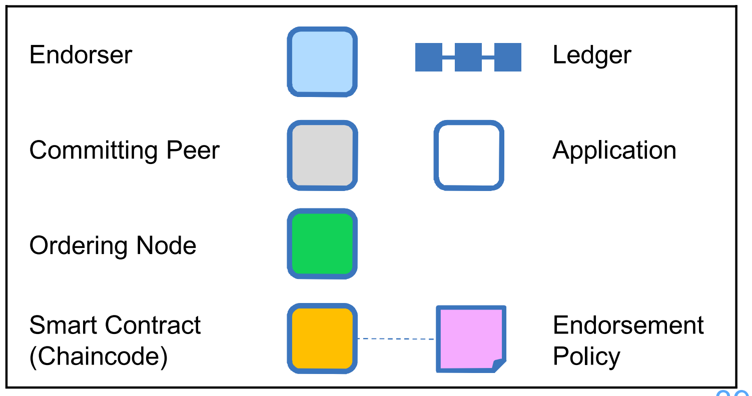
Orderer delivers to all committing peers

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

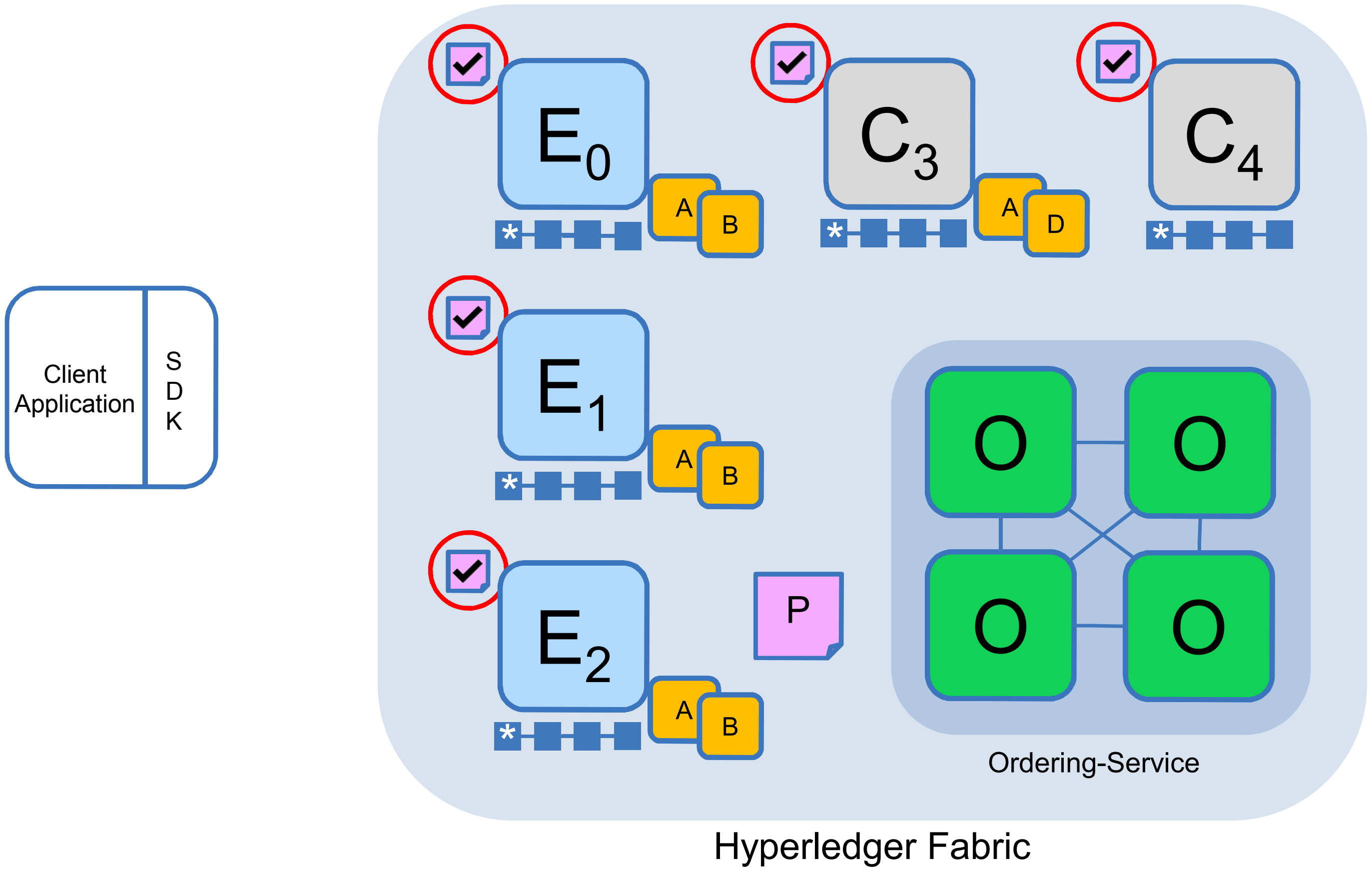
Different ordering algorithms available:

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

Key:



Sample transaction: Step 6/7 – Validate Transaction



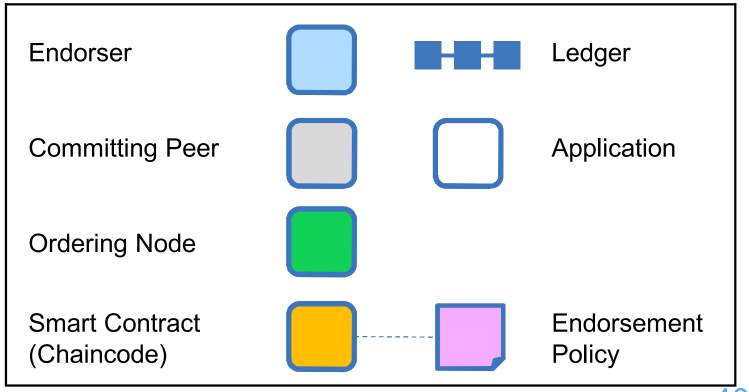
Committing peers validate transactions

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

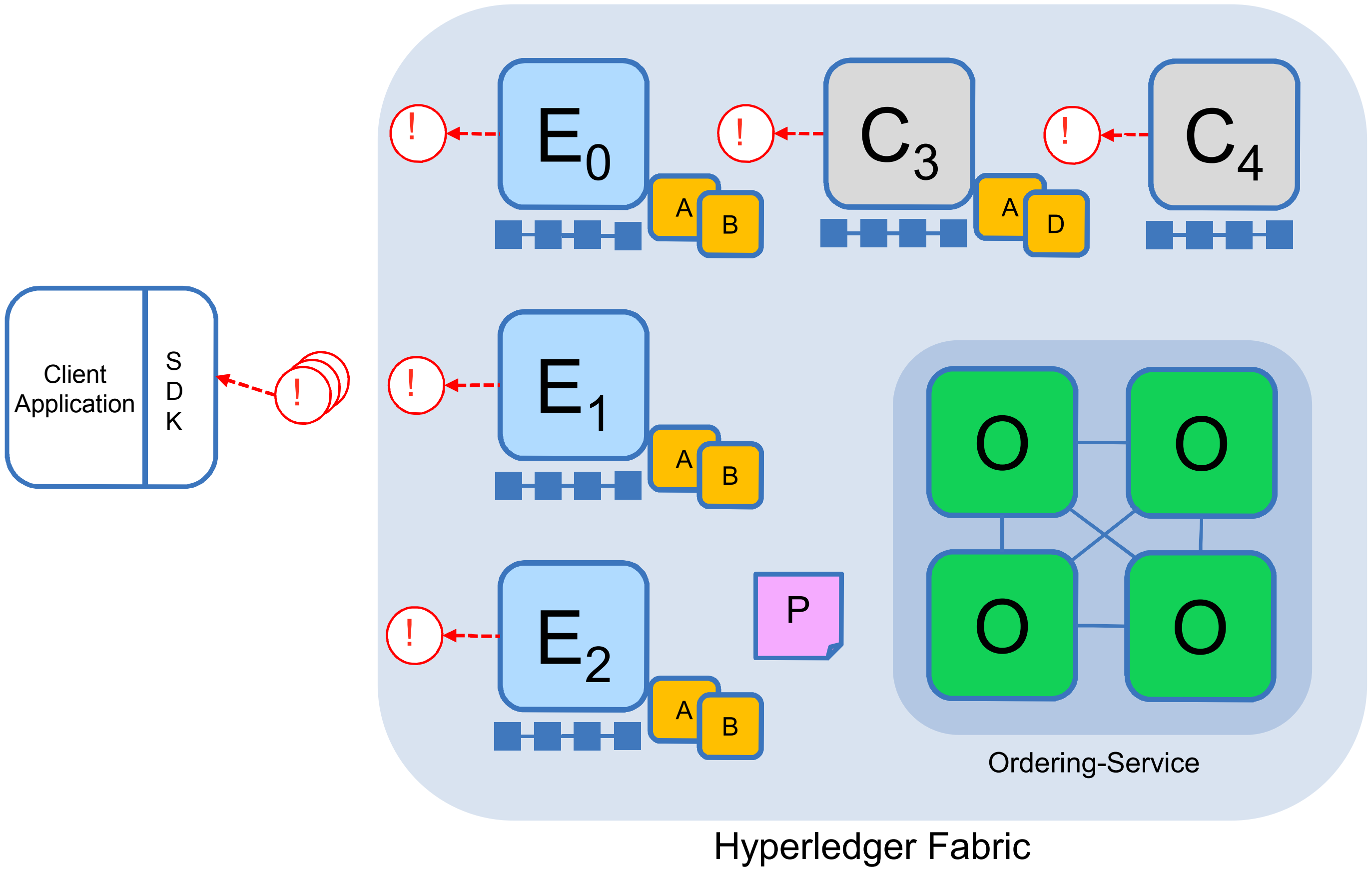
Validated transactions are applied to the world state and retained on the ledger

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

Key:



Sample transaction: Step 7/7 – Notify Transaction

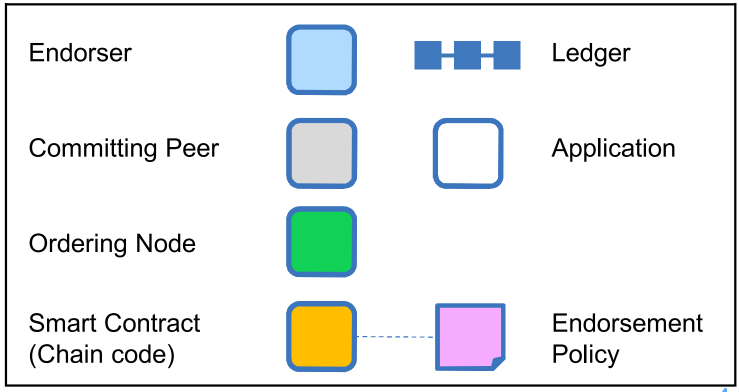


Committing peers notify applications

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:

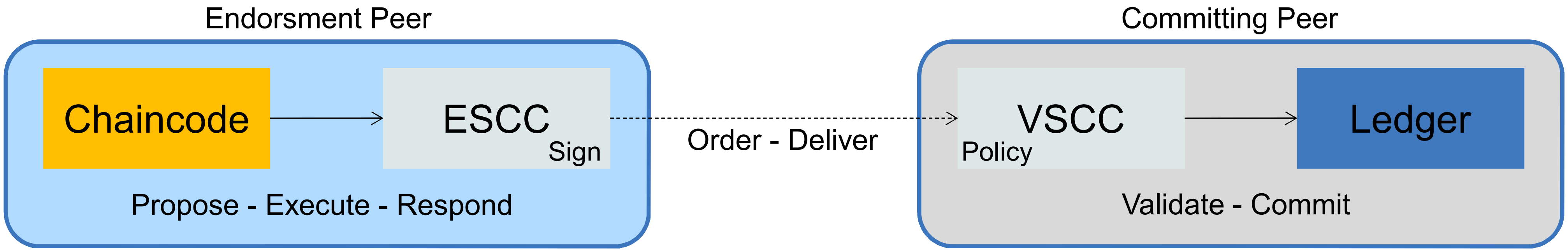


Endorsement Policies

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 associated with an Endorsement Policy
- Default implementation: Simple declarative language for the policy
- ESCC (Endorsement System ChainCode) signs the proposal response on the endorsing peer
- VSCC (Validation System ChainCode) validates the endorsements



Endorsment Peer

Chaincode

ESCC

Sign

Propose - Execute - Respond

Order - Deliver

Committing Peer

VSCC

Policy

Ledger

Validate - Commit

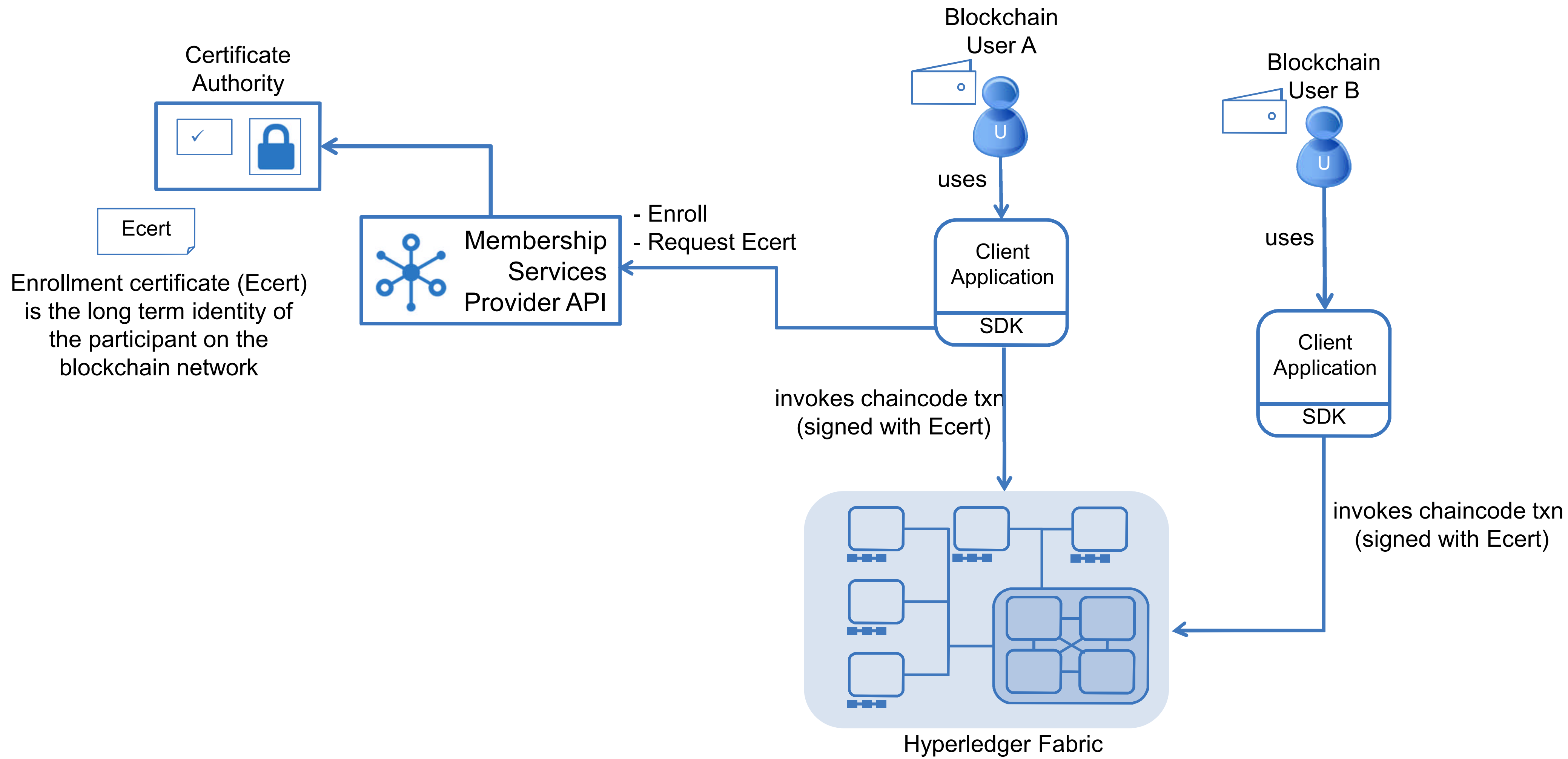
Endorsement Policy Examples

Examples of policies:

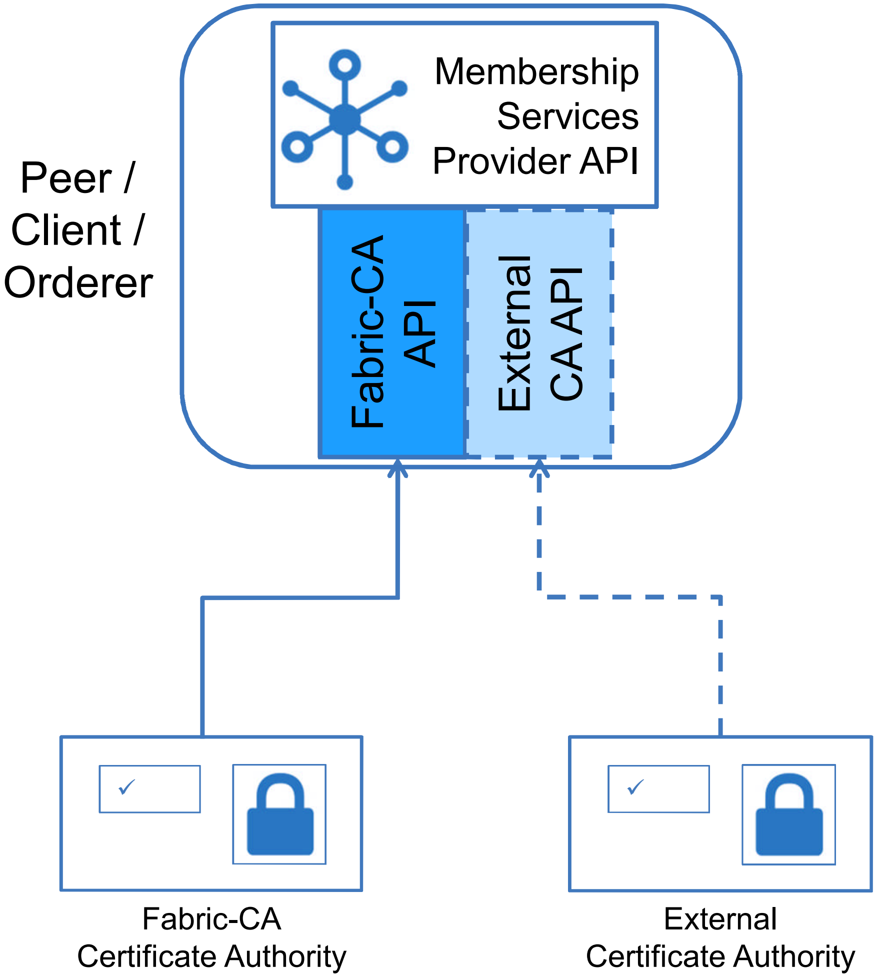
- **Request 1 signature from all three principals**
  - AND('Org1.member', 'Org2.member', 'Org3.member')
- **Request 1 signature from either one of the two principals**
  - OR('Org1.member', 'Org2.member')
- **Request either one signature from a member of the Org1 MSP or (1 signature from a member of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

Permissioned Ledger Access

Membership Services Overview


Certificate Authority
Ecert
Enrollment certificate (Ecert) is the long term identity of the participant on the blockchain network
Membership Services Provider API
- Enroll
- Request Ecert
Blockchain User A
U
uses
Client Application
SDK
invokes chaincode txn (signed with Ecert)
Blockchain User B
U
uses
Client Application
SDK
invokes chaincode txn (signed with Ecert)
Hyperledger Fabric

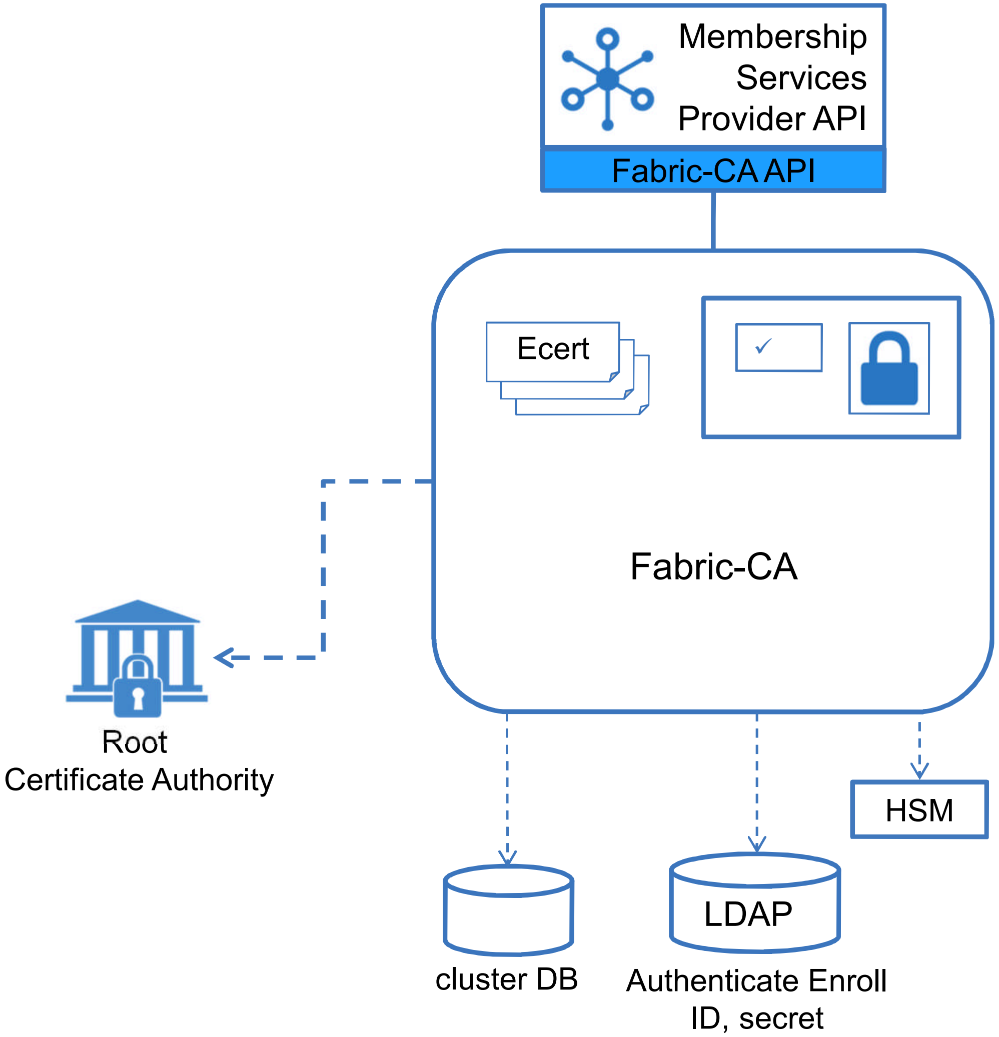
Membership Services Provider API



Membership Services Provider API

- Pluggable interface supporting a range of credential architectures
- Default implementation calls Fabric-CA.
- Governs identity for Peers and Users.
- Provides:
  - User authentication
  - User credential validation
  - Signature generation and verification
  - Optional credential issuance
- Additional offline enrollment options possible (eg File System).

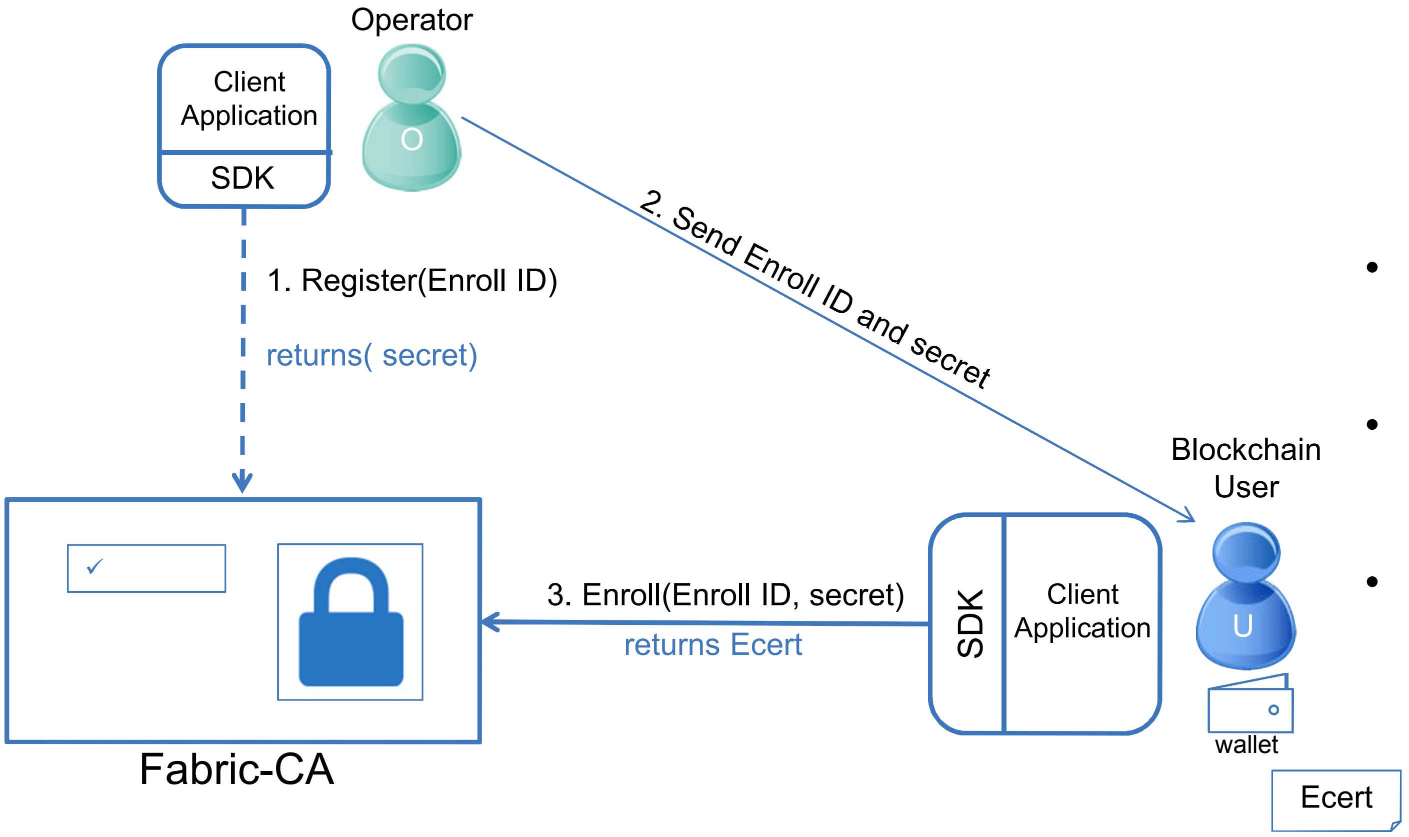
Fabric-CA Details


Membership Services Provider API
Fabric-CA API
Ecert
✓
Fabric-CA
Root Certificate Authority
HSM
cluster DB
LDAP
Authenticate Enroll ID, secret

Fabric-CA

- Default implementation of the Membership Services Provider Interface.
- Issues Ecerts (long-term identity)
- Supports clustering for HA characteristics
- Supports LDAP for user authentication
- Supports HSM

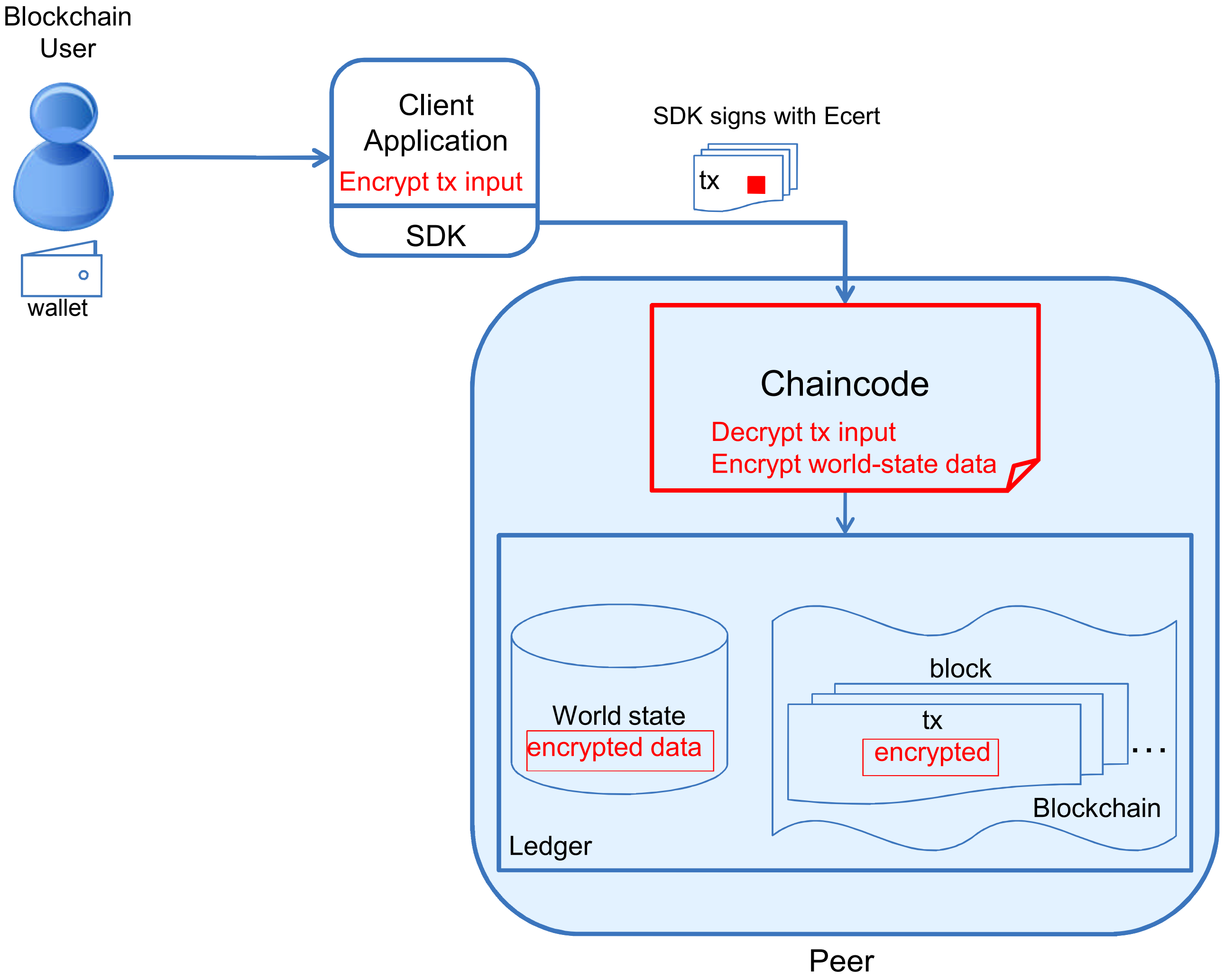
New User Registration and Enrollment



Registration and Enrollment

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

Application Level Encryption



Data Encryption

Handled in the application domain.

Multiple options for encrypting:

- Transaction Data
- Chaincode*
- World-State data

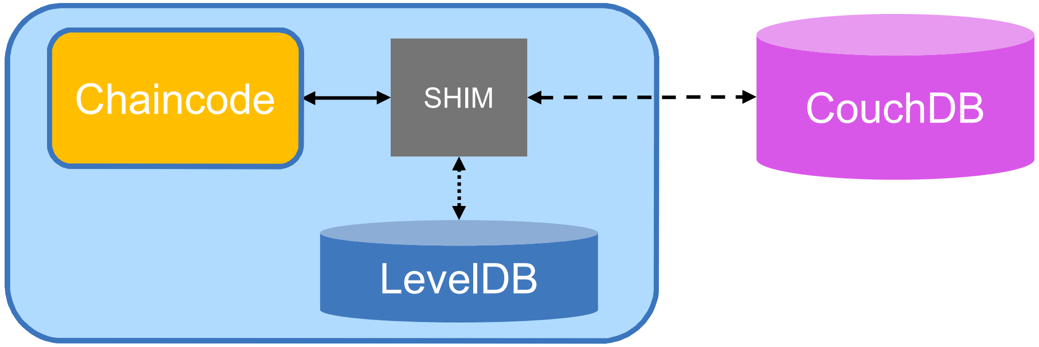
Chaincode optionally deployed with cryptographic material, or receive it in the transaction from the client application using the transient data field (not stored on the ledger).

*Encryption of application chaincode requires additional development of system chaincode.

Pluggable World State

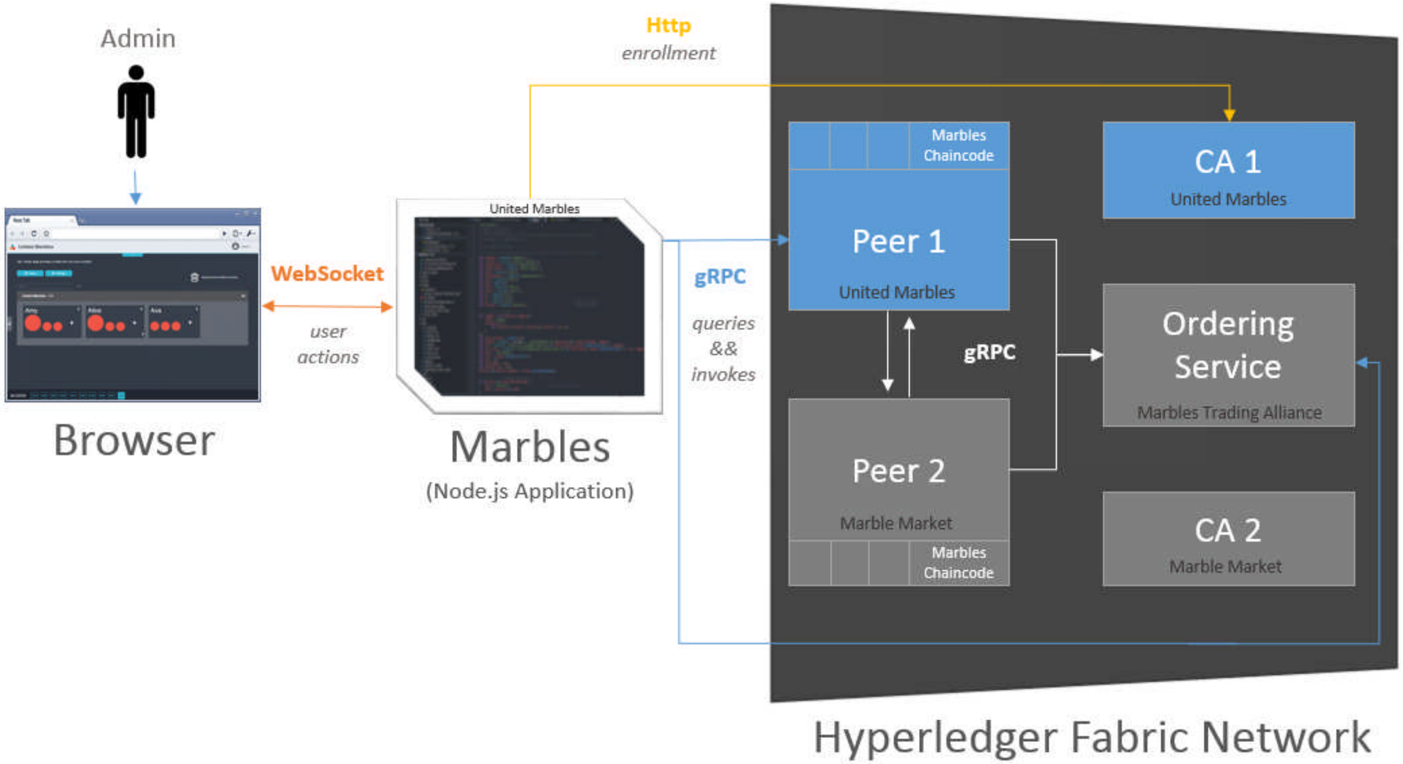
WorldState Database

- Pluggable worldstate database
- Default embedded key/value implementation using LevelDB
  - Support for keyed queries, but cannot query on value
- Support for Apache CouchDB
  - Full query support on key and value (JSON documents)
  - Meets a large range of chaincode, auditing, and reporting requirements
  - Will support reporting and analytics via data replication to an analytics engine such as Spark (future)
  - Id/document data model compatible with existing chaincode key/value programming model



Marbles Application

https://github.com/IBM-Blockchain/marbles



Marbles Application

- The underlying network for this application is the Hyperledger Fabric
- **The application is to aid a developer learn the basics of chaincode and app development with a Fabric network.**
- It is a simple asset transfer application. Multiple users can create and transfer marbles with each other.

Marbles Application

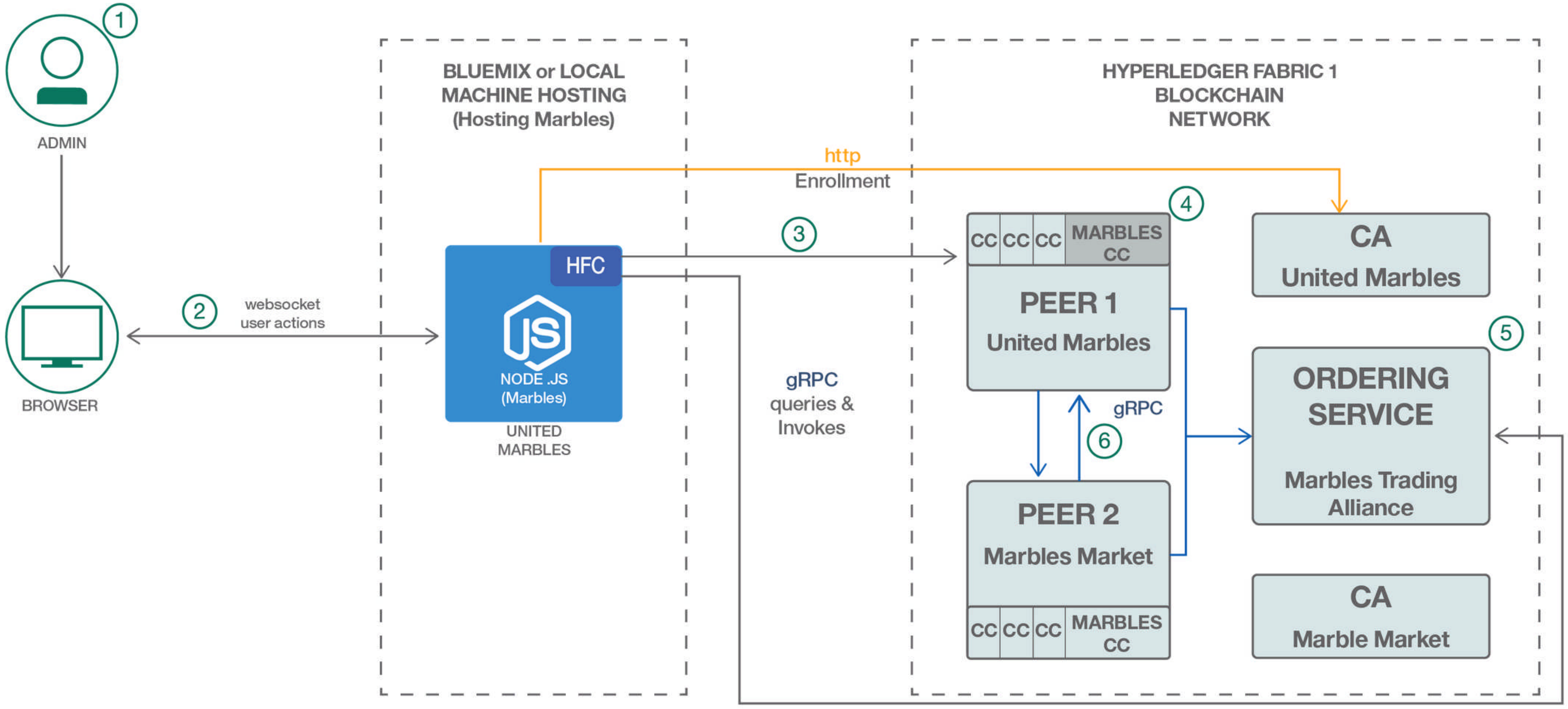
There are 3 distinct parts/worlds that you need to keep straight.

1. **The Chaincode Part** - This is GoLang code that runs on/with a peer on your blockchain network. Also, called cc. All marbles/blockchain interactions ultimately happen here. These files live in /chaincode.

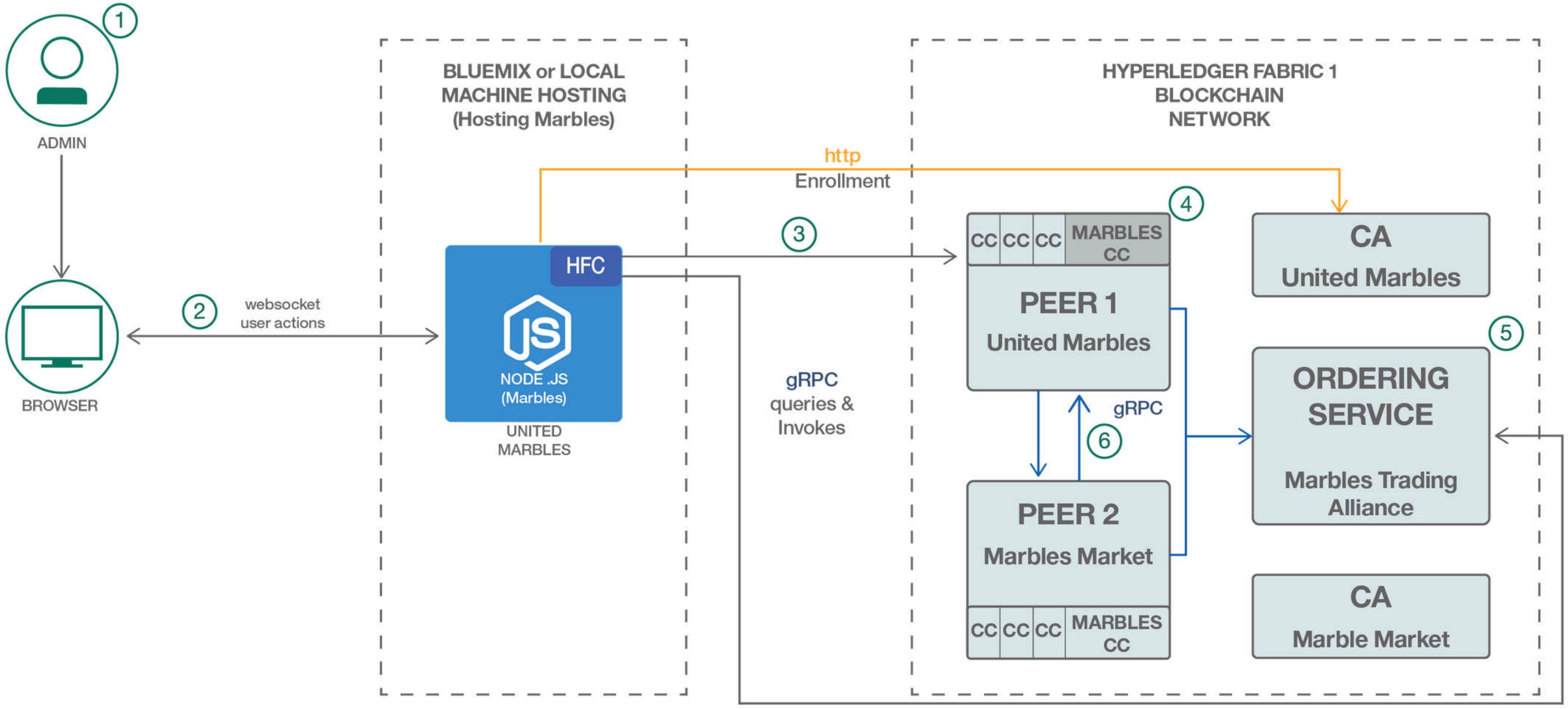
2. **The Client Side JS Part** - This is JavaScript code running in the user's browser. User interface interaction happens here. These files live in /public/js.

3. **The Server Side JS Part** - This is JavaScript code running our application's backend. ie Node.js code which is the heart of Marbles! Sometimes referred to as our node or server code. Functions as the glue between the marble admin and our blockchain. These files live in /utils and /routes.

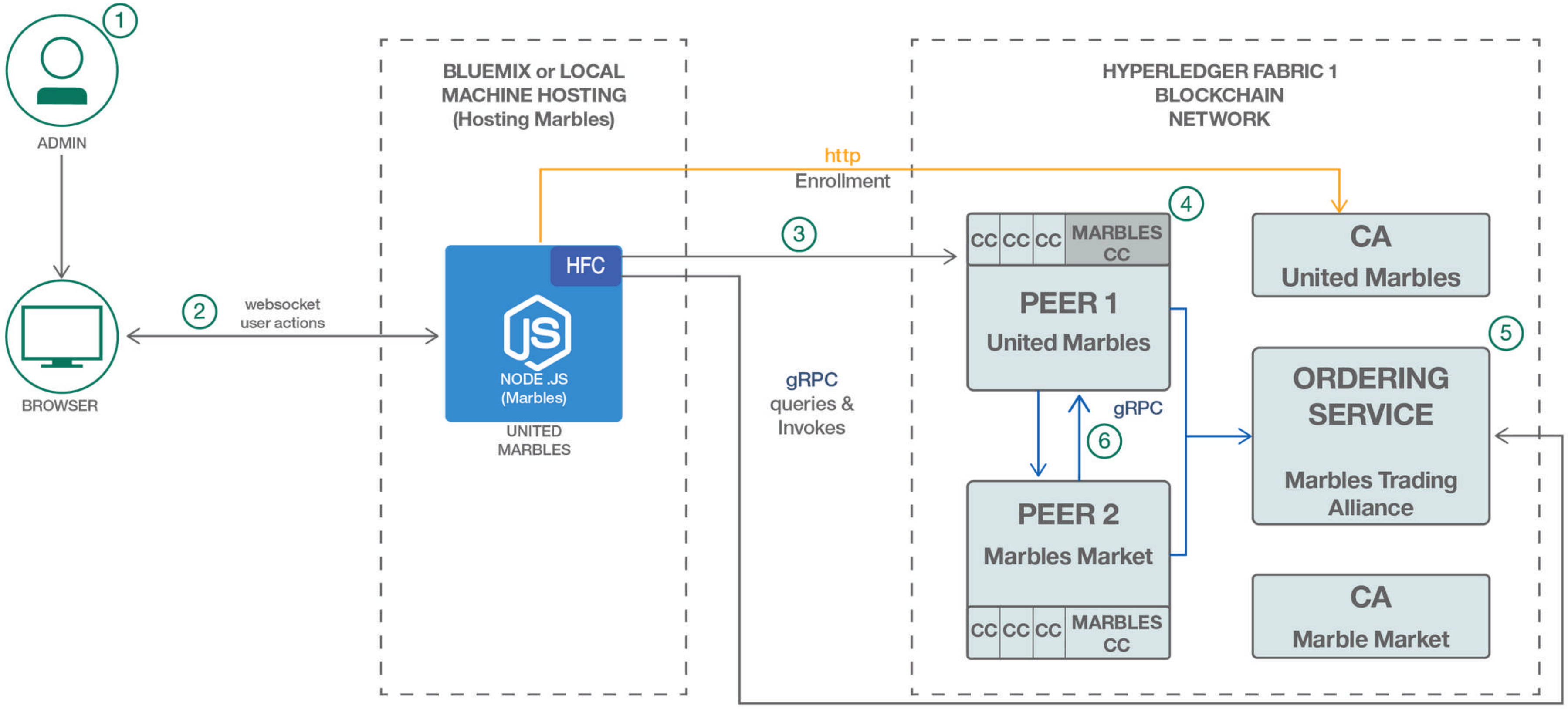
1 The administrator interacts with Marbles, our Node.js application, through a browser.



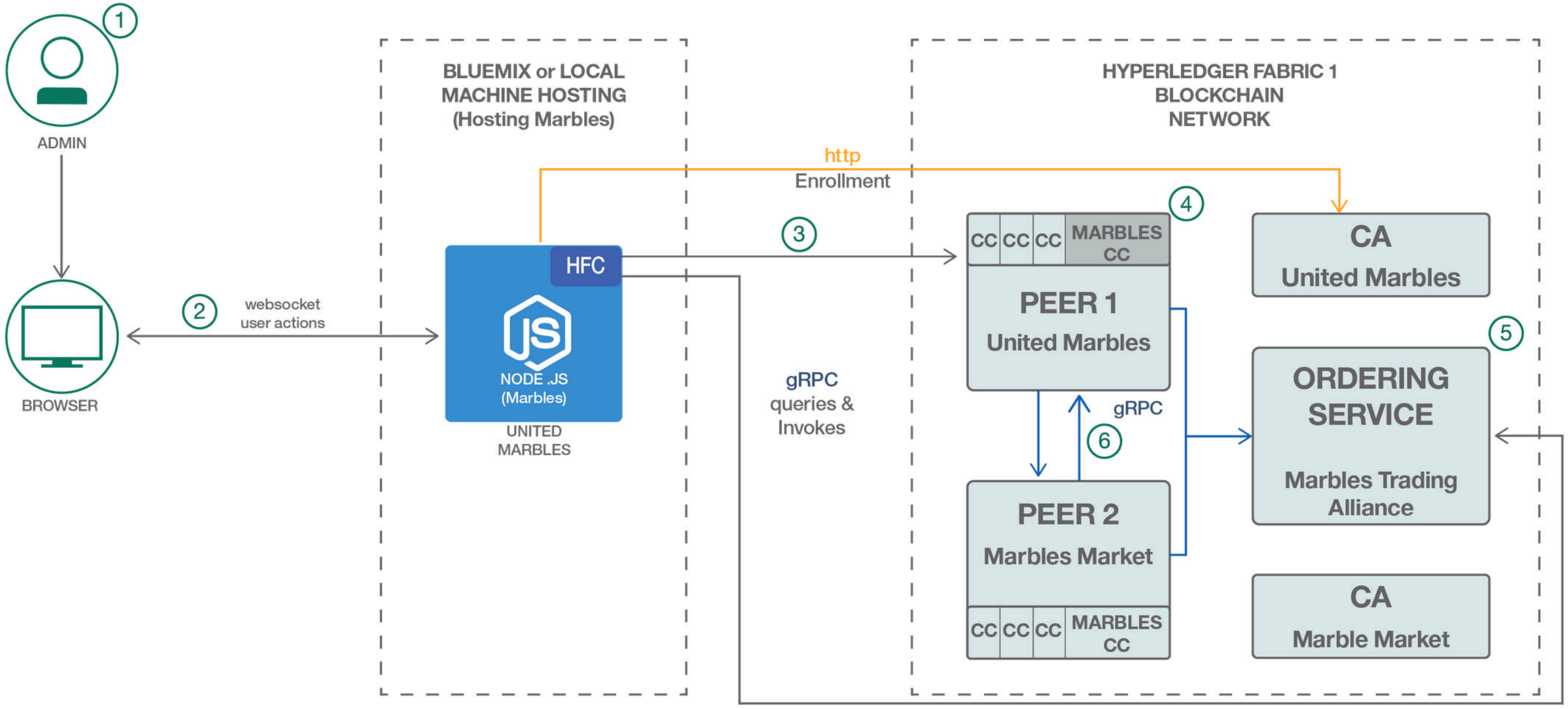
2 The client side JavaScript code opens a websocket to the backend Node.js application and instructions are sent to the application from the browser.



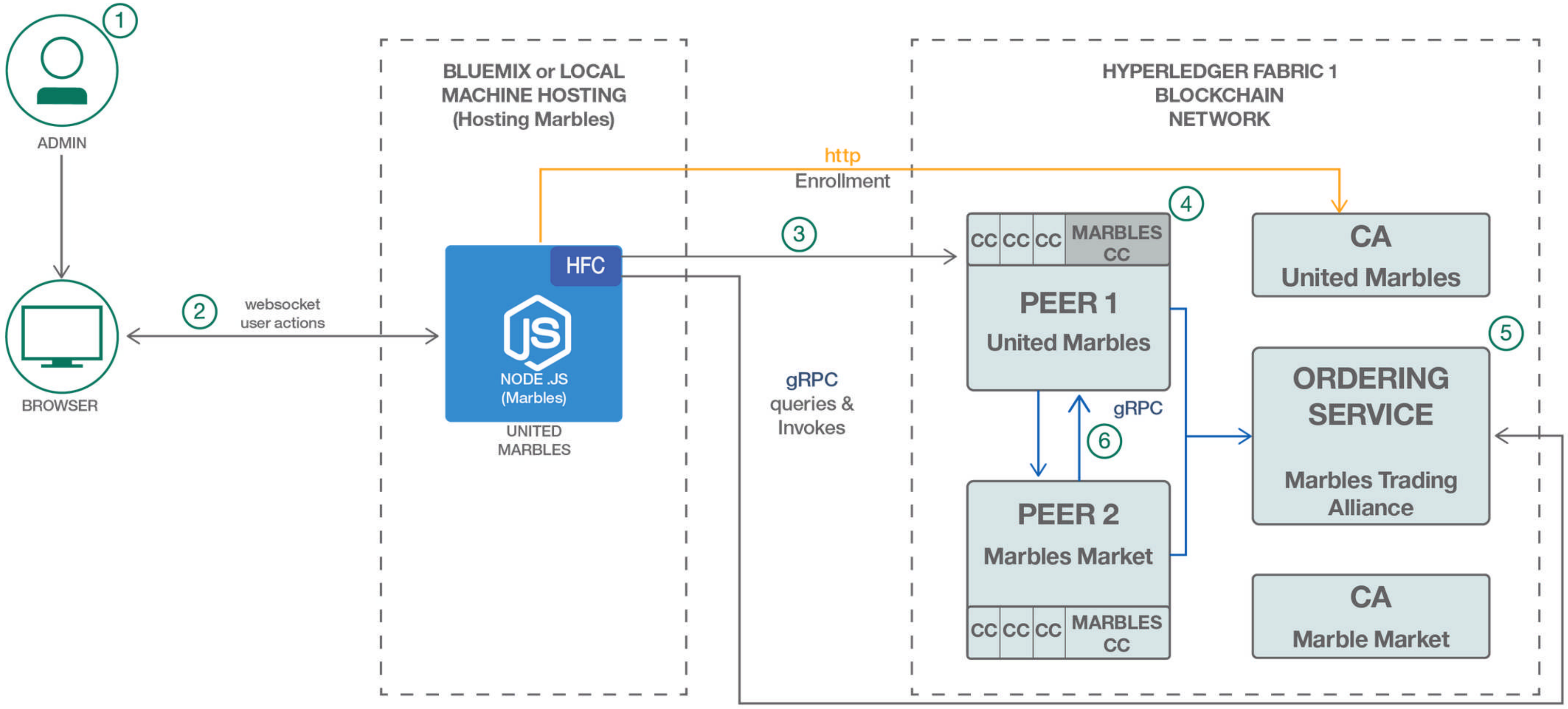
3 The proposal accesses the ledger to stimulate a transaction. This proposal is built by Marbles (using the SDK) and then sent to a blockchain peer.



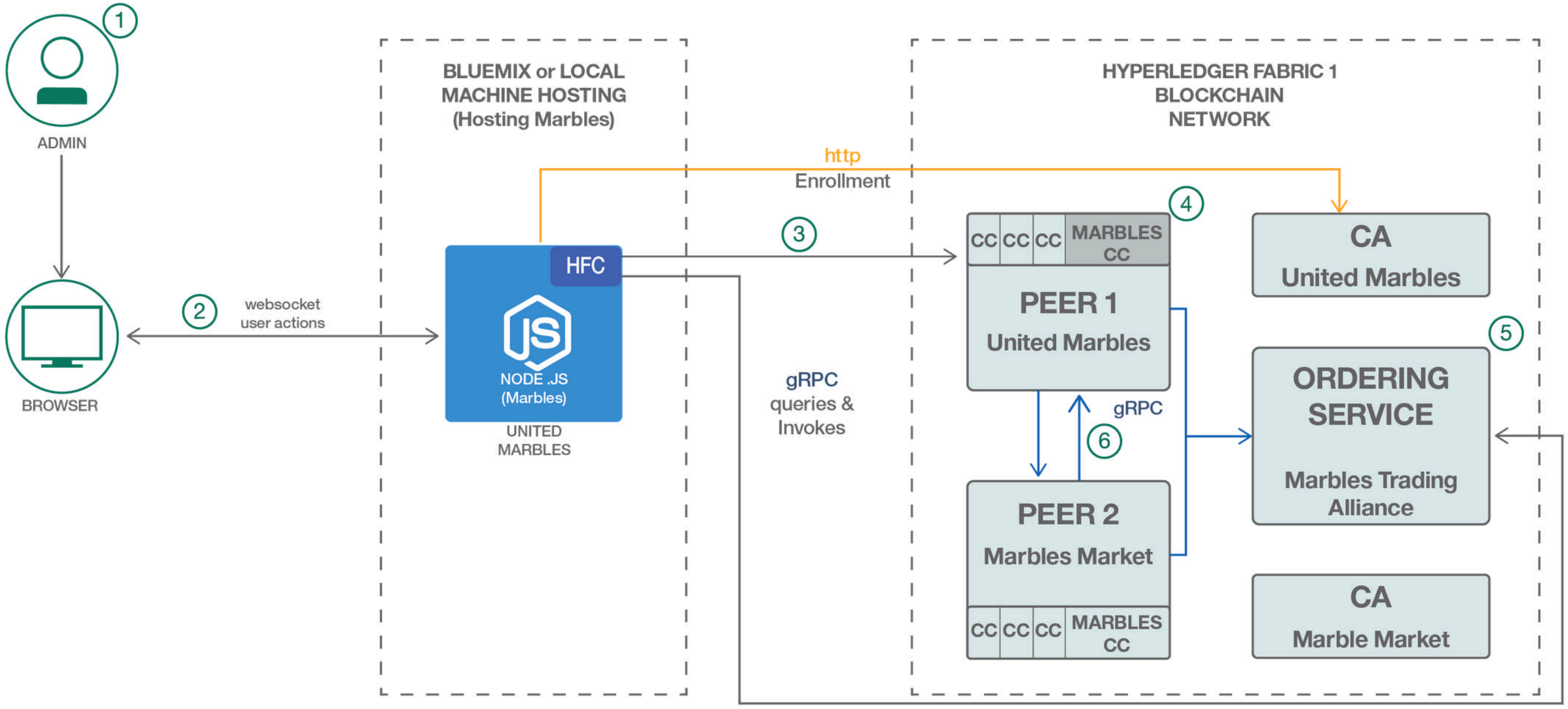
4 The endorser (process on the peer) will endorse (or sign) the transaction if there are no issue.



5 The SDK collects all the signed proposals and, if the policy is fulfilled, sends the transaction with the signed endorsements to the ordering service. The orderer service orders the transactions, creates a block, and delivers it to the appropriate peers..



6 The peer validates the block and writes it to its ledger. The transaction has now taken effect and any subsequent reads will reflect this change.



Marbles Demo
with
IBM BLOCKCHAIN

Summary and Next Steps

- Apply shared ledgers and smart contracts to your Business Network
- Spend time thinking about realistic business use cases
- Get some hands-on experience with the technology
- Start with a First Project
- IBM can help with your journey

THANK YOU