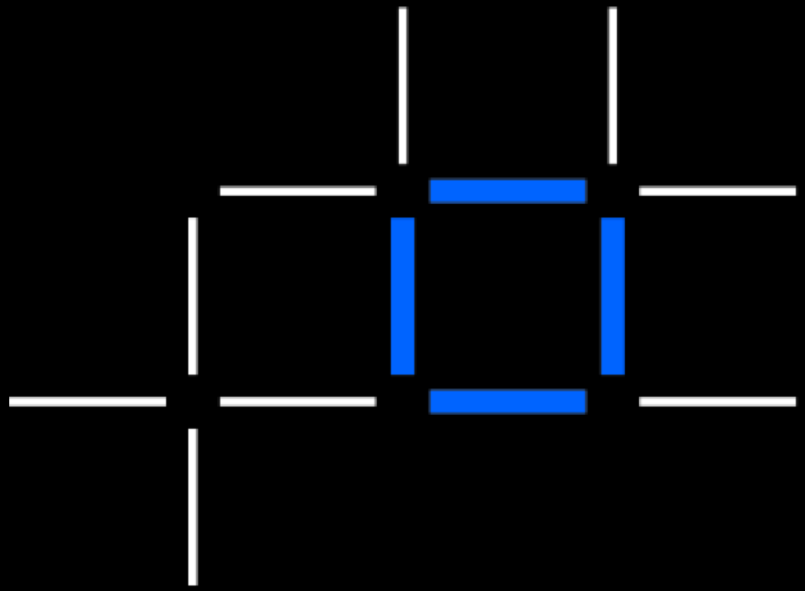
Blockchain Explored, Deep Dive Part 1

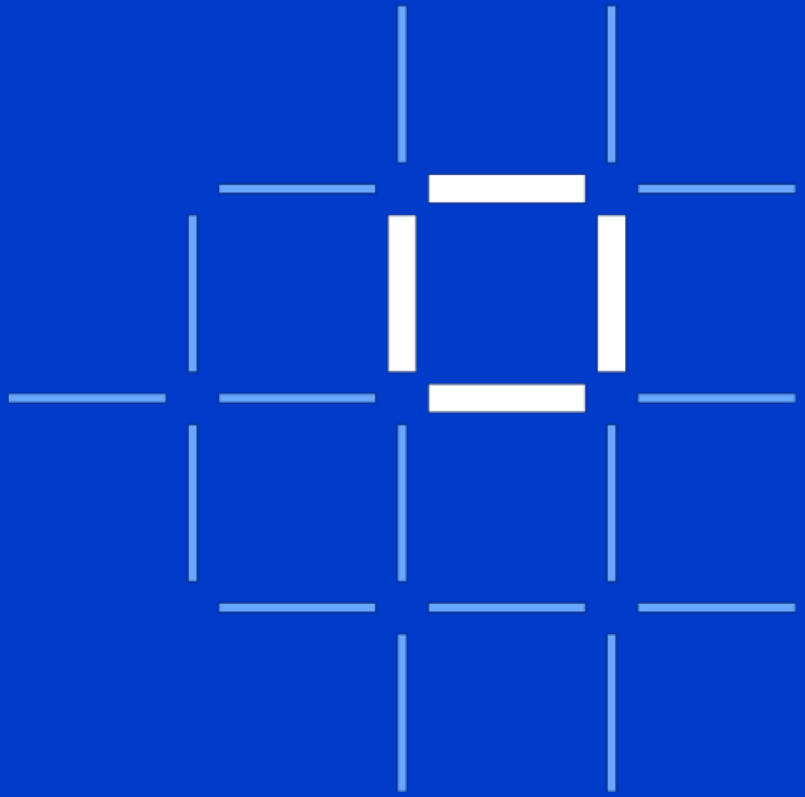
A Technical Deep-Dive on Hyperledger Fabric



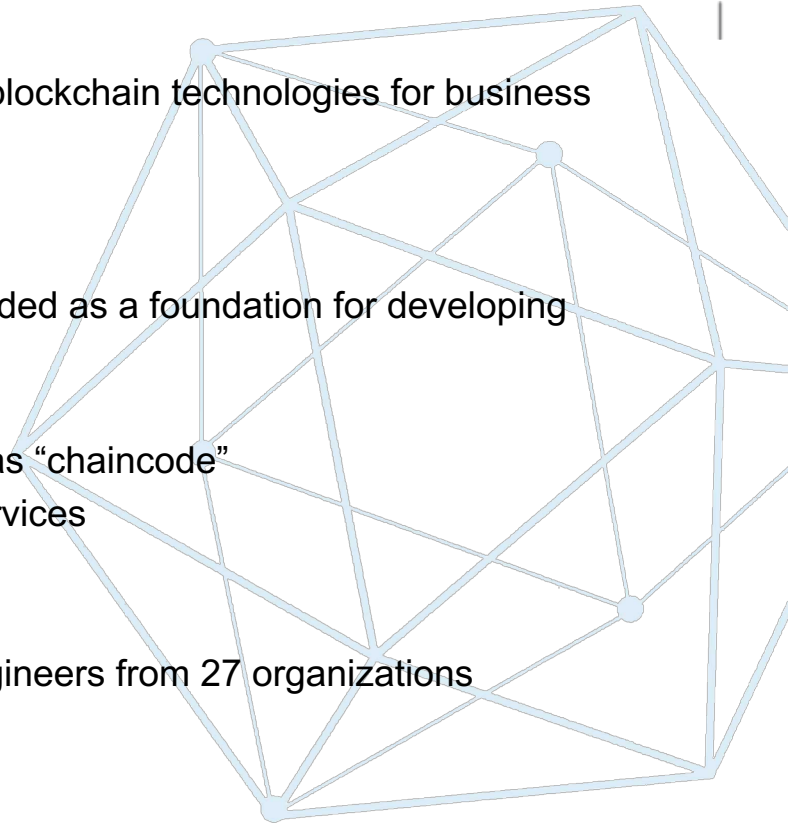
Barry Silliman
Blockchain Enablement on IBM Z and LinuxONE
IBM Washington Systems Center
silliman@us.ibm.com

Project Status

Technical Deep Dive



What is Hyperledger Fabric



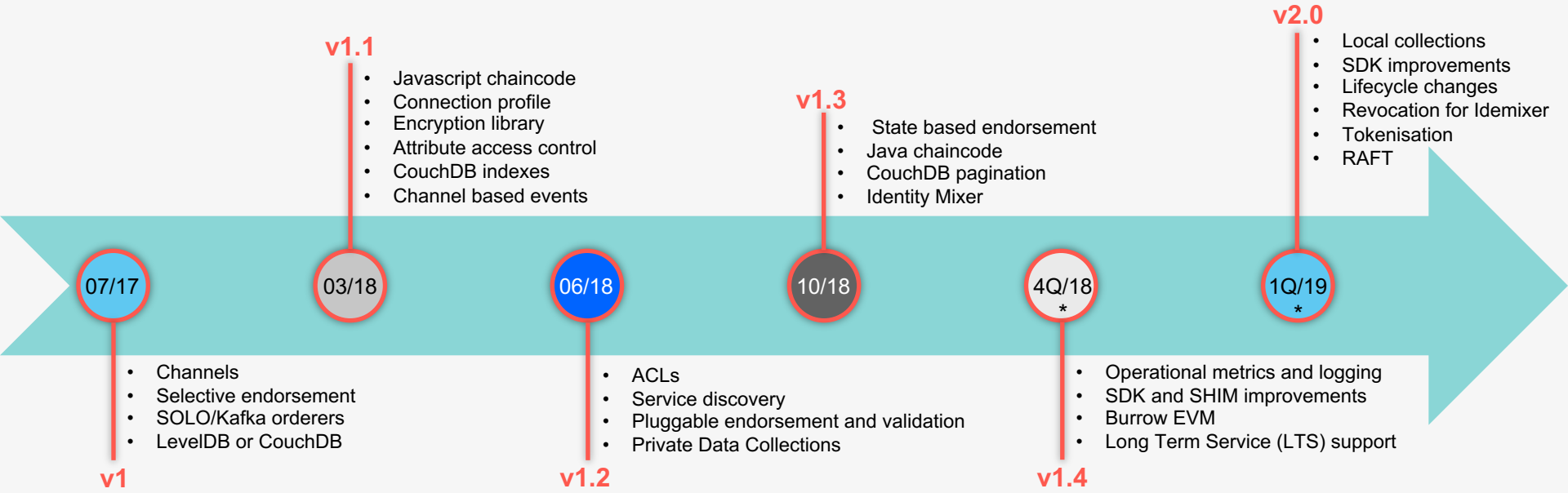
- A project within the Hyperledger open source collaboration
  - A collaborative effort created to advance cross-industry blockchain technologies for business
  - Hosted by the Linux Foundation

- 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

- First major version released July 2017: contributions by 159 engineers from 27 organizations
  - IBM is one of the contributors to Hyperledger Fabric

HYPERLEDGER FABRIC

Past, present and future

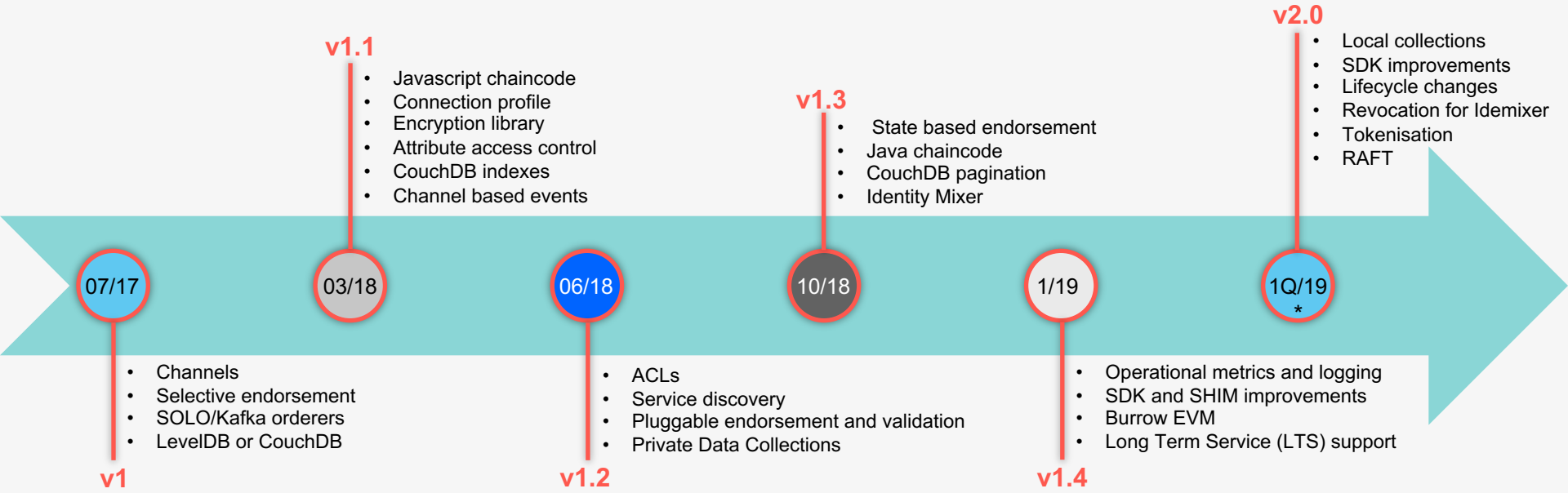


HYPERLEDGER
BLOCKCHAIN TECHNOLOGIES FOR BUSINESS

Based on https://wiki.hyperledger.org/projects/fabric/roadmap - Dates determined by the Hyperledger community - (*) Subject to change

HYPERLEDGER FABRIC

Past, present and future

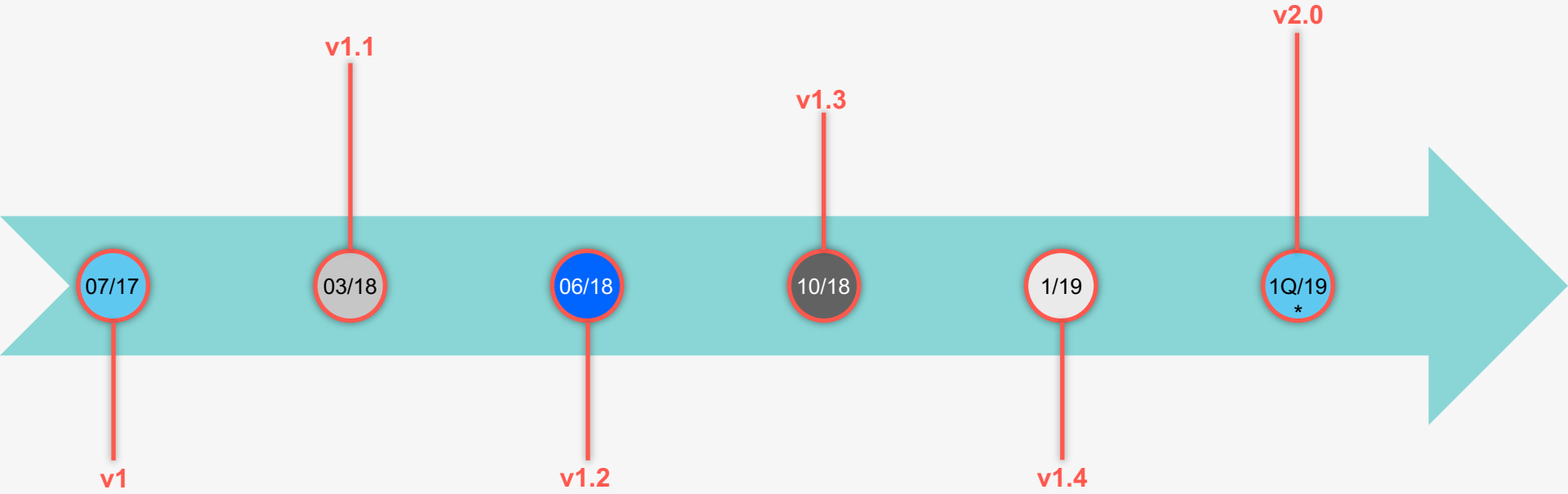


HYPERLEDGER
BLOCKCHAIN TECHNOLOGIES FOR BUSINESS

Based on https://wiki.hyperledger.org/projects/fabric/roadmap - Dates determined by the Hyperledger community - (*) Subject to change

HYPERLEDGER FABRIC

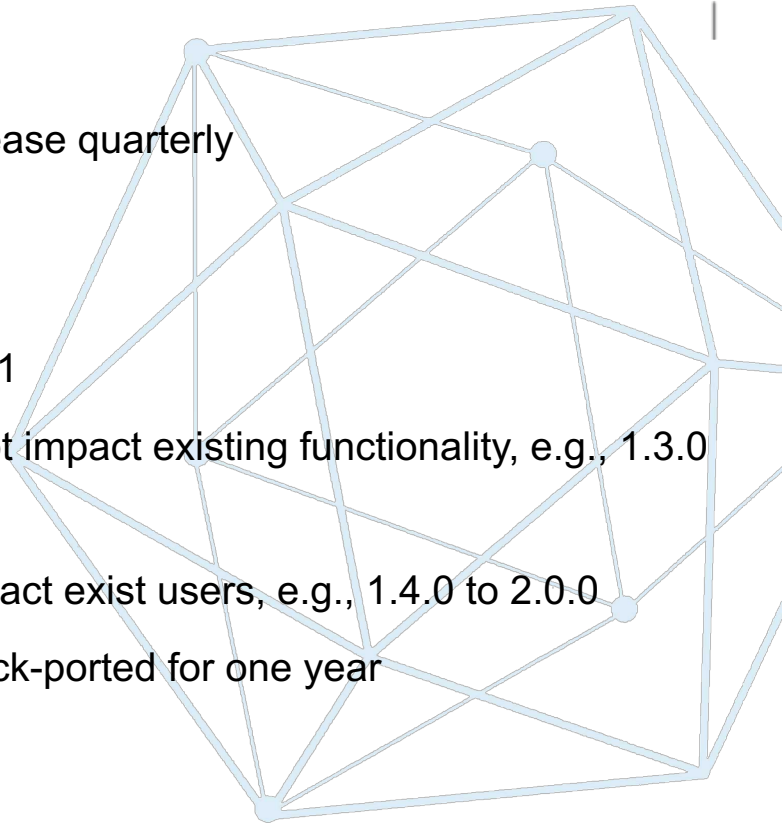
Past, present and future



HYPERLEDGER
BLOCKCHAIN TECHNOLOGIES FOR BUSINESS

Based on https://wiki.hyperledger.org/projects/fabric/roadmap - Dates determined by the Hyperledger community - (*) Subject to change

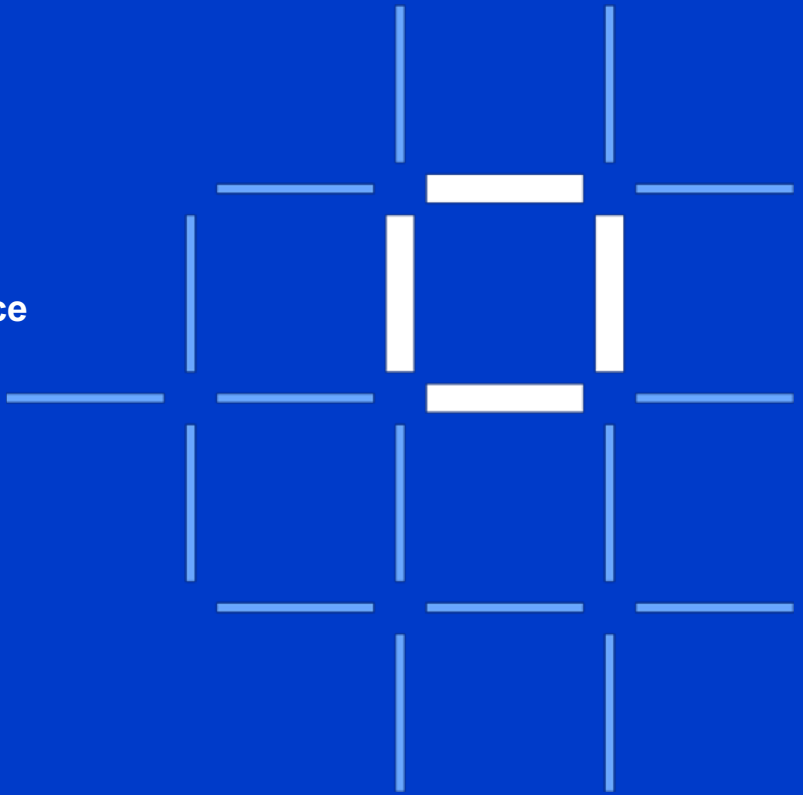
Release Cycle and Semantic Versioning



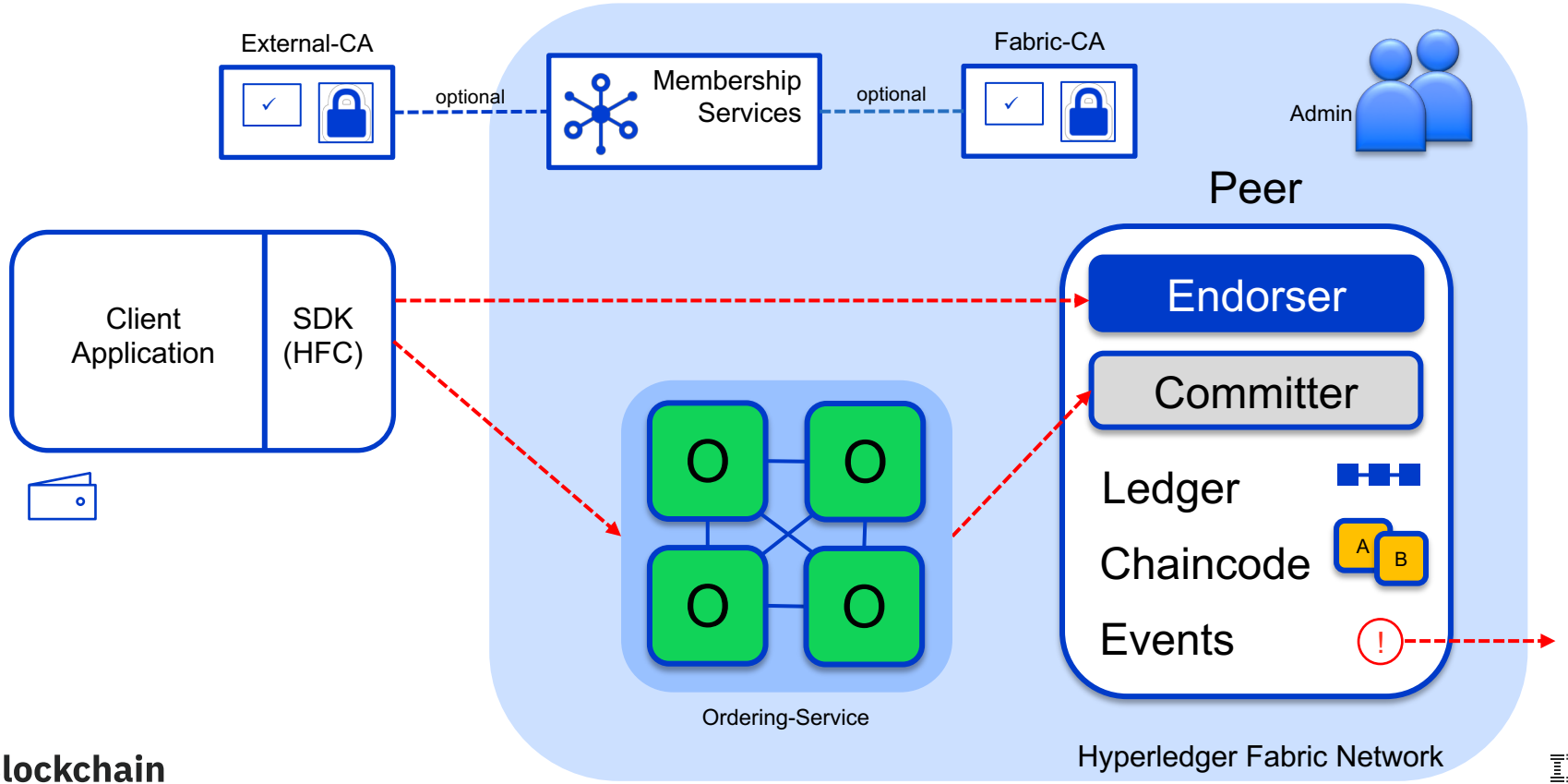
- Hyperledger Fabric is targeting a new major or minor release quarterly
- Semantic Versioning – https://semver.org
  - MAJOR.MINOR.PATCH, e.g., 1.4.1
- Increase patch number if just bug fixes, e.g., 1.4.0 to 1.4.1
- Increase minor number if new functionality that should not impact existing functionality, e.g., 1.3.0 to 1.4.0
- Increase major number if new functionality that could impact exist users, e.g., 1.4.0 to 2.0.0
- Long term support release (1.4.0 was the first) – fixes back-ported for one year

Technical Deep Dive

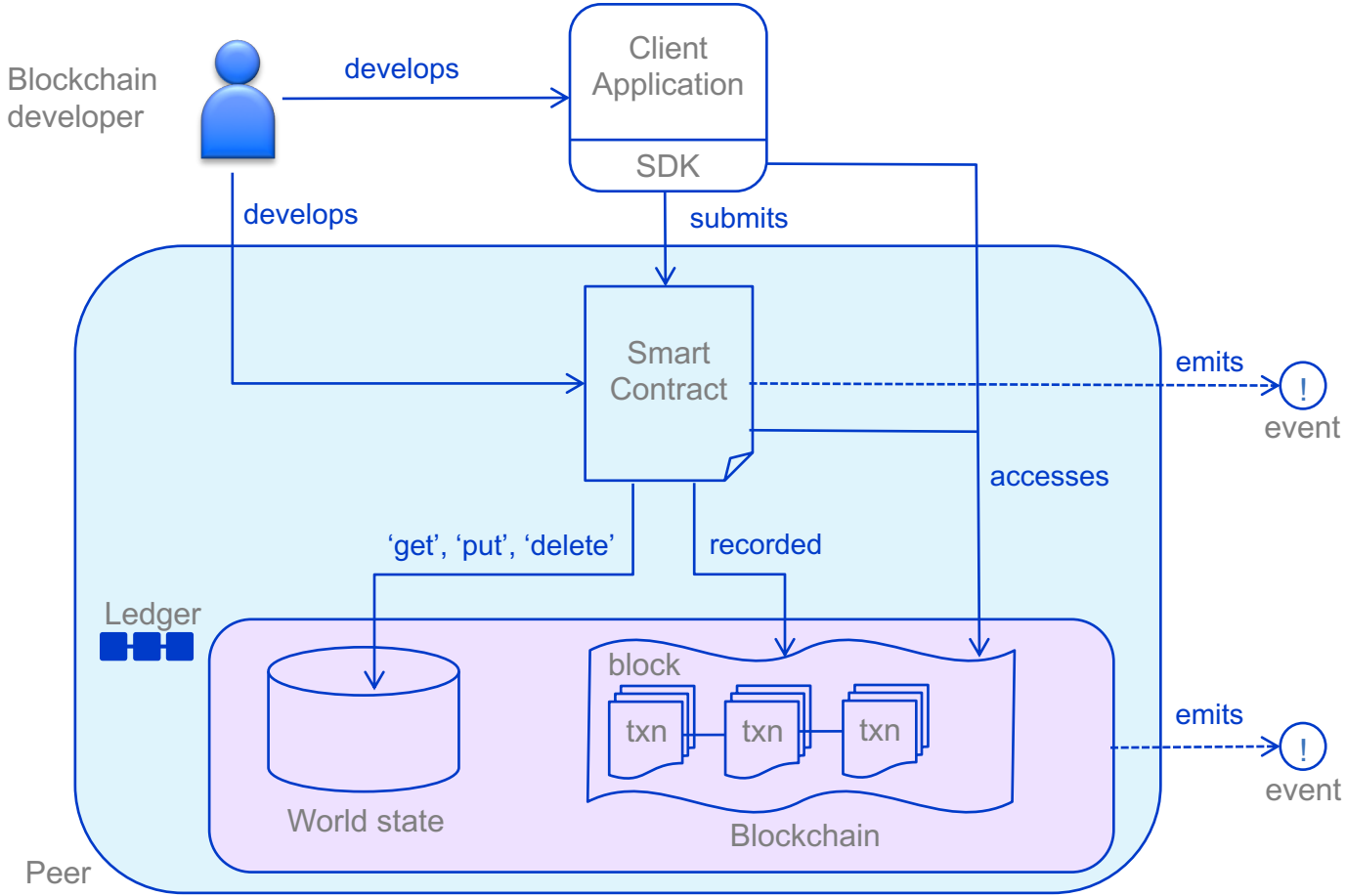
- **[Architectural Overview]**
- **Network Consensus**
- **Channels and Ordering Service**
- Components
- Network setup
- Endorsement Policies
- Membership Services



Hyperledger Fabric V1.x Architecture

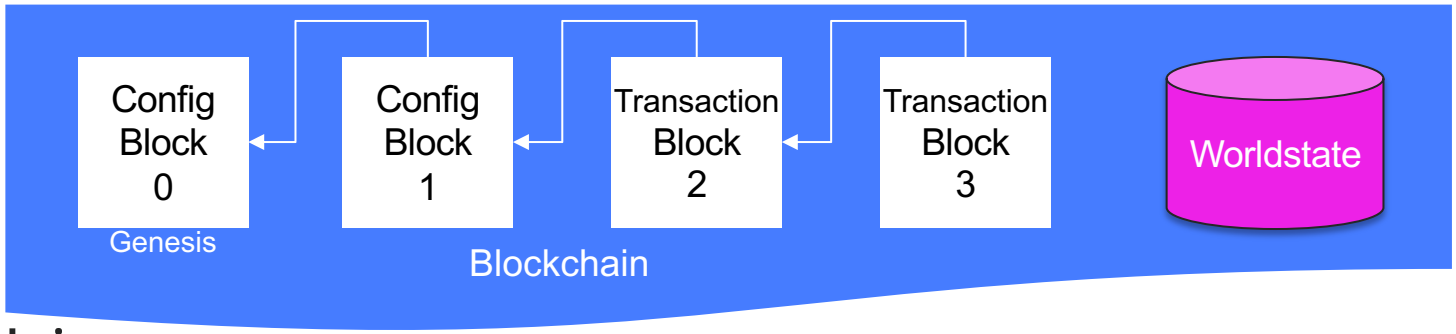


How applications interact with the ledger

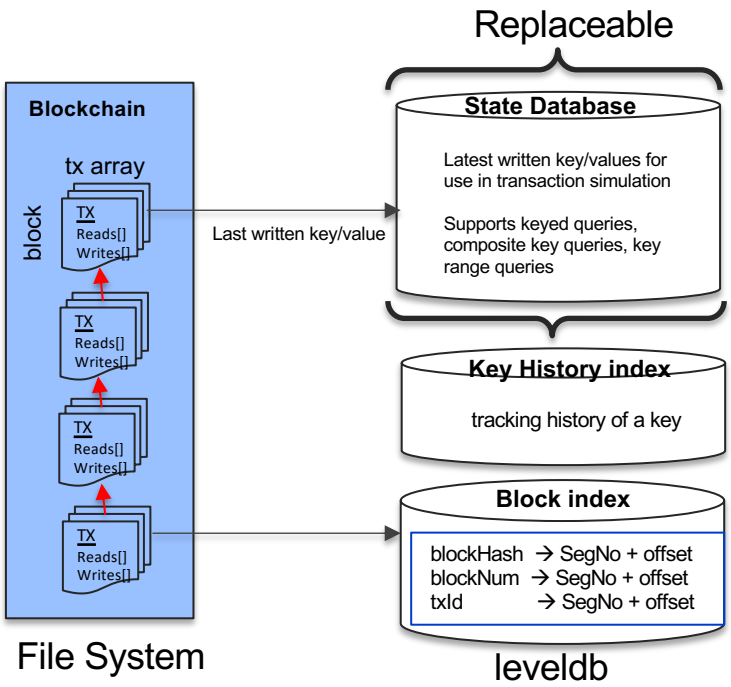


Fabric Ledger

- 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 Contract decides what is written to the worldstate



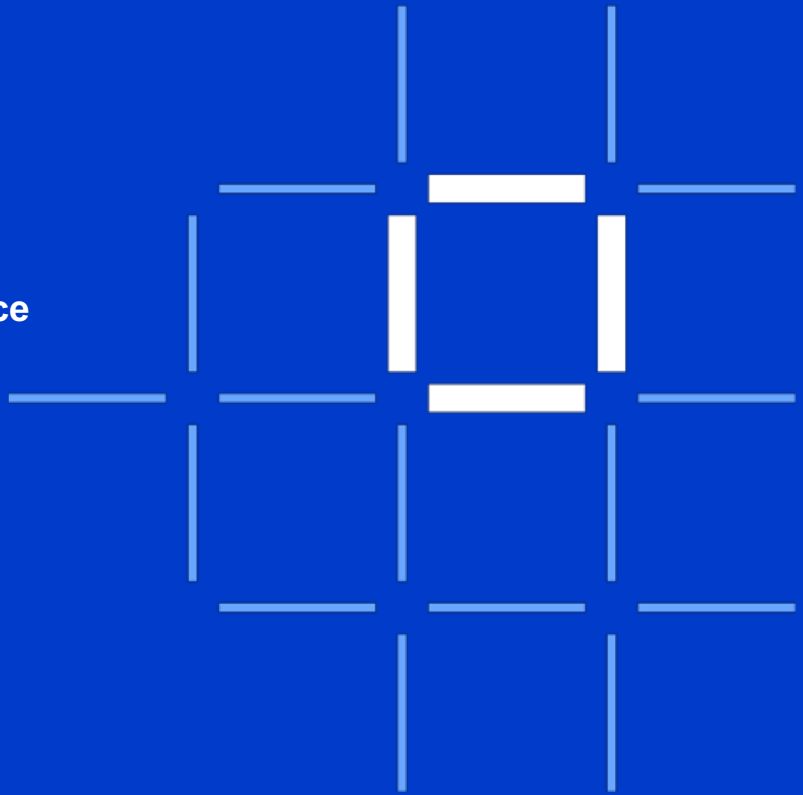
More ledger legerdemain



CouchDB (external option) adds rich data queries

Technical Deep Dive

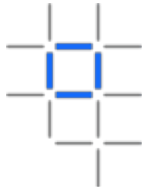
- **Architectural Overview**
- **[Network Consensus]**
- **Channels and Ordering Service**
- Components
- Network setup
- Endorsement Policies
- Membership Services



Nodes and roles

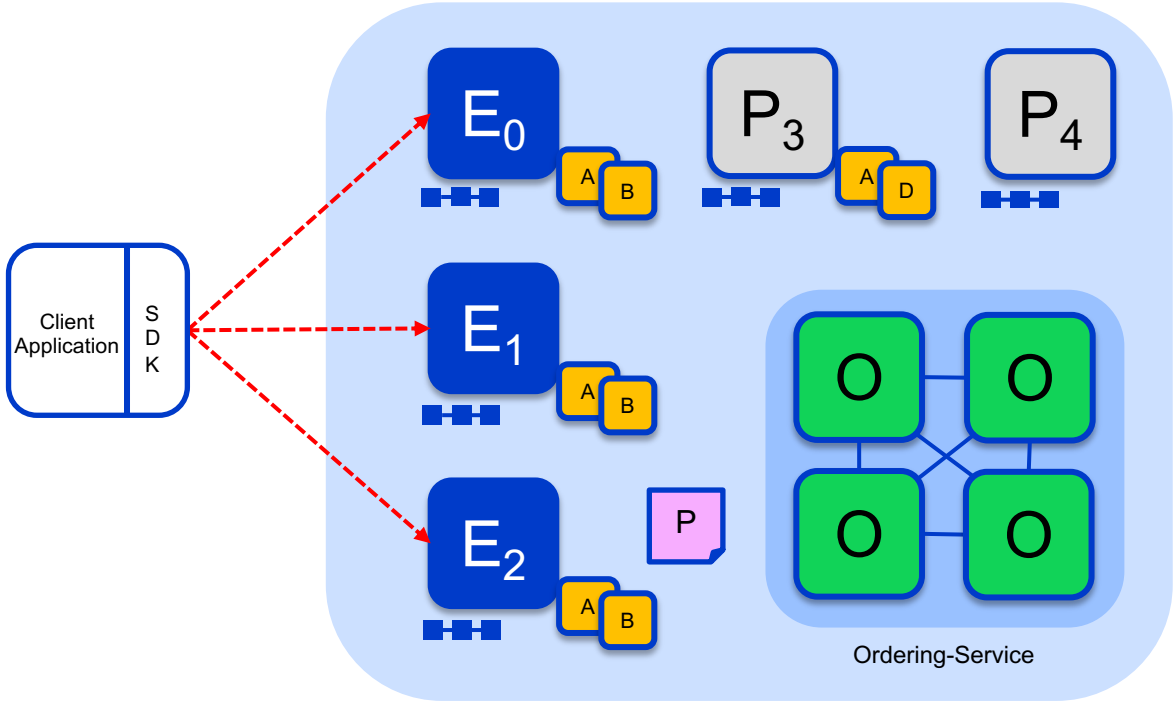
	**Peer:** Maintains ledger and state. Commits transactions. May hold smart contract (chaincode).
	**Endorsing Peer:** Specialized peer also endorses transactions by receiving a transaction proposal and responds by granting or denying endorsement. Must hold smart contract.
	**Ordering Node:** Approves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

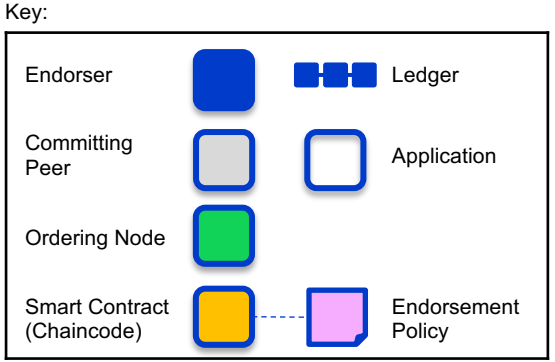


Hyperledger Fabric Network

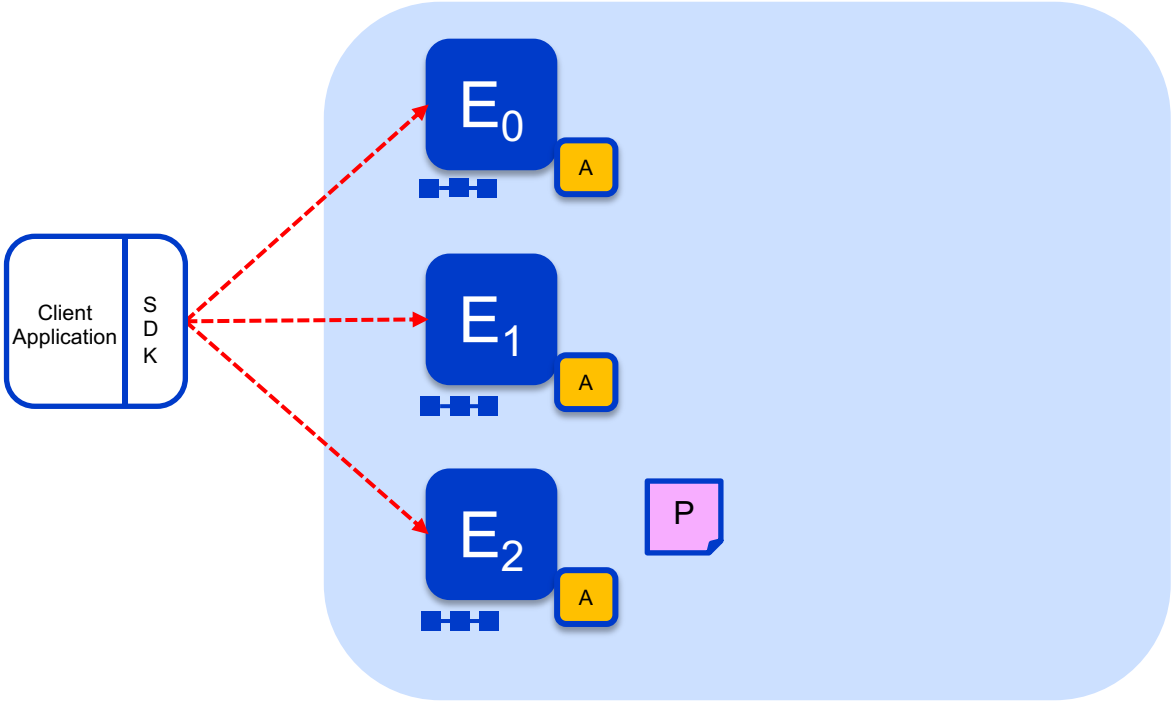
Application proposes transaction

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}



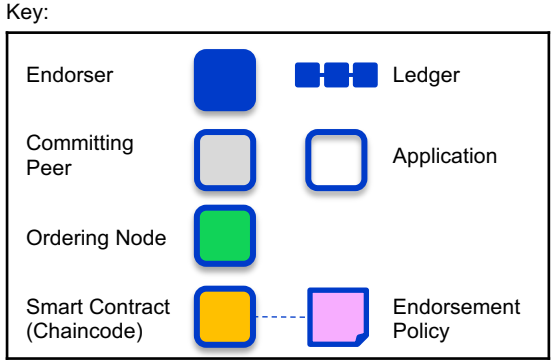
Sample transaction: Step 1/7 – Propose transaction



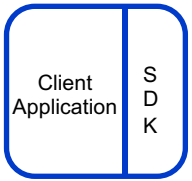
Application proposes transaction

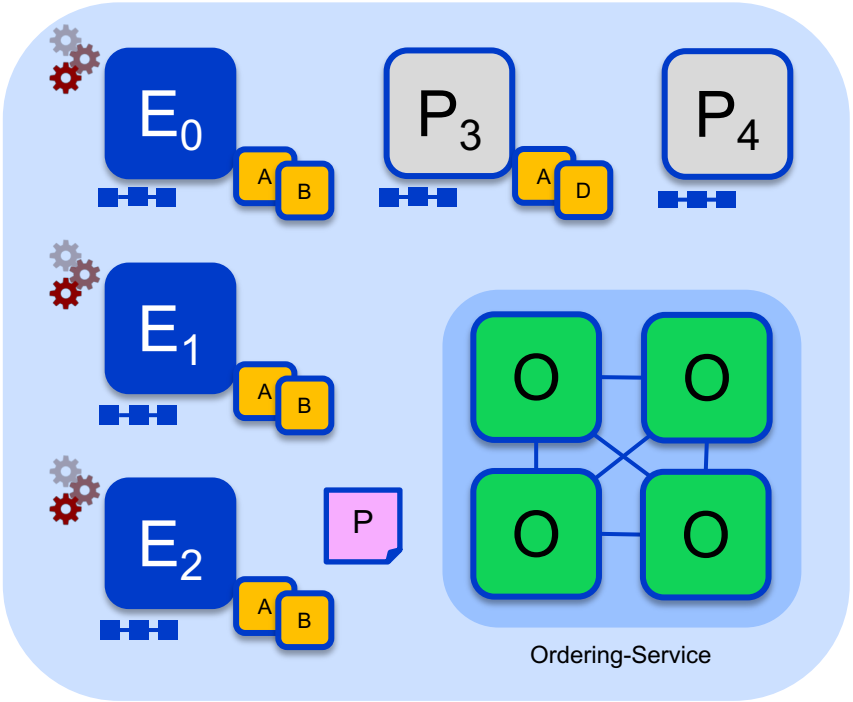
Endorsement policy:
- "E_0, E_1 and E_2 must sign"
- (P_3, P_4 are not part of the policy)

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



Sample transaction: Step 2/7 – Execute proposal





Hyperledger Fabric Network

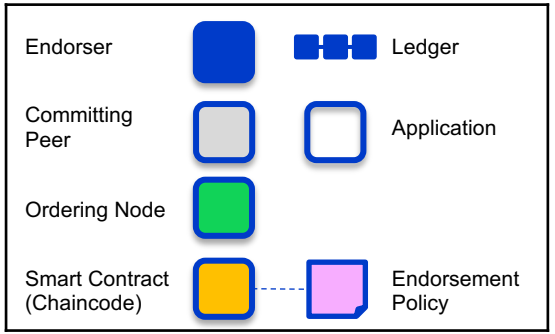
Endorsers Execute Proposals

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

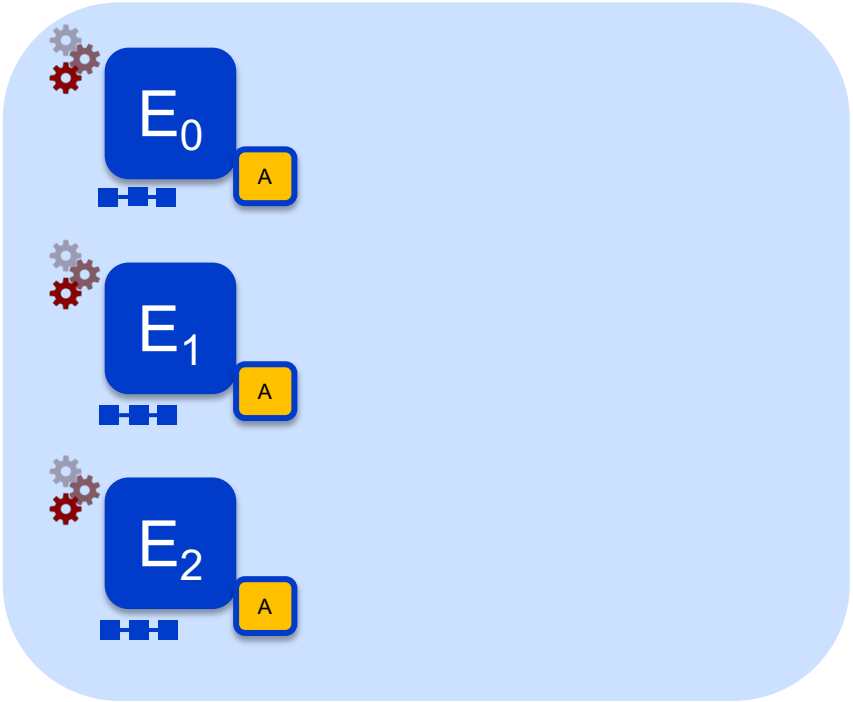
Each execution will capture the set of Read and Written data, called RW sets, which will now flow in the fabric.

Transactions can be signed & encrypted

Key:



Sample transaction: Step 2/7 – Execute proposal



Hyperledger Fabric Network

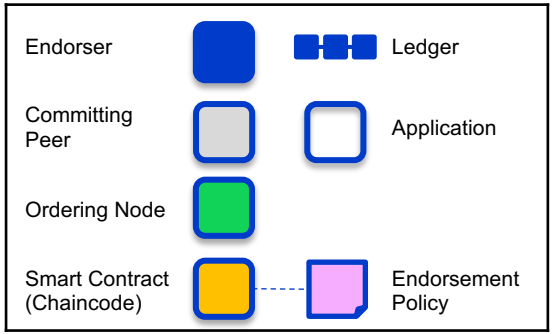
Endorsers Execute Proposals

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

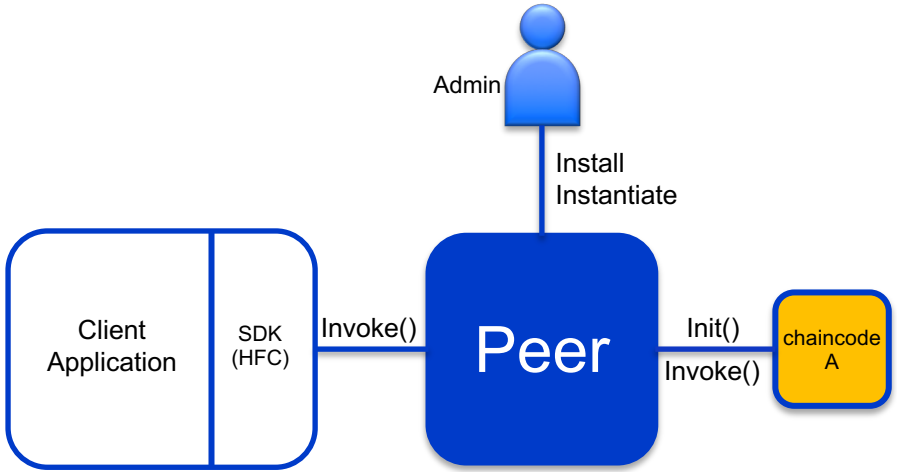
Key:



Smart Contract

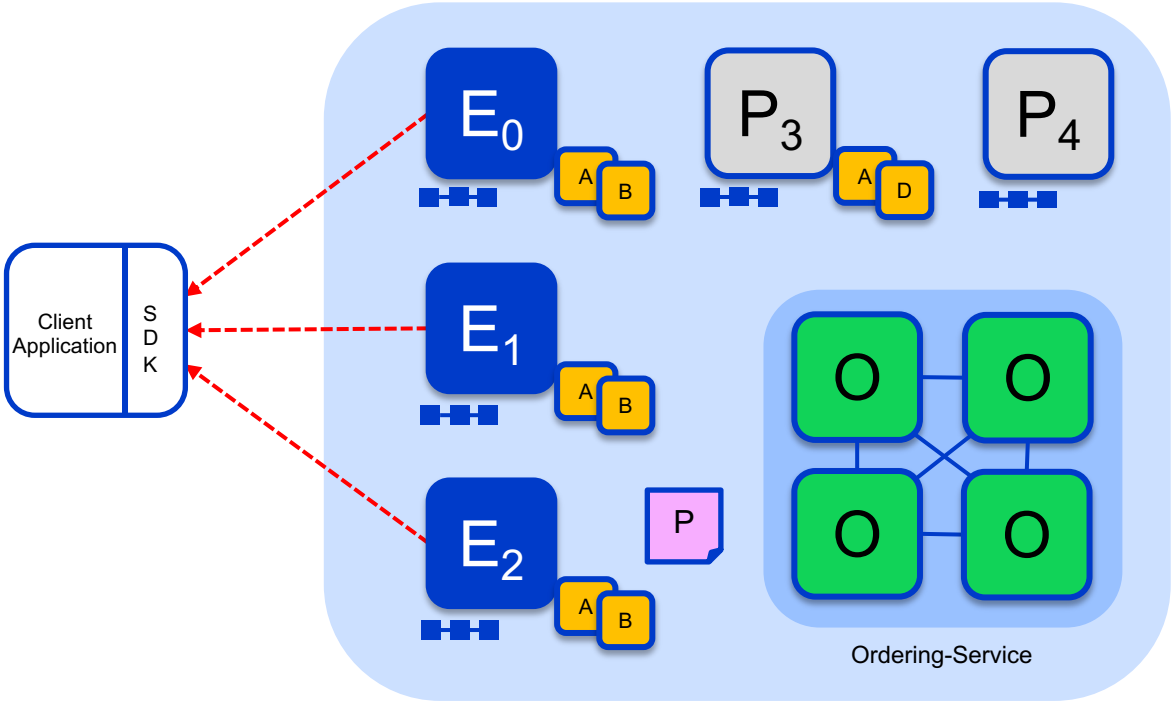
Chaincode (Smart Contract) contains business logic deployed to peers

- Installed on peers and instantiated on channels
- Run in secured docker images separate to the peer
- Interact with the world state through the Fabric shim interface
- Each chaincode has its own scoped world state
- Language support for:
  - Golang
  - Node.js
  - Java
- Implements:
  - Init() - Called on instantiate and upgrade
  - Invoke() – Called from client application



https://jira.hyperledger.org/browse/FAB-8063

Sample transaction: Step 3/7 – Proposal Response



Hyperledger Fabric Network

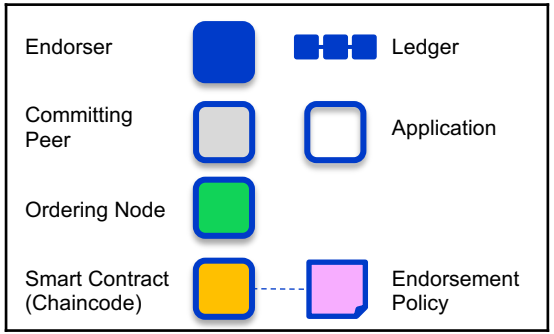
Application receives responses

RW sets are asynchronously returned to application

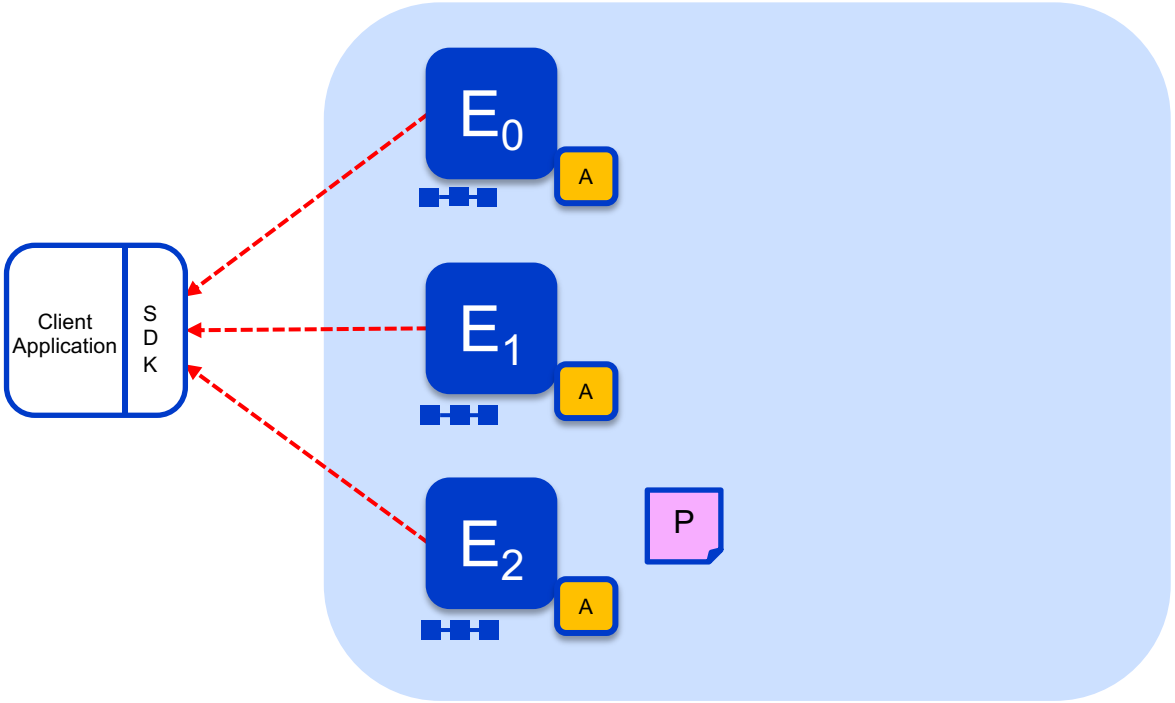
The RW sets are signed by each endorser, and also includes each record version number

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

Key:



Sample transaction: Step 3/7 – Proposal Response



Hyperledger Fabric Network

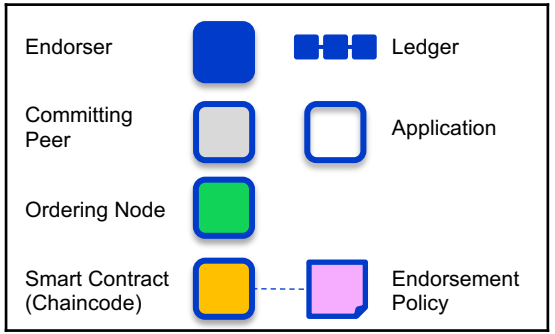
Application receives responses

RW sets are asynchronously returned to application

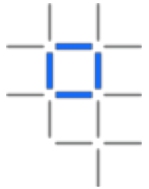
The RW sets are signed by each endorser, and also includes each record version number

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

Key:



Non-determinism in Blockchain



- Blockchain is a distributed processing system
  - Smart contracts are run multiple times and in multiple places
  - As we will see, smart contracts need to run deterministically in order for consensus to work
    - Particularly when updating the world state
- It's particularly difficult to achieve determinism with off-chain processing
  - Implement oracle services that are guaranteed to be consistent for a given transaction, or
  - Detect duplicates for a transaction in the blockchain, middleware or external system

random()

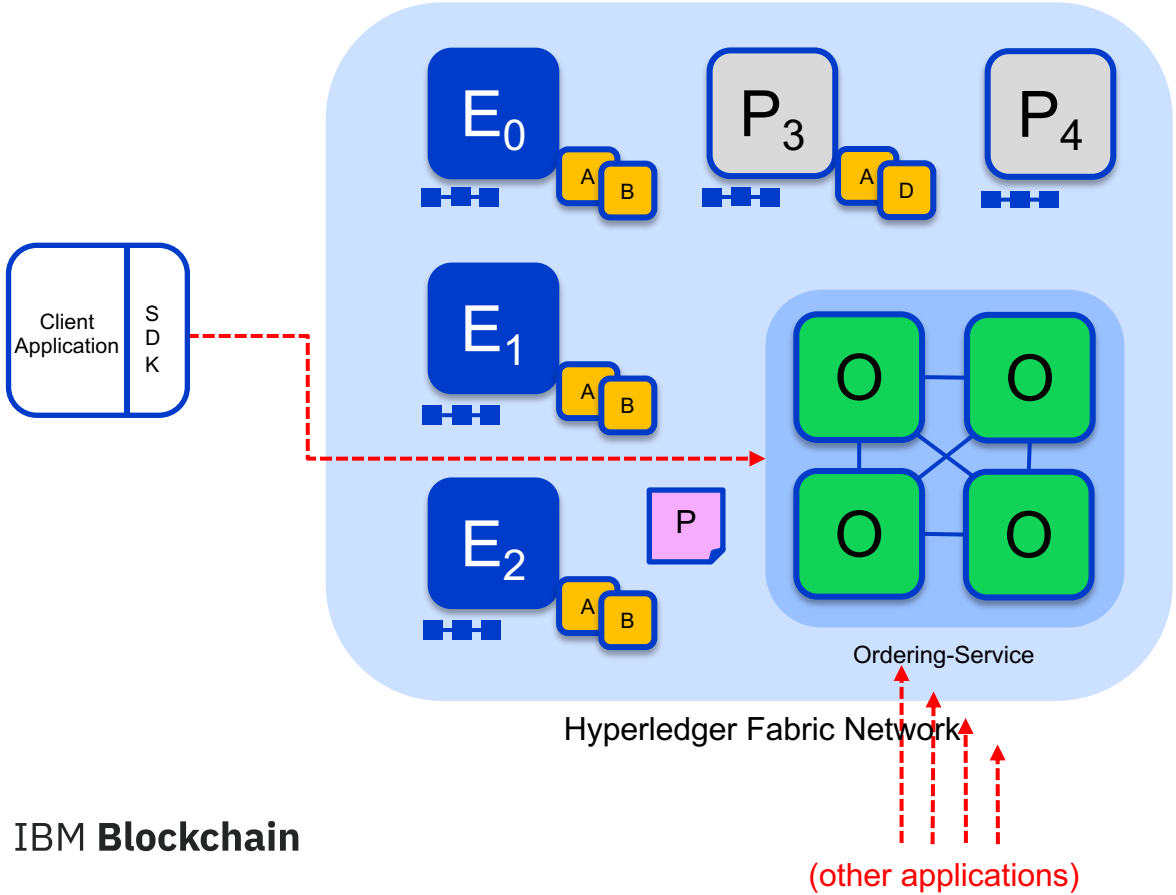
getExchangeRate()

getDateTime()

getTemperature()

incrementValue inExternalSystem(…)

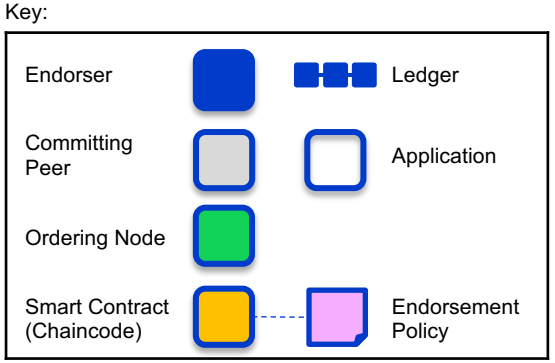
Sample transaction: Step 4/7 – Order Transaction



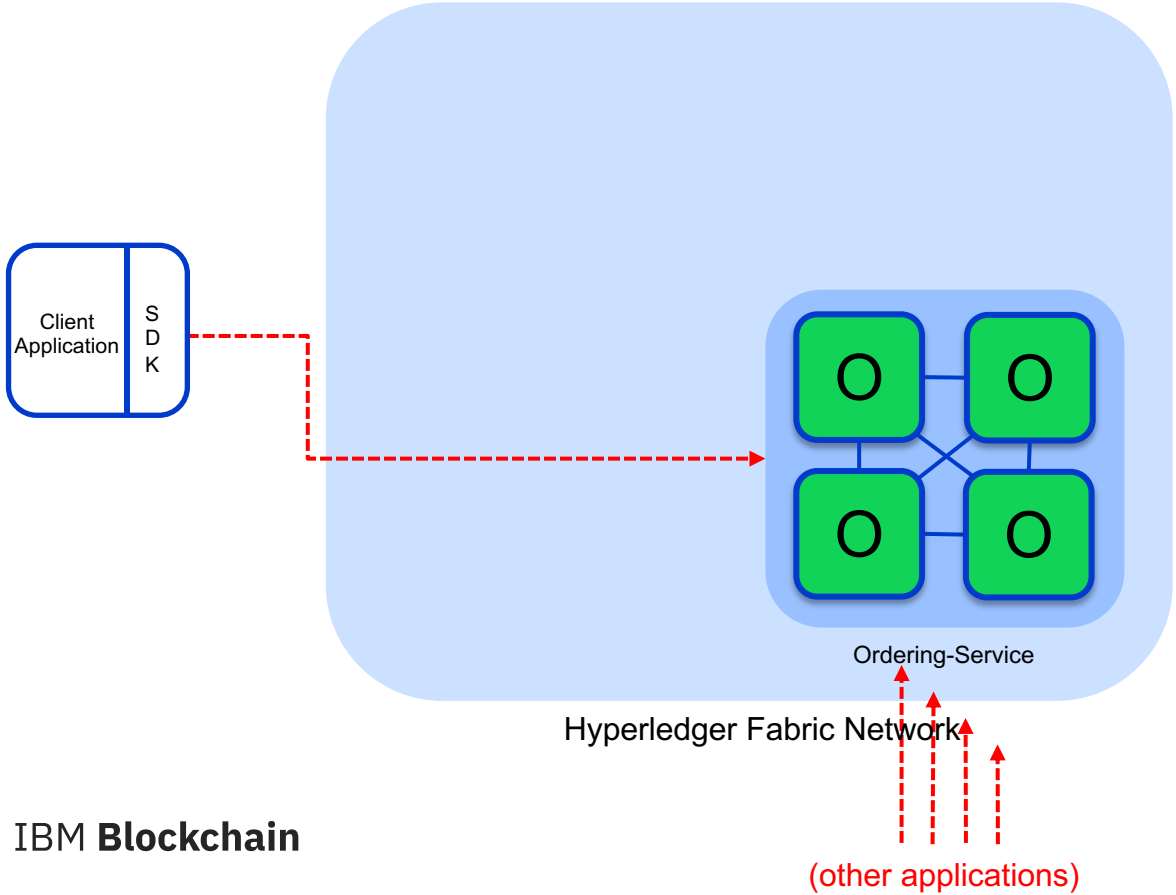
Responses submitted for ordering

Application submits responses as a transaction to be ordered.

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



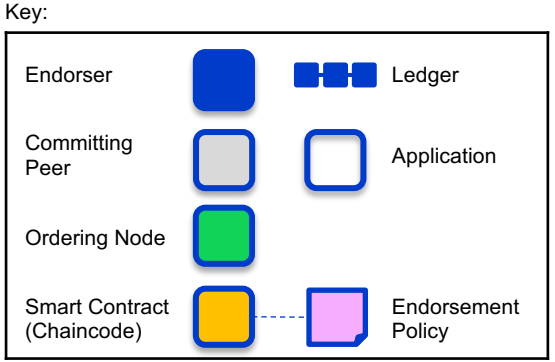
Sample transaction: Step 4/7 – Order Transaction



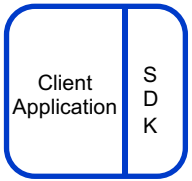
Responses submitted for ordering

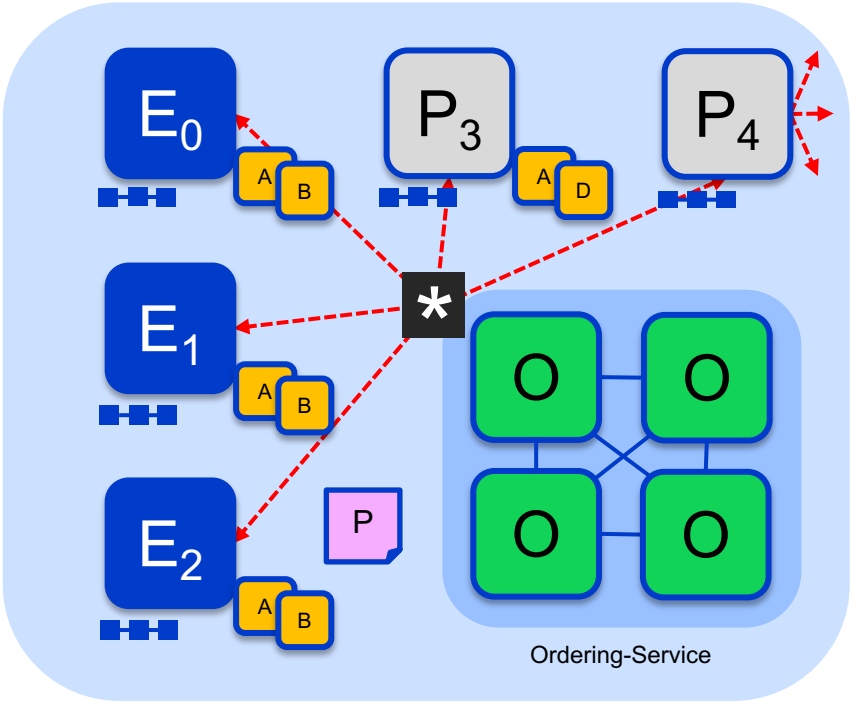
Application submits responses as a transaction to be ordered.

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



Sample transaction: Step 5/7 – Deliver Transaction





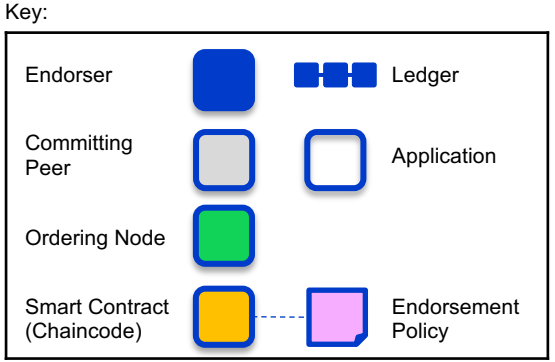
Hyperledger Fabric Network

Orderer delivers to committing peers

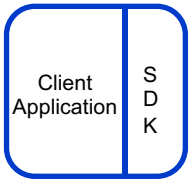
Ordering service collects transactions into proposed blocks for distribution to committing peers. Peers can deliver to other peers in a hierarchy (not shown)

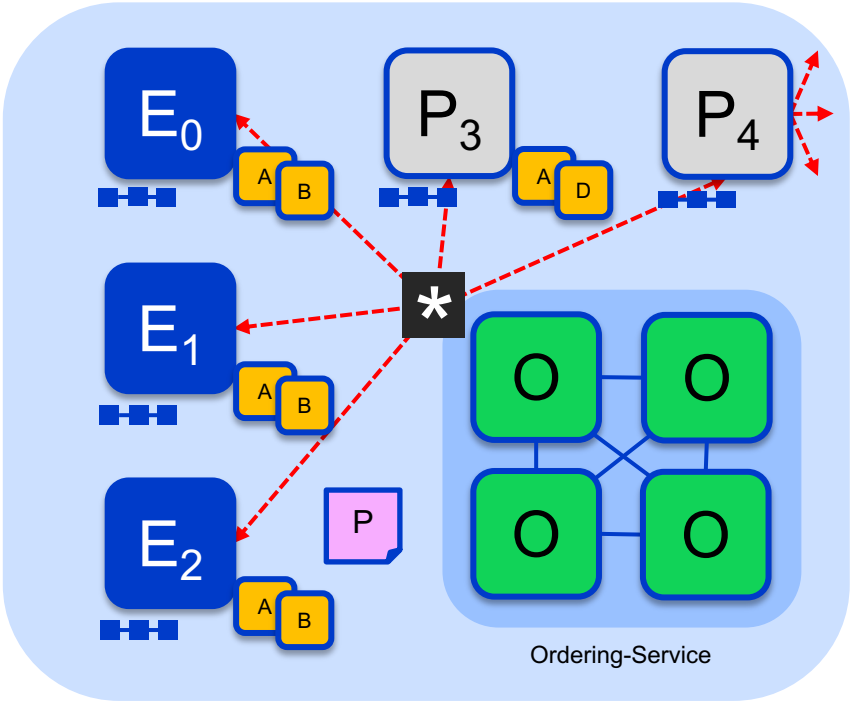
Different ordering algorithms available:

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



Sample transaction: Step 5/7 – Deliver Transaction





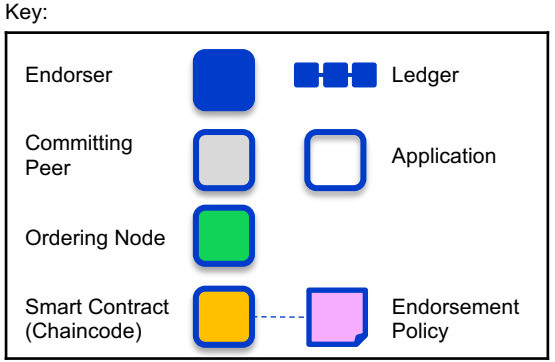
Hyperledger Fabric Network

Orderer delivers to committing peers

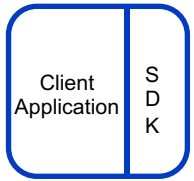
Ordering service collects transactions into proposed blocks for distribution to committing peers. Peers can deliver to other peers in a hierarchy (not shown)

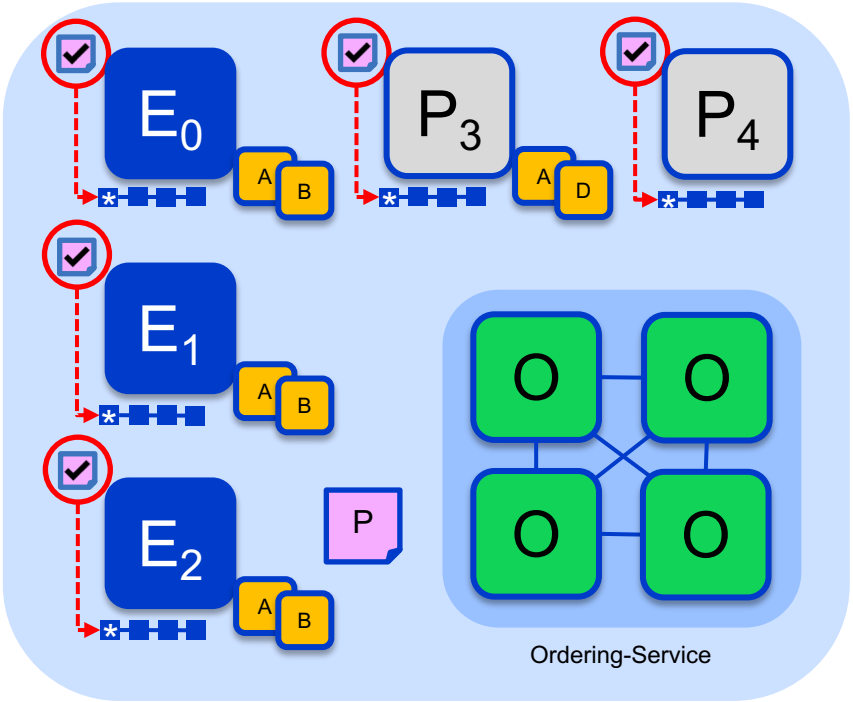
Different ordering algorithms available:

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



Sample transaction: Step 6/7 – Validate Transaction





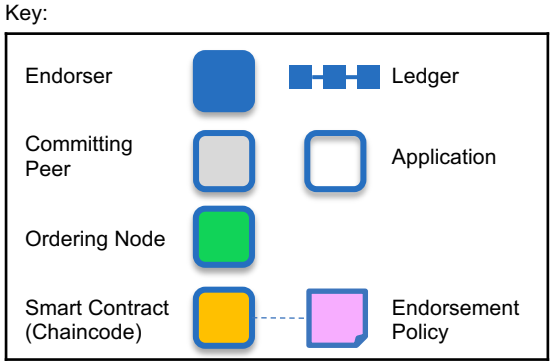
Hyperledger Fabric Network

Committing peers validate transactions

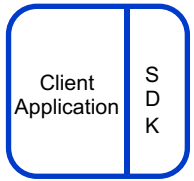
Every committing peer validates against the endorsement policy. Also check RW sets are still valid for current world state

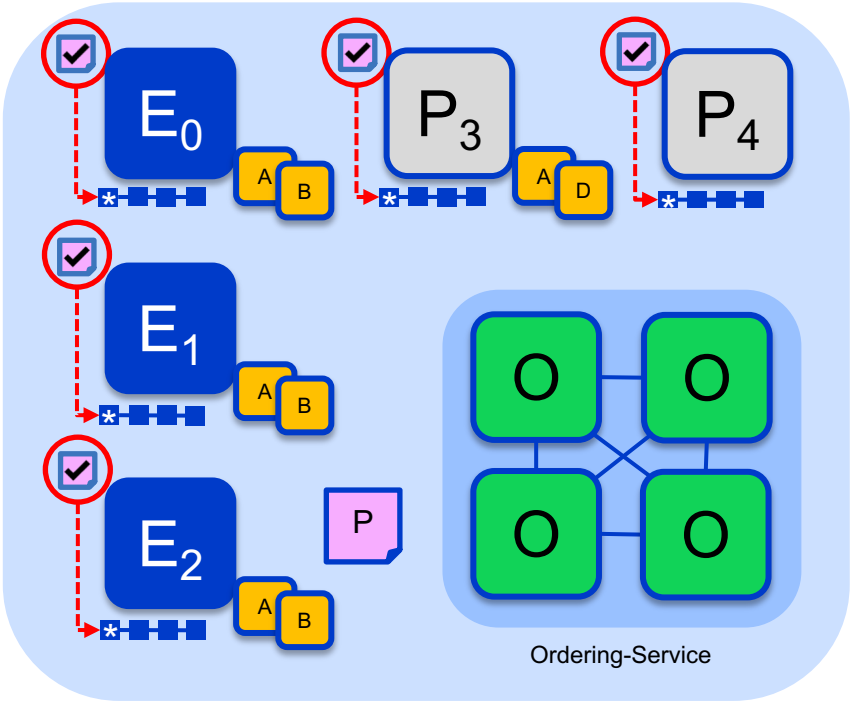
Validated transactions are applied to the world state and retained on the ledger

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



Sample transaction: Step 6/7 – Validate Transaction





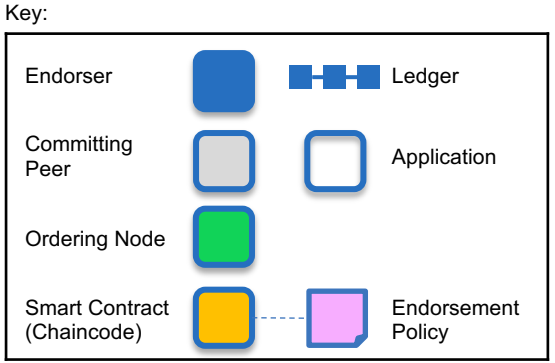
Hyperledger Fabric Network

Committing peers validate transactions

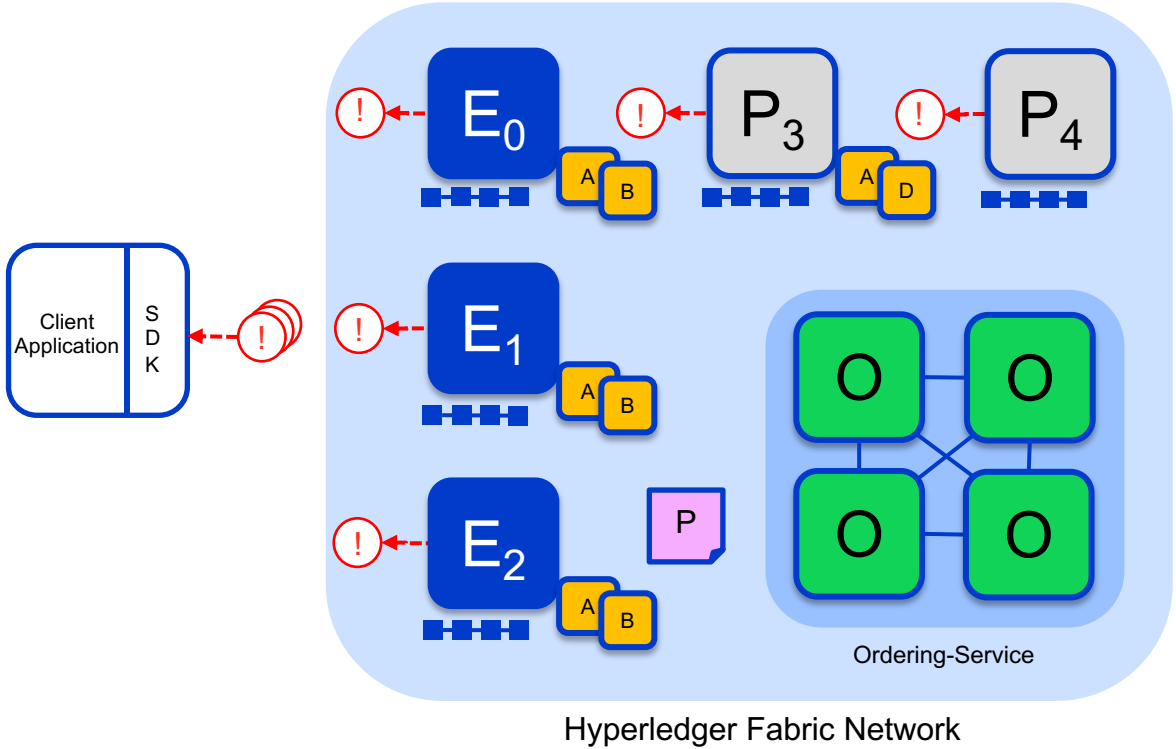
Every committing peer validates against the endorsement policy. Also check RW sets are still valid for current world state

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

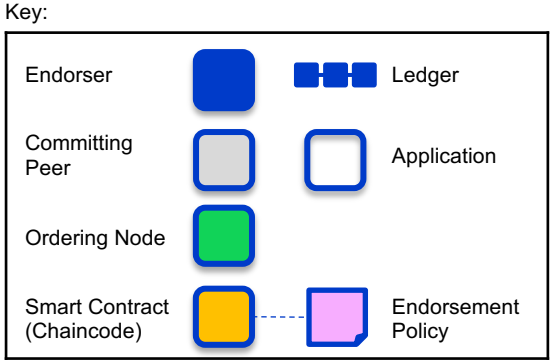


Hyperledger Fabric Network

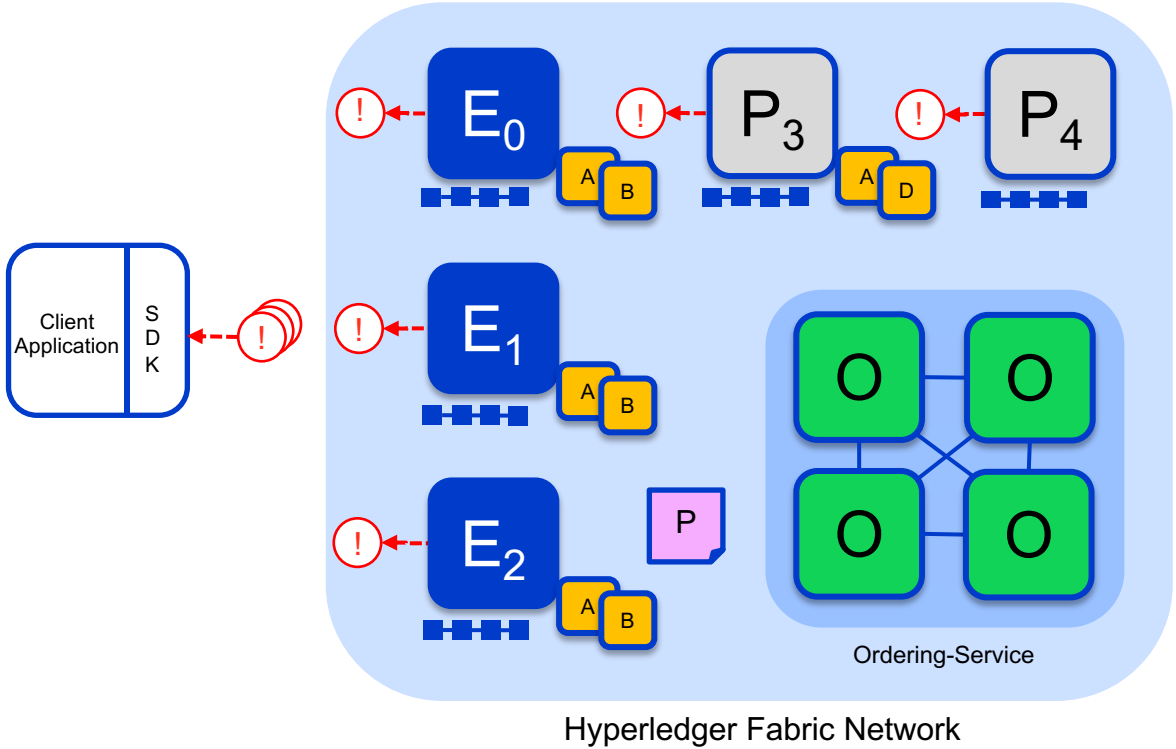
Committing peers notify applications

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



Sample transaction: Step 7/7 – Notify Transaction

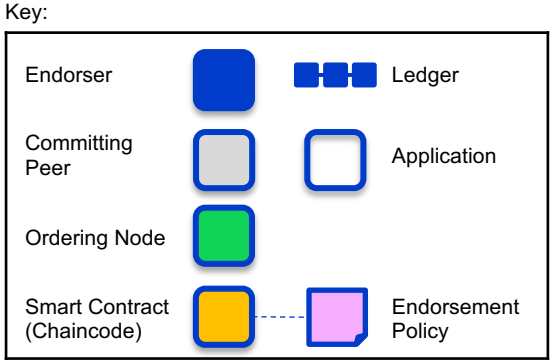


Hyperledger Fabric Network

Committing peers notify applications

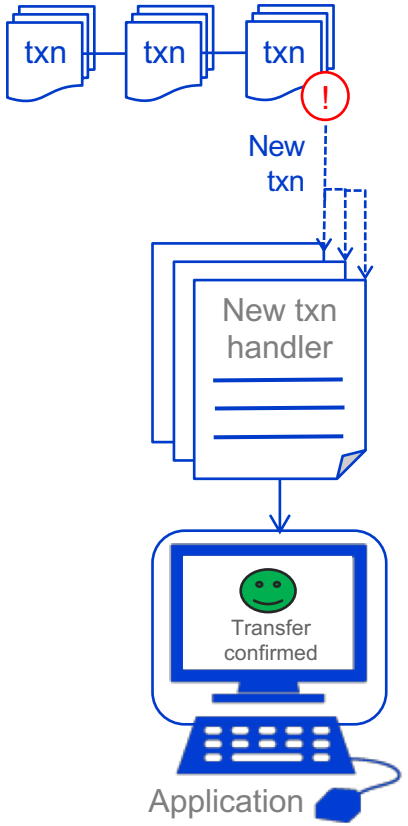
Applications can register to be notified when transactions succeed or fail, and when blocks are added to the ledger

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

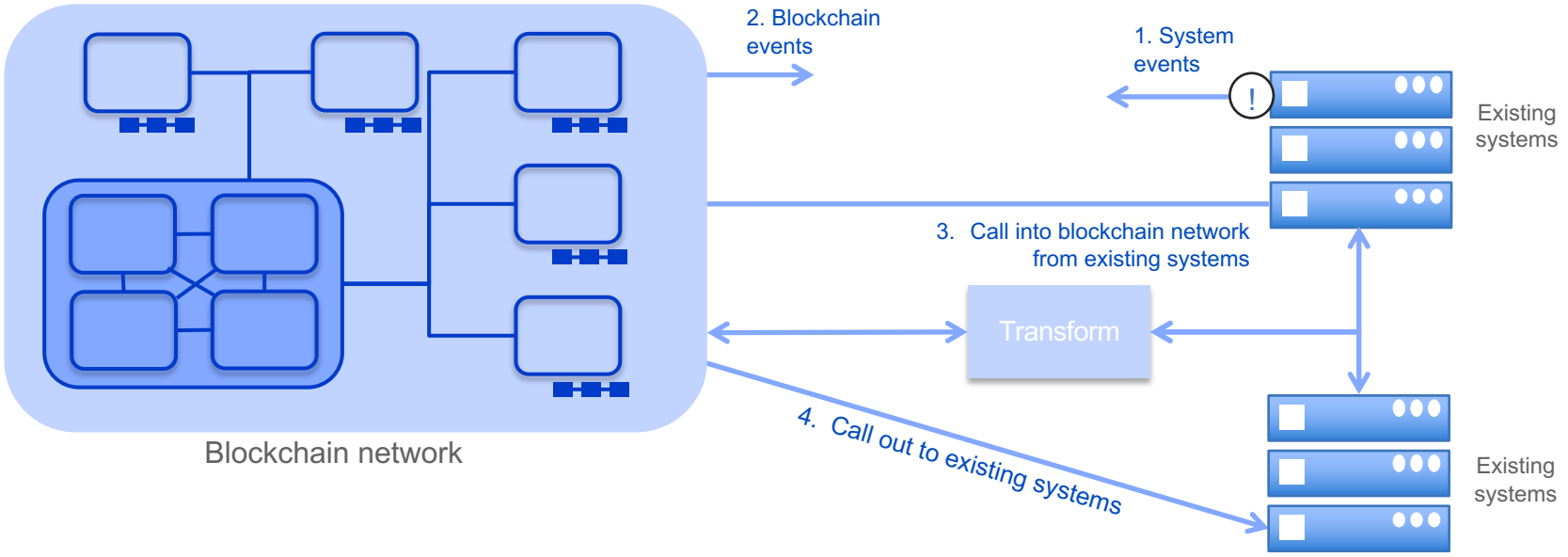


How Events are Used in Blockchain

- In computing, an event is an occurrence that can trigger handlers
  - e.g. disk full, fail transfer completed, mouse clicked, message received, temperature too hot…
- Events are important in asynchronous processing systems like blockchain
- The blockchain can emit events that are useful to application programmers
  - e.g. Transaction has been validated or rejected, block has been added…
- Events from external systems might also trigger blockchain activity
  - e.g. exchange rate has gone below a threshold, the temperature has gone up, a time period has elapsed…

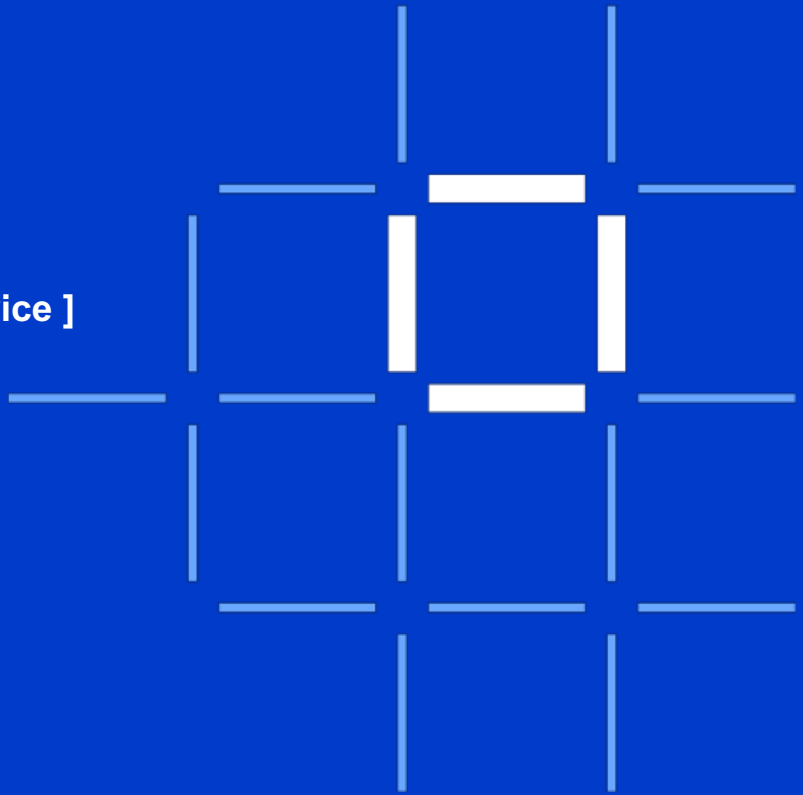


Integrating with Existing Systems – Possibilities



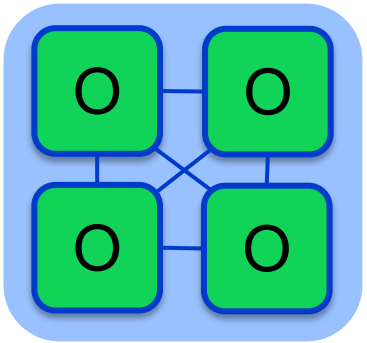
Technical Deep Dive

- **Architectural Overview**
- **Network Consensus**
- **[Channels and Ordering Service]**
- Components
- 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.

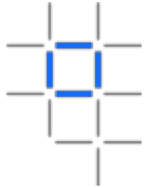


Ordering Service

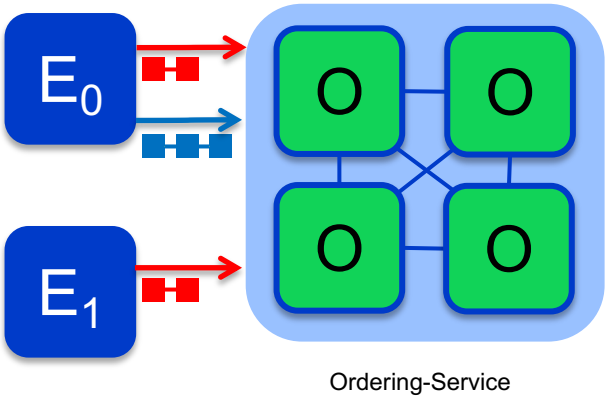
Different configuration options for the ordering service include:

- SOLO
  - Single node for development
- Kafka : Crash fault tolerant consensus
  - Minimum recommendation
    - 3 Orderer nodes
    - 4 Kafka nodes
    - 3 Zookeeper nodes, must be odd number

Channels

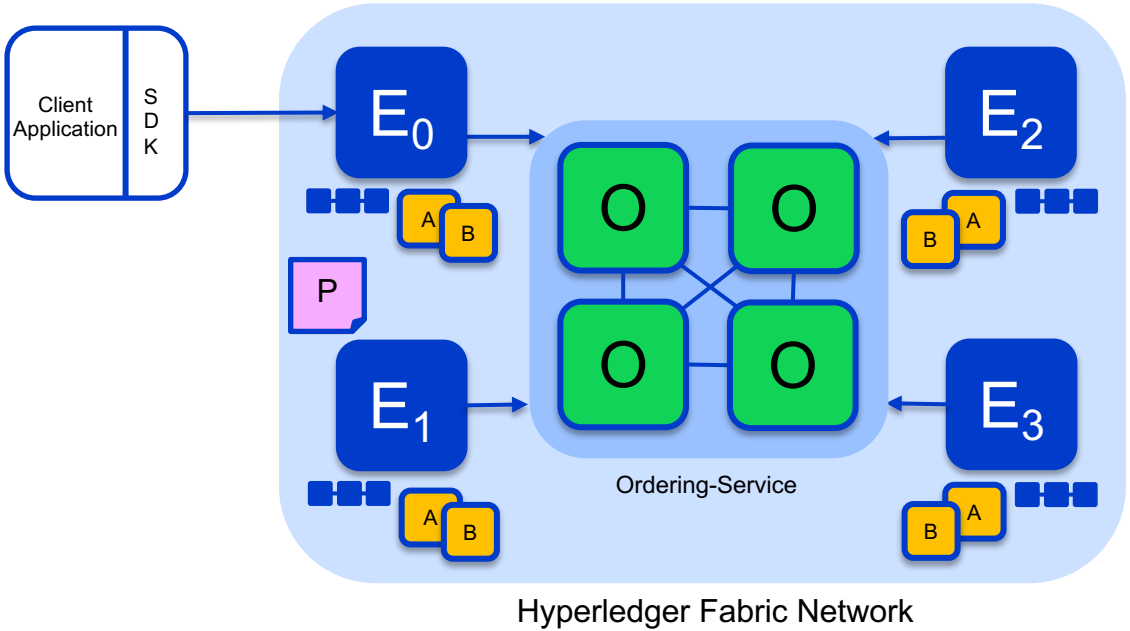


Channels provide privacy between different ledgers

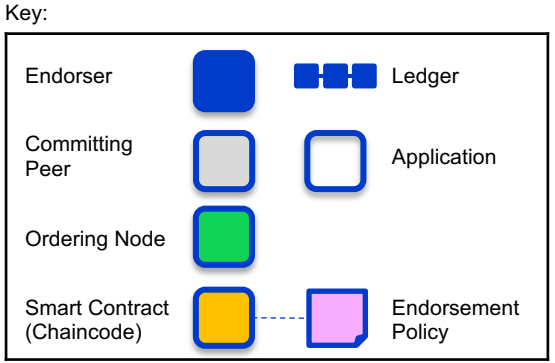


- Ledgers exist in the scope of a channel
  - Channels can be shared across an entire network of peers
  - Channels can be permissioned for a specific set of participants
- Chaincode is installed on peers to access the worldstate
- Chaincode is instantiated on specific channels
- Peers can participate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

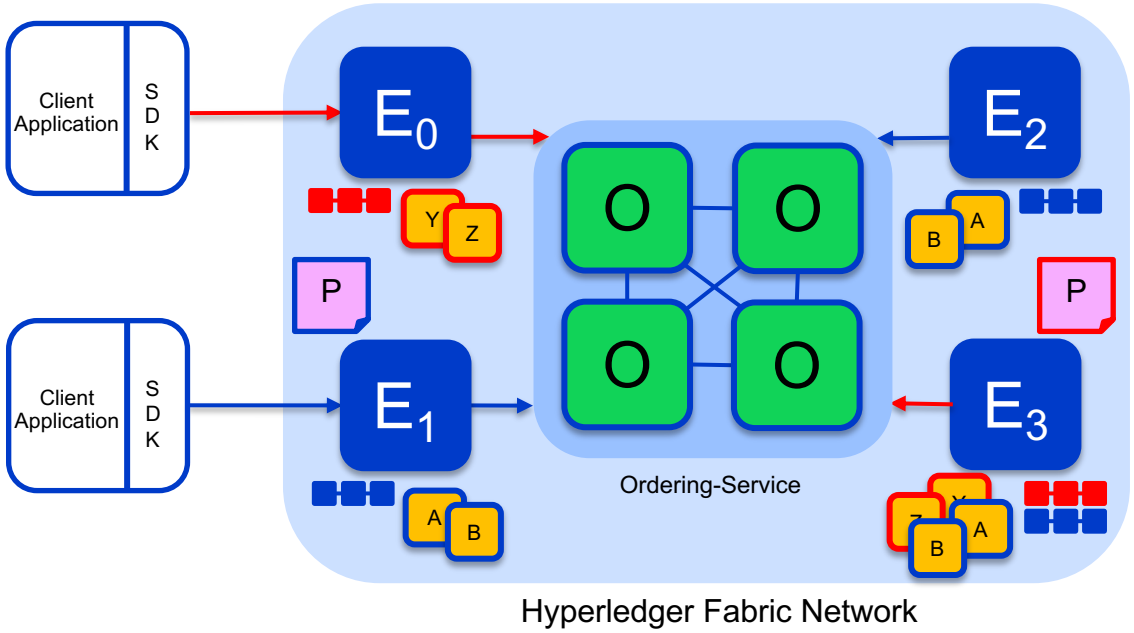
Single Channel Network



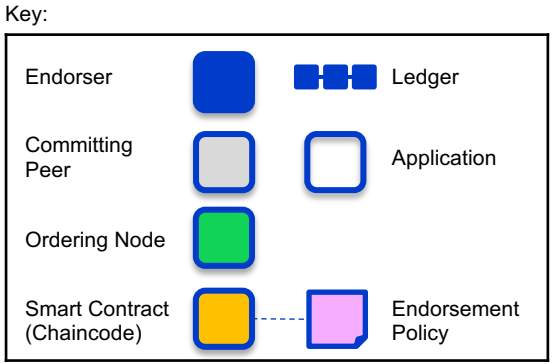
- Similar to v0.6 PBFT model
- All peers connect to the same system channel (blue).
- All peers have the same chaincode and maintain the same ledger
- Endorsement by peers E_0, E_1, E_2 and E_3



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



Thank you

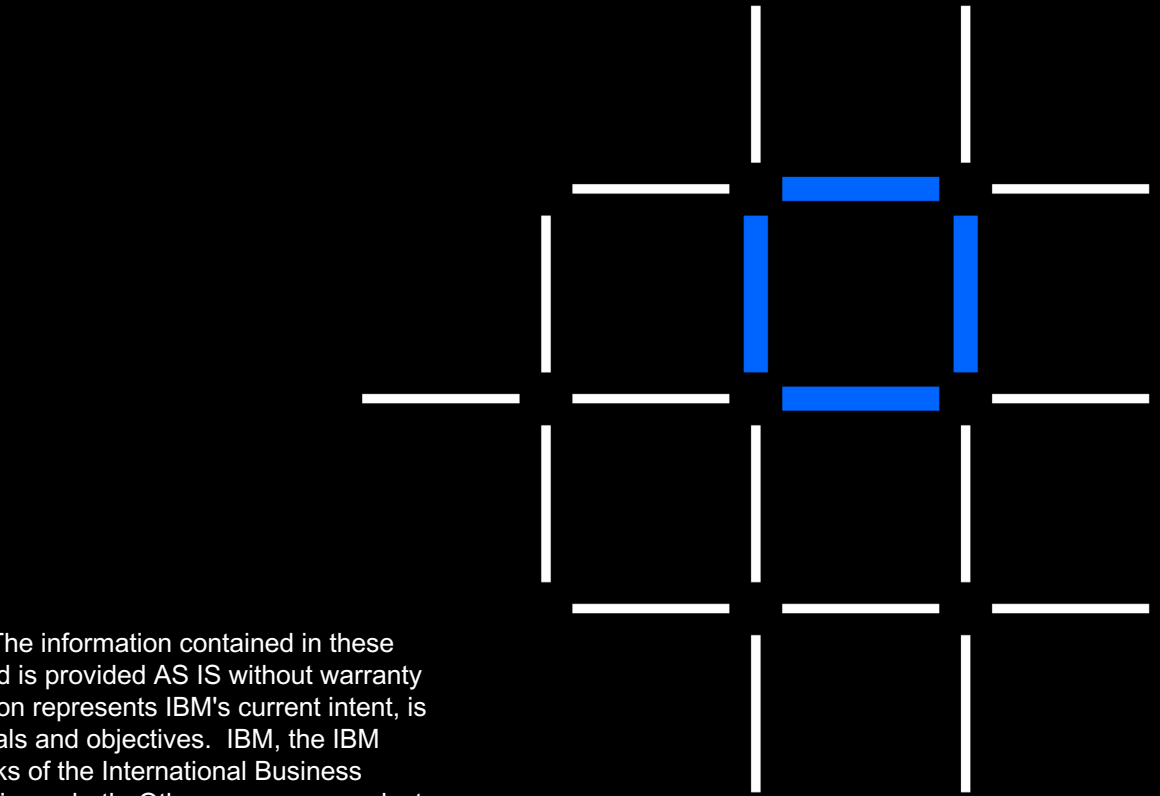
Barry Silliman
silliman@us.ibm.com

IBM **Blockchain**

www.ibm.com/blockchain

developer.ibm.com/blockchain

www.hyperledger.org



IBM