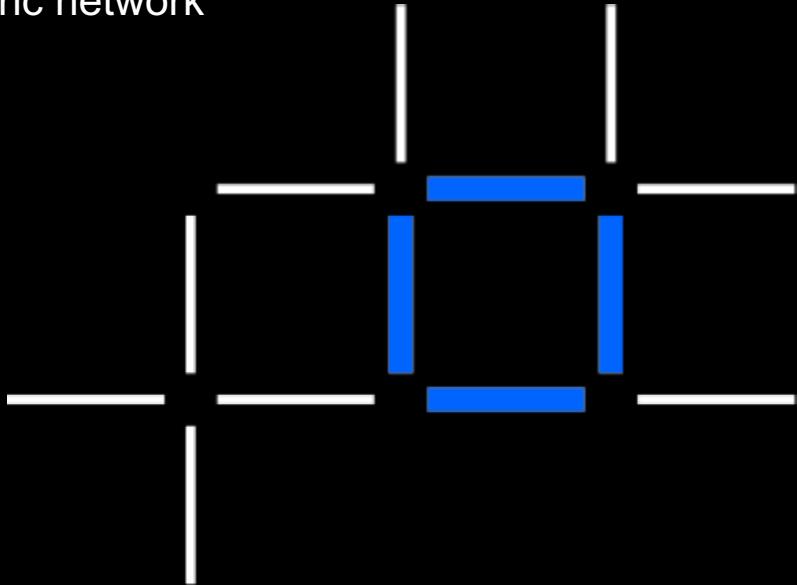


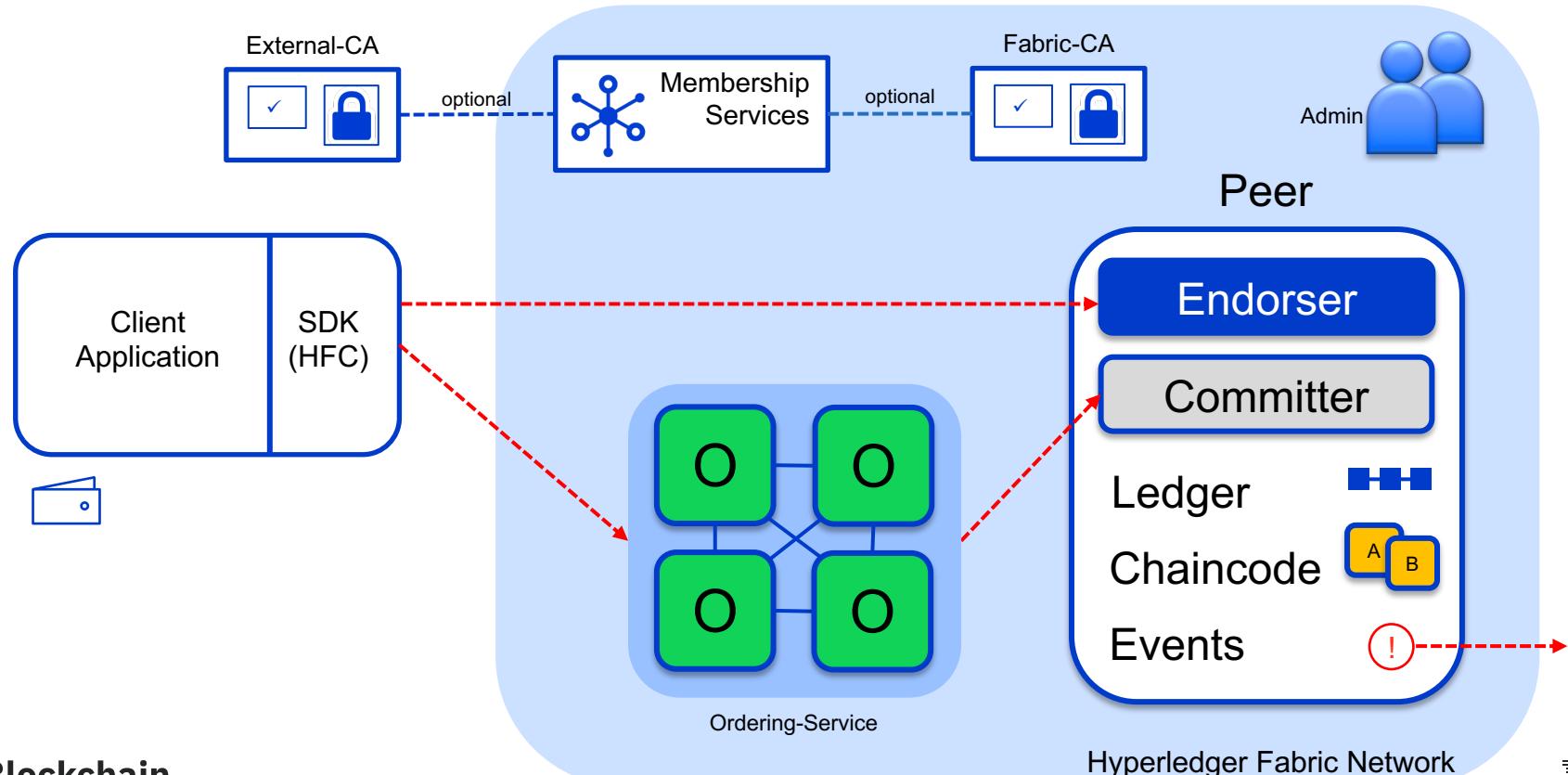
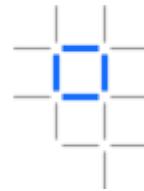
Blockchain Explored, Part 2

Transaction flow and consensus in a Hyperledger Fabric network

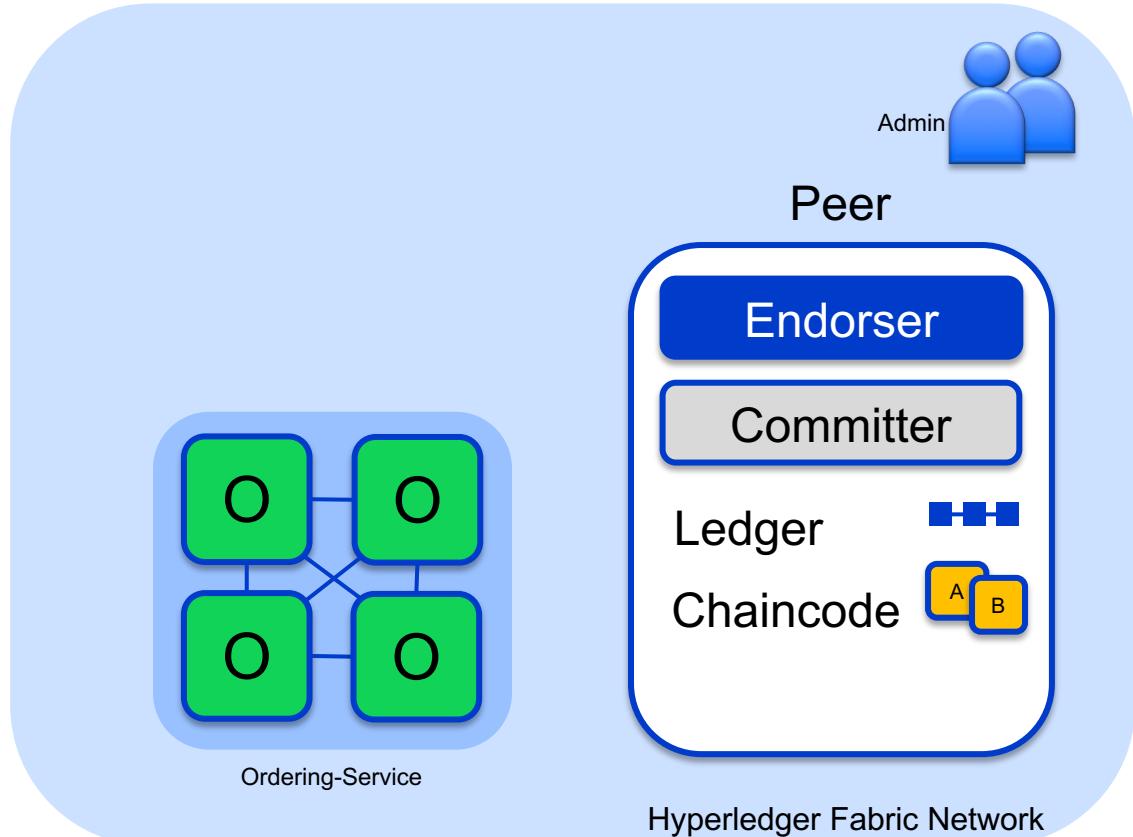
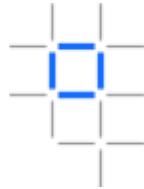
*Barry Silliman
IBM Washington Systems Center
silliman@us.ibm.com*



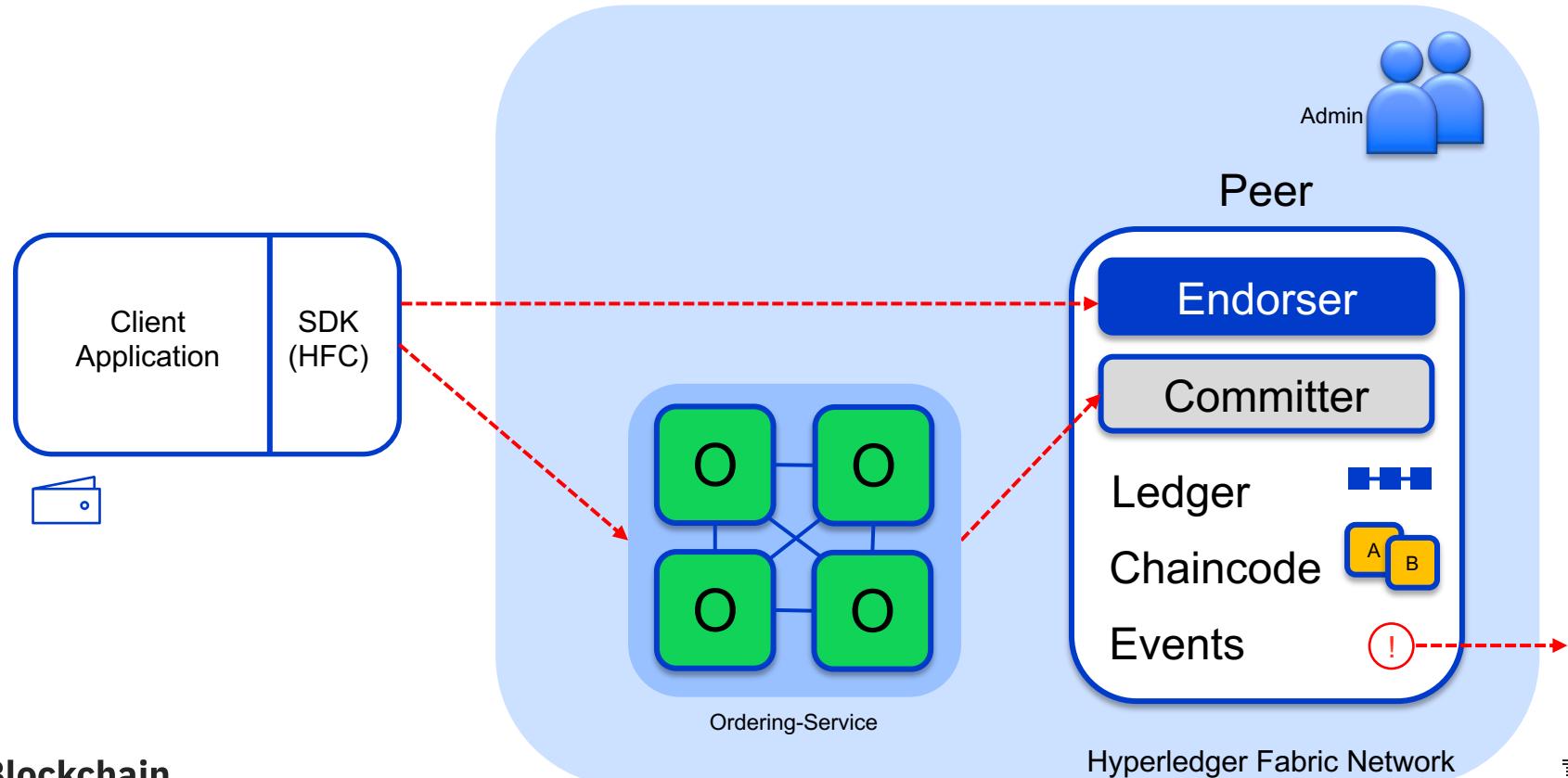
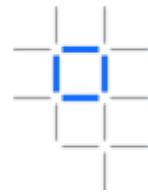
Hyperledger Fabric V1.x Architecture

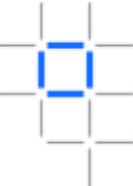


Hyperledger Fabric V1.x Architecture

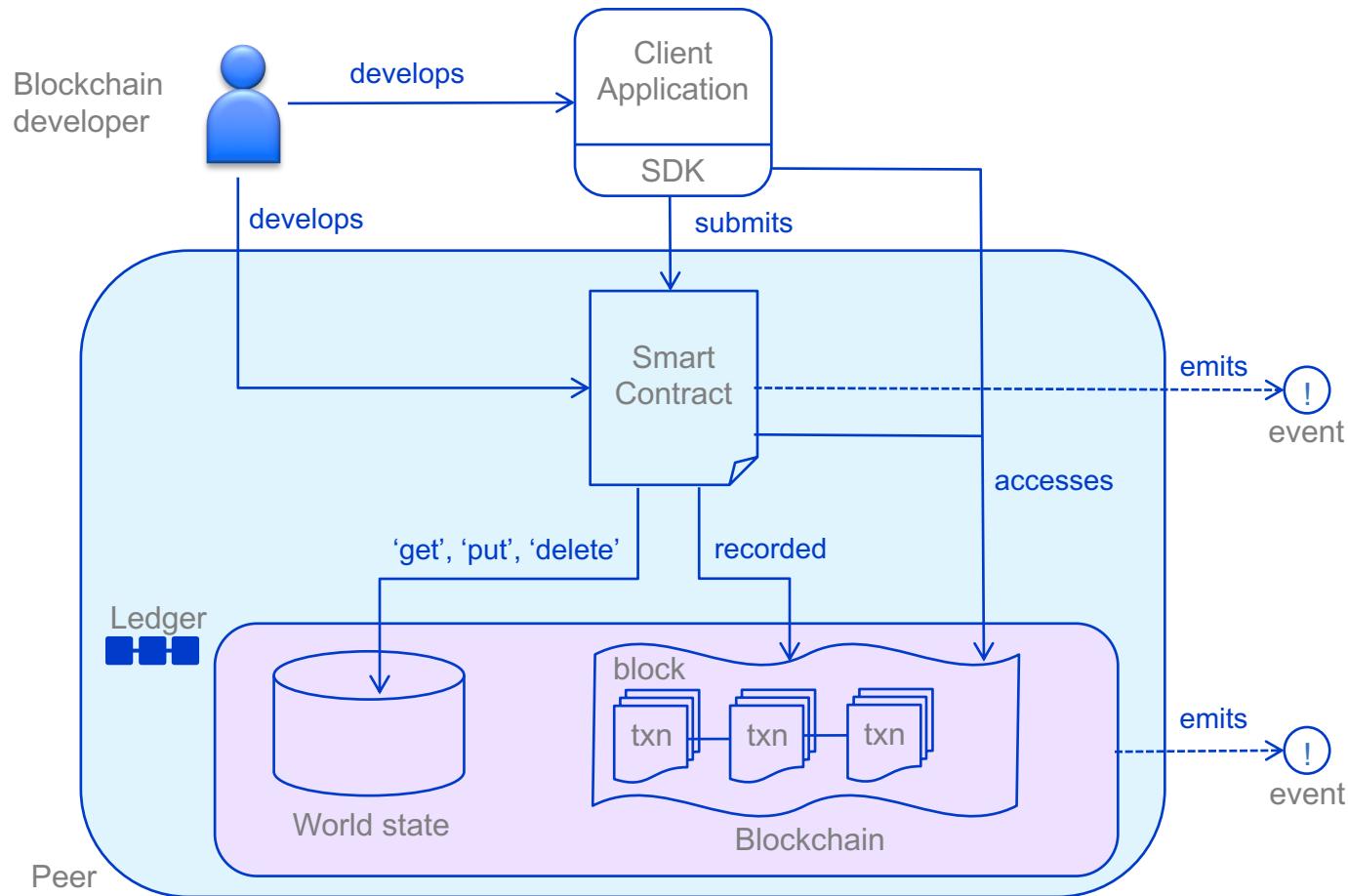


Hyperledger Fabric V1.x Architecture

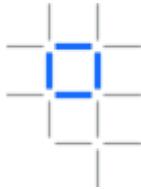




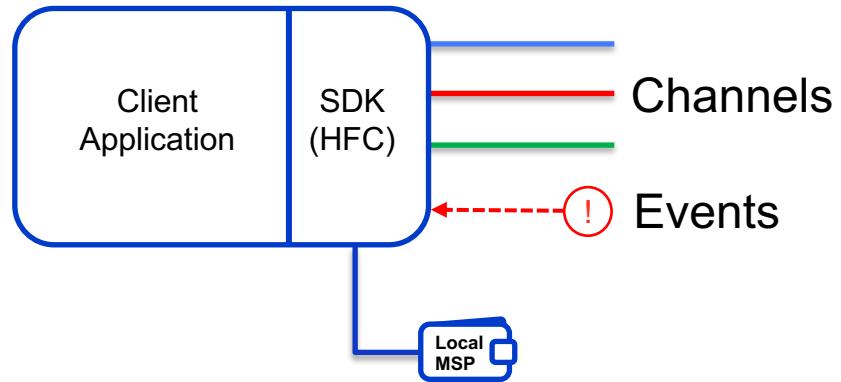
How applications interact with the ledger



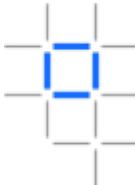
Client Application



- Each client application uses Fabric SDK to:
 - Connect over channels to one or more peers
 - Connect over channels to one or more orderer nodes
 - Receive events from peers
- Local MSP provides client [crypto material](#)
- Client can be written in different languages
(Node.js, Go, Java, Python)

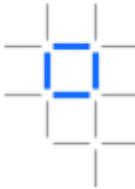


Nodes and roles

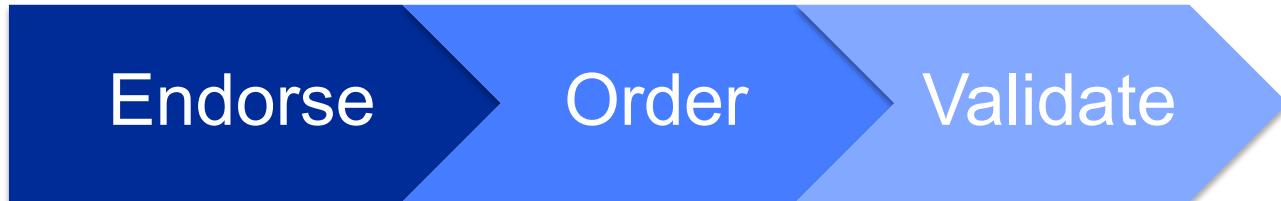


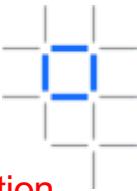
	<p>Peer: Maintains ledger and state. Commits transactions. May hold smart contract (chaincode).</p>
	<p>Endorsing Peer: Specialized peer also endorses transactions by receiving a transaction proposal and responds by granting or denying endorsement. Must hold smart contract.</p>
	<p>Ordering Node: Approves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

Hyperledger Fabric Consensus

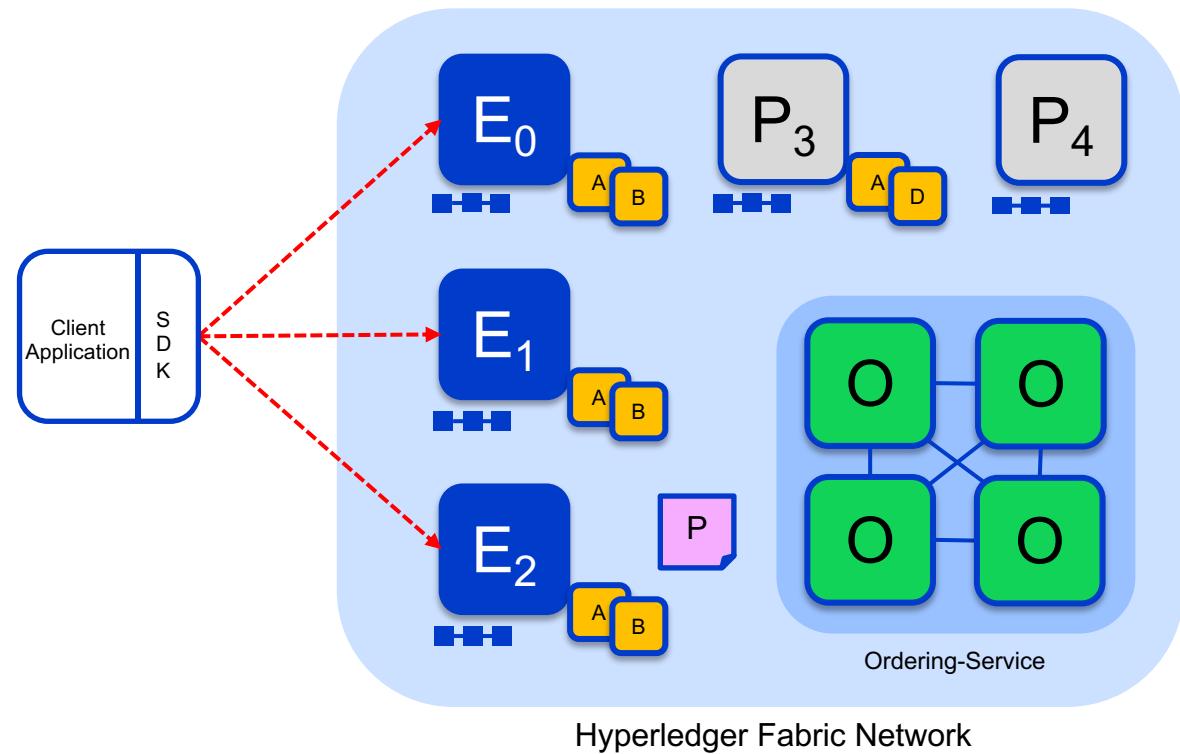


Consensus is achieved using the following transaction flow:





Sample transaction: Step 1/7 – Propose transaction



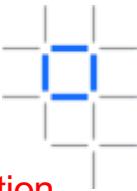
Application proposes transaction

Endorsement policy:
• “ E_0 , E_1 and E_2 must sign”
• (P_3 , P_4 are not part of the policy)

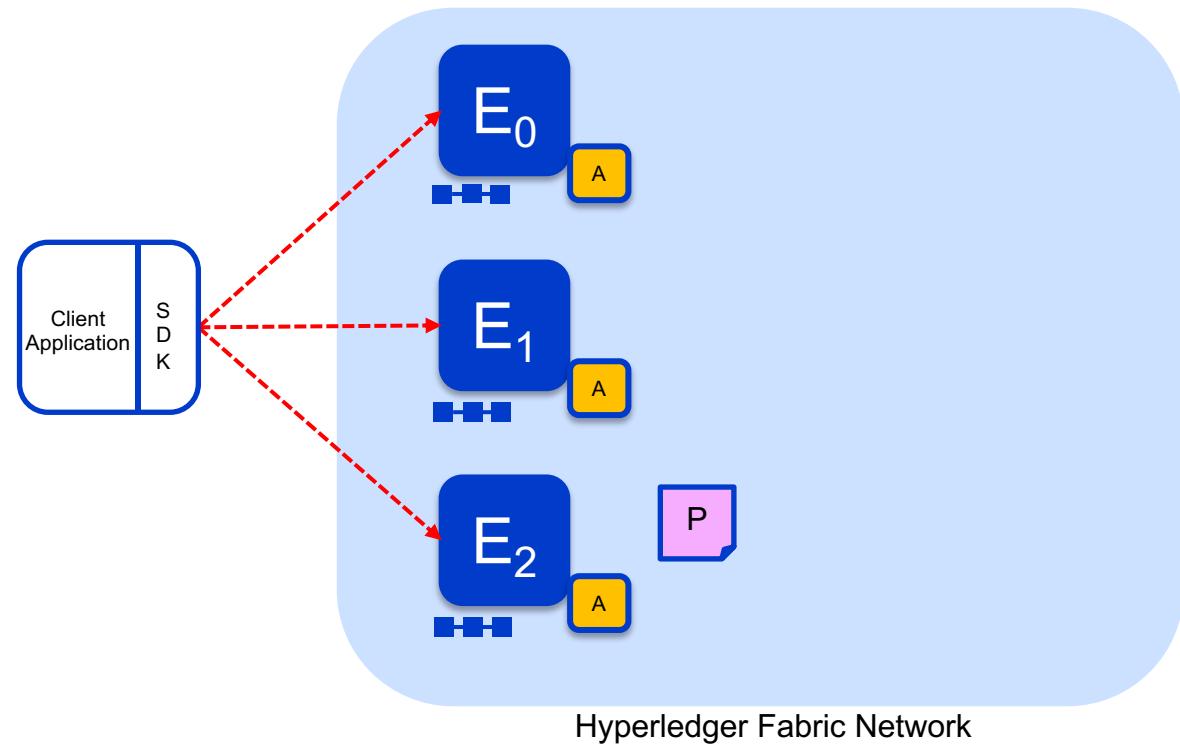
Client application submits a transaction proposal for Smart Contract A. It must target the required peers $\{E_0, E_1, E_2\}$

Key:

Endorser		Ledger
Committing Peer		Application
Ordering Node		
Smart Contract (Chaincode)		Endorsement Policy



Sample transaction: Step 1/7 – Propose transaction



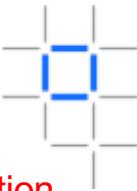
Application proposes transaction

Endorsement policy:
• “ E_0 , E_1 and E_2 must sign”
• (P_3 , P_4 are not part of the policy)

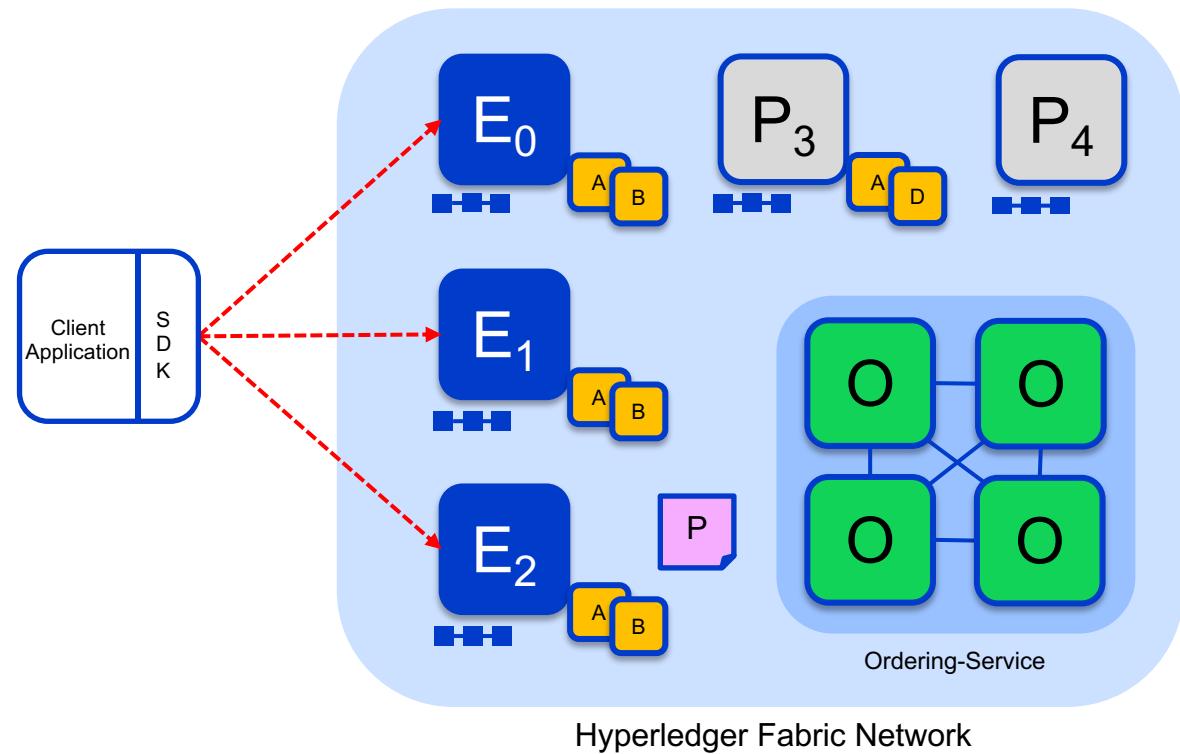
Client application submits a transaction proposal for Smart Contract A. It must target the required peers $\{E_0, E_1, E_2\}$

Key:

Endorser		Ledger
Committing Peer		Application
Ordering Node		
Smart Contract (Chaincode)		Endorsement Policy



Sample transaction: Step 1/7 – Propose transaction



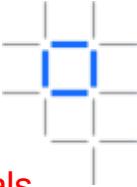
Application proposes transaction

Endorsement policy:
• “ E_0 , E_1 and E_2 must sign”
• (P_3 , P_4 are not part of the policy)

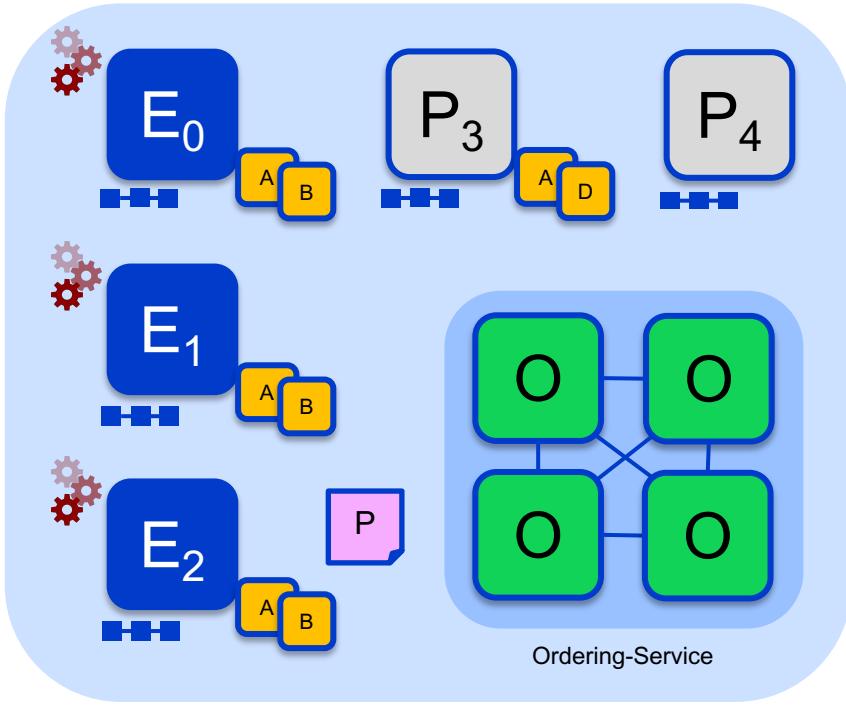
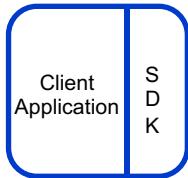
Client application submits a transaction proposal for Smart Contract A. It must target the required peers $\{E_0, E_1, E_2\}$

Key:

Endorser		Ledger
Committing Peer		Application
Ordering Node		
Smart Contract (Chaincode)		Endorsement Policy



Sample transaction: Step 2/7 – Execute proposal



Endorsers Execute Proposals

E₀, E₁ & E₂ will each execute the proposed transaction. None of these executions will update the ledger

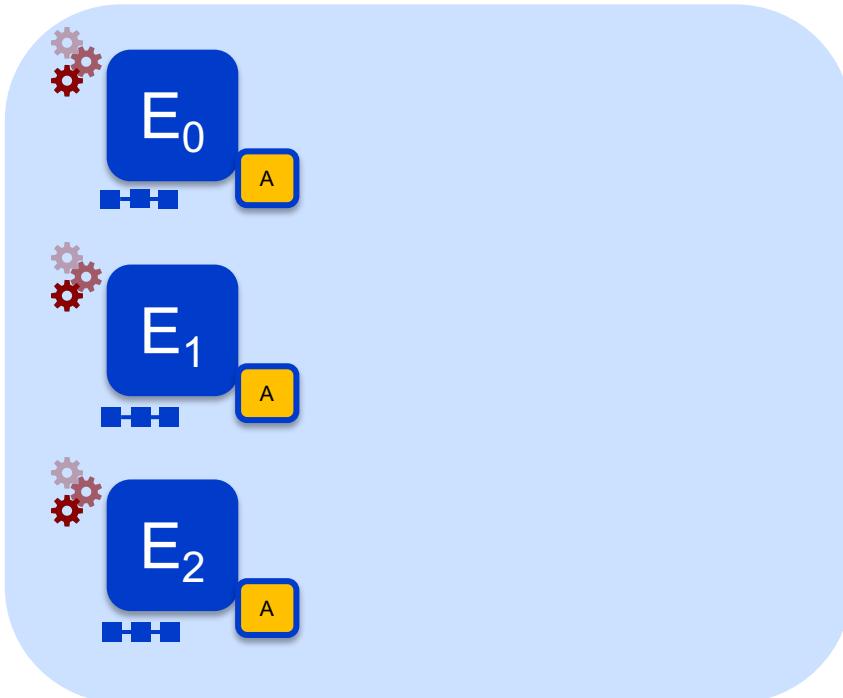
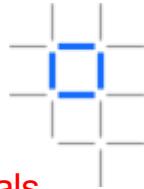
Each execution will capture the set of Read and Written data, called RW sets, which will now flow in the fabric.

Transactions can be signed & encrypted

Key:

Endorser		Ledger
Committing Peer		Application
Ordering Node		
Smart Contract (Chaincode)		Endorsement Policy

Sample transaction: Step 2/7 – Execute proposal



Endorsers Execute Proposals

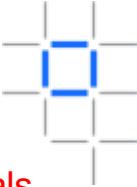
E₀, E₁ & E₂ will each execute the proposed transaction. None of these executions will update the ledger

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

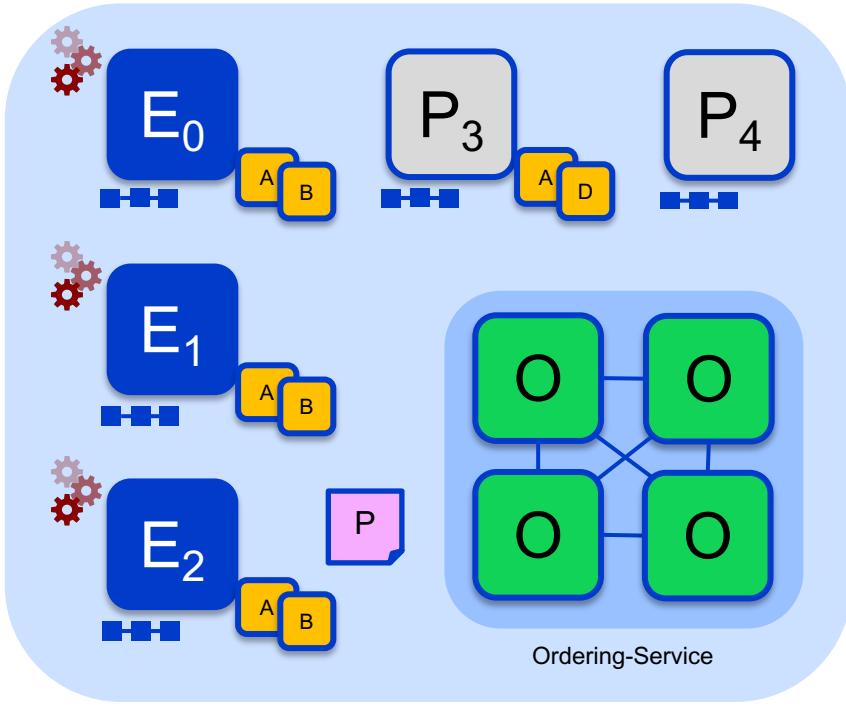
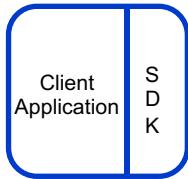
Transactions can be signed & encrypted

Key:

Endorser		Ledger
Committing Peer		Application
Ordering Node		
Smart Contract (Chaincode)		Endorsement Policy



Sample transaction: Step 2/7 – Execute proposal



Endorsers Execute Proposals

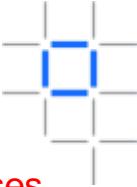
E₀, E₁ & E₂ will each execute the proposed transaction. None of these executions will update the ledger

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

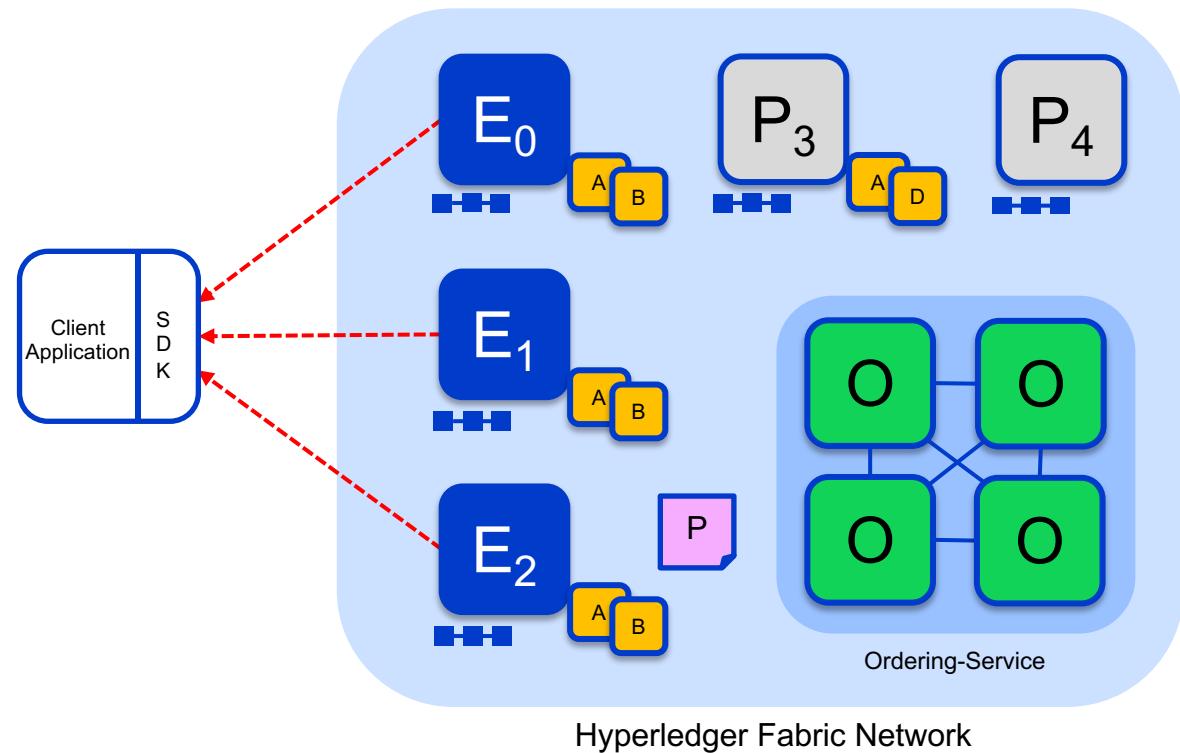
Transactions can be signed & encrypted

Key:

Endorser		Ledger
Committing Peer		Application
Ordering Node		
Smart Contract (Chaincode)		Endorsement Policy



Sample transaction: Step 3/7 – Proposal Response



Application receives responses

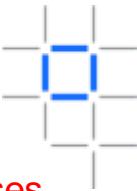
RW sets are asynchronously returned to application

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

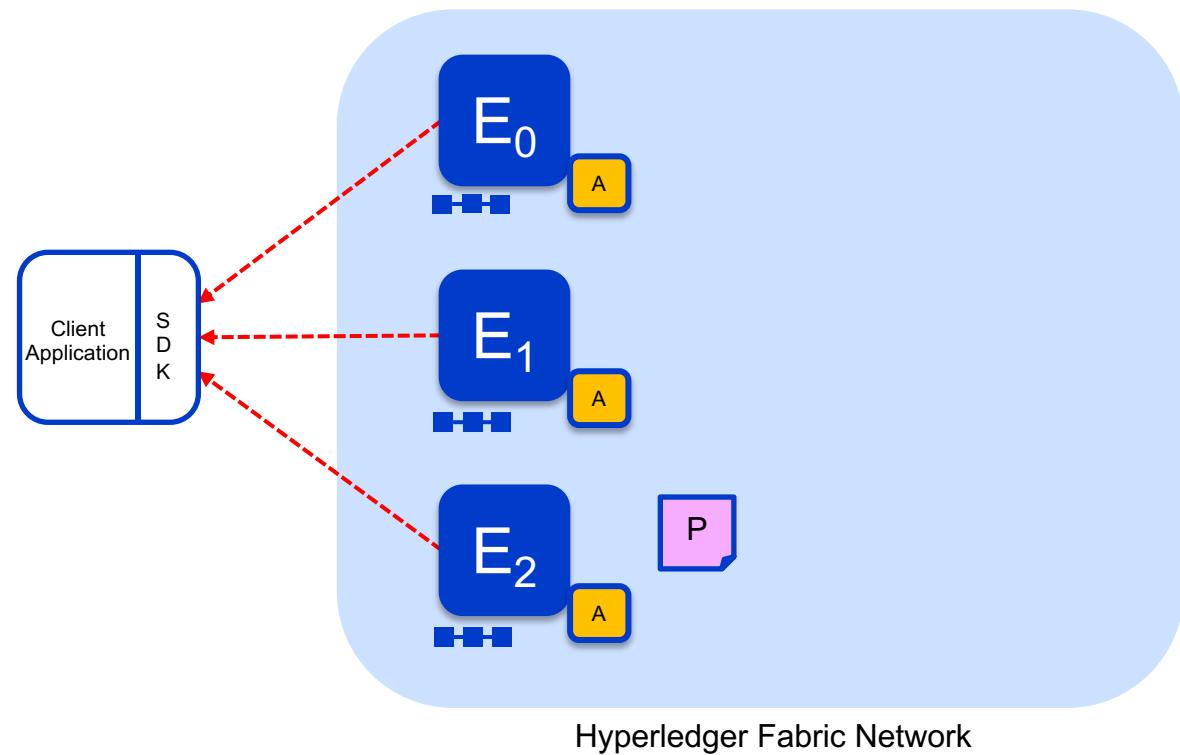
(This information will be checked much later in the consensus process)

Key:

Endorser		Ledger
Committing Peer		Application
Ordering Node		
Smart Contract (Chaincode)		Endorsement Policy



Sample transaction: Step 3/7 – Proposal Response



Application receives responses

RW sets are asynchronously returned to application

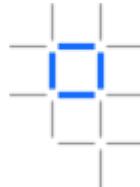
The RW sets are signed by each endorser, and also includes each record version number

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

Key:

Endorser			Ledger
Committing Peer			Application
Ordering Node			
Smart Contract (Chaincode)			Endorsement Policy

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

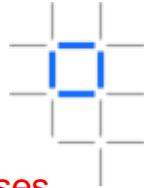
random()

getExchangeRate()

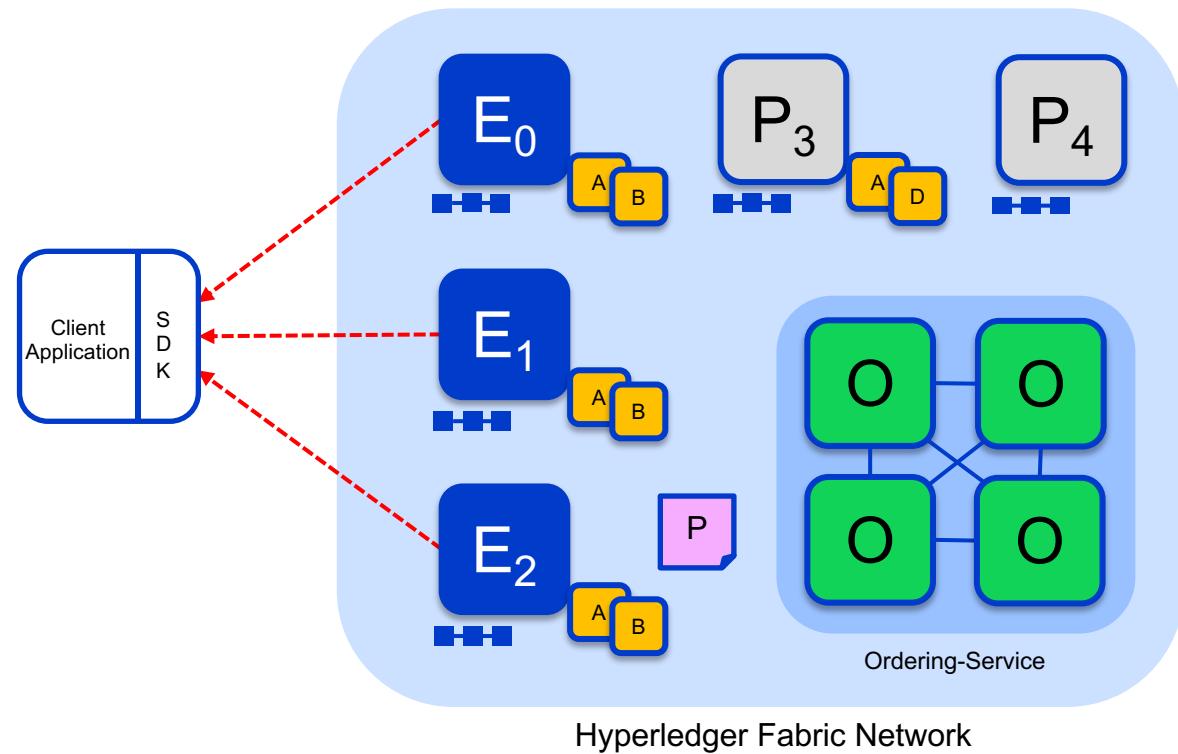
getDateTime()

getTemperature()

incrementValue
inExternalSystem(...)



Sample transaction: Step 3/7 – Proposal Response



Application receives responses

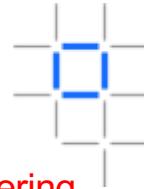
RW sets are asynchronously returned to application

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

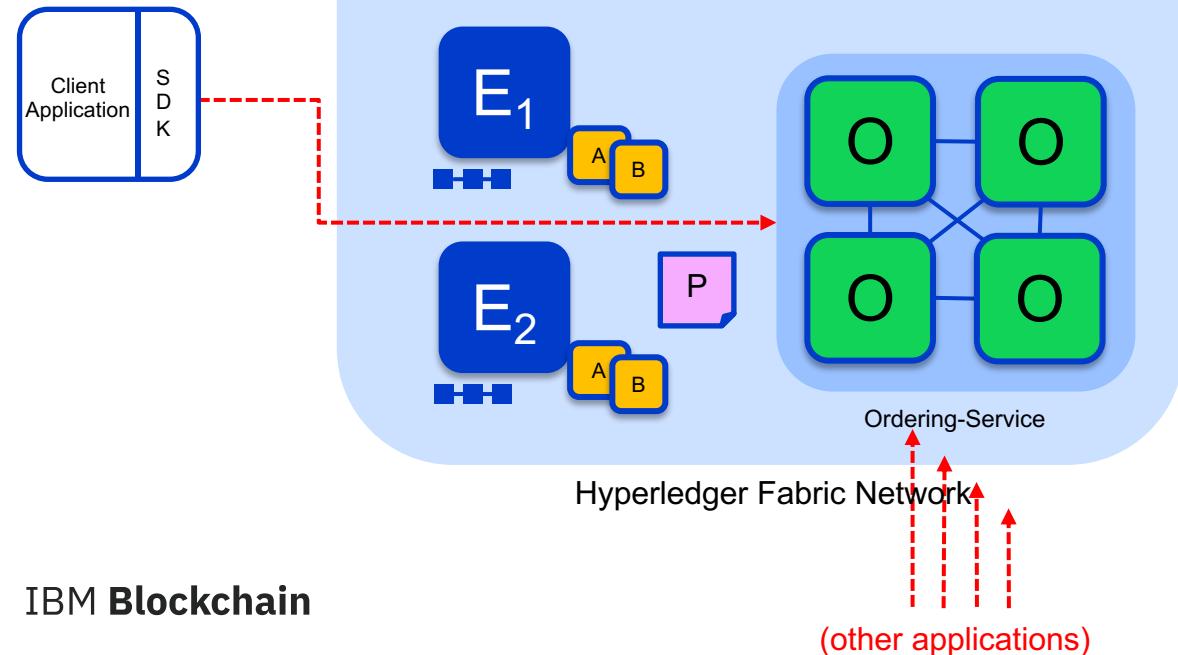


Sample transaction: Step 4/7 – Order Transaction

Responses submitted for ordering

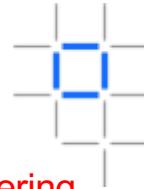
Application submits responses as a transaction to be ordered.

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

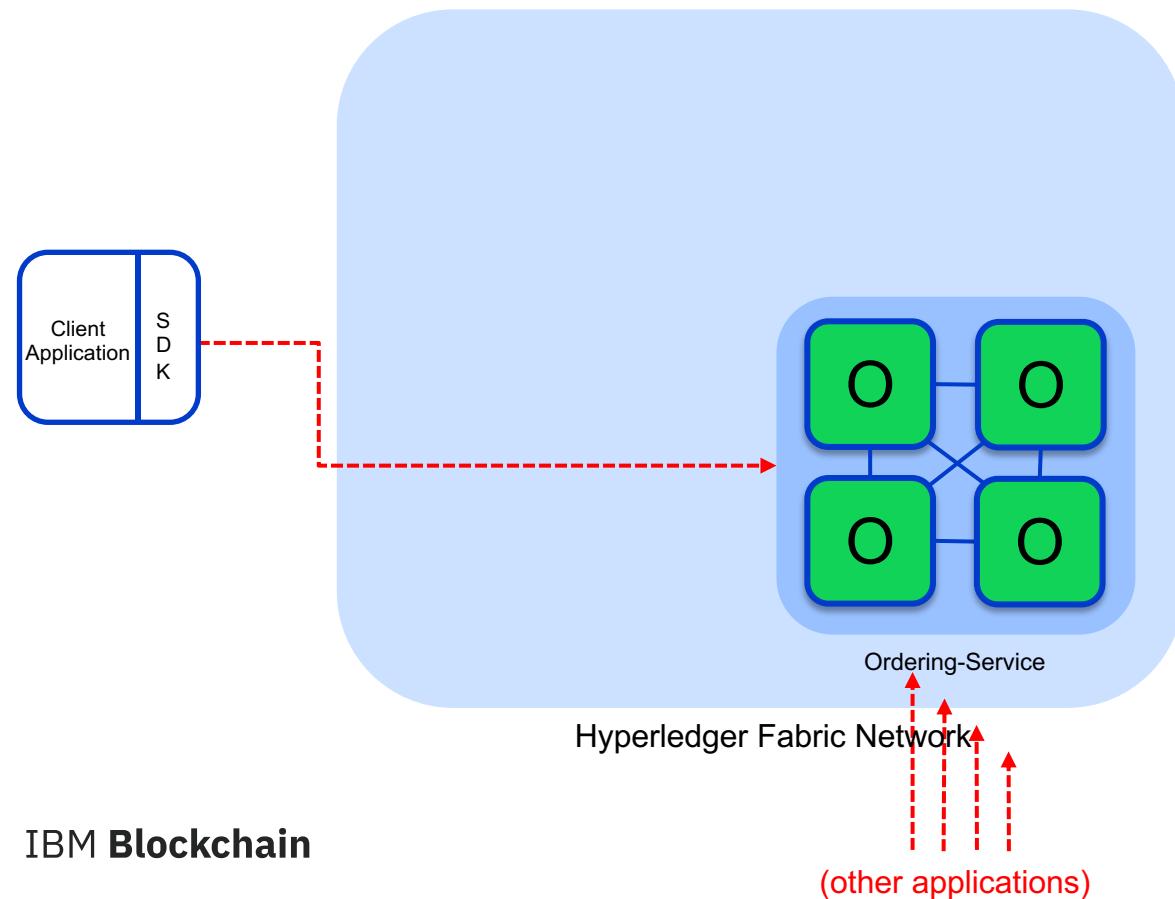


Key:

Endorser		Ledger
Committing Peer		Application
Ordering Node		
Smart Contract (Chaincode)		Endorsement Policy



Sample transaction: Step 4/7 – Order Transaction



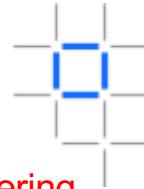
Responses submitted for ordering

Application submits responses as a transaction to be ordered.

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

Key:

Endorser		Ledger
Committing Peer		Application
Ordering Node		
Smart Contract (Chaincode)		Endorsement Policy

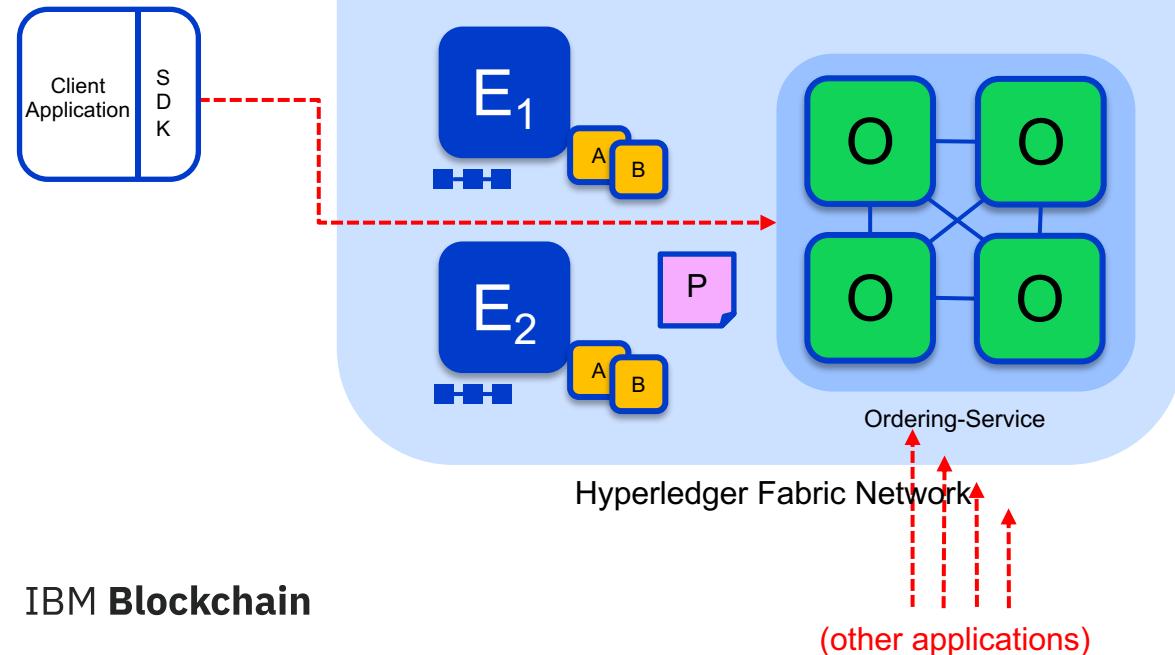


Sample transaction: Step 4/7 – Order Transaction

Responses submitted for ordering

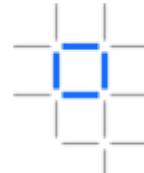
Application submits responses as a transaction to be ordered.

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

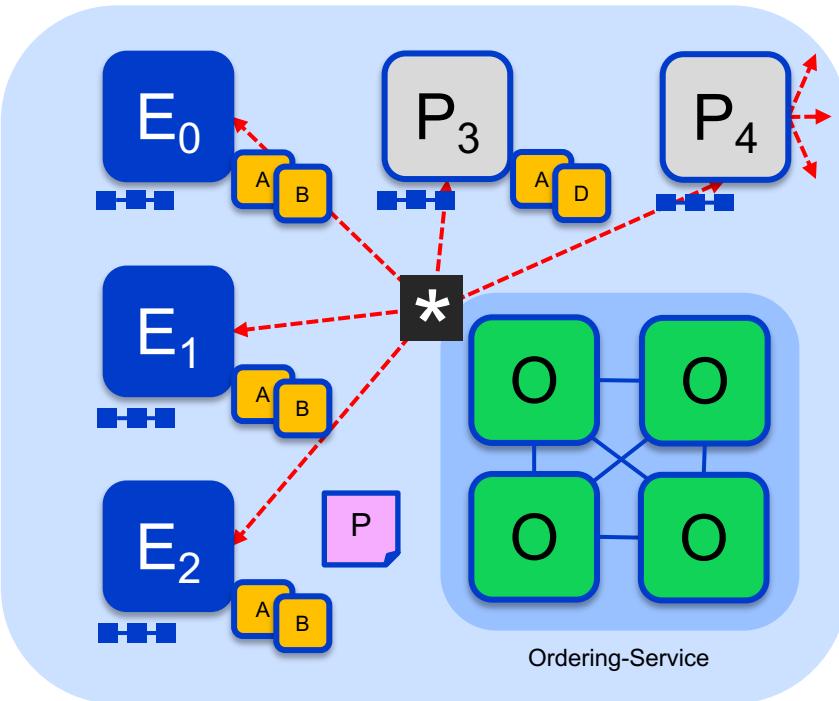


Key:

Endorser		Ledger
Committing Peer		Application
Ordering Node		
Smart Contract (Chaincode)		Endorsement Policy



Sample transaction: Step 5/7 – Deliver Transaction



Orderer delivers to committing peers

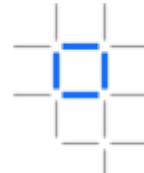
Ordering service collects transactions into proposed blocks for distribution to committing peers. Peers can deliver to other peers in a hierarchy (not shown)

Different ordering algorithms available:

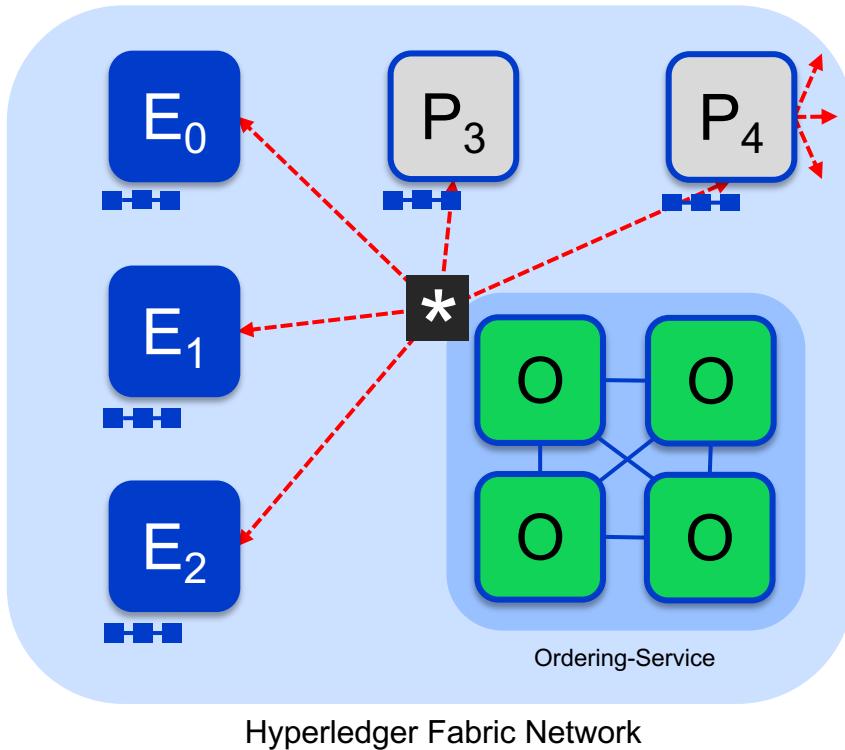
- SOLO (Single node, development)
- Kafka (Crash fault tolerance)

Key:

Endorser		Ledger
Committing Peer		Application
Ordering Node		
Smart Contract (Chaincode)		Endorsement Policy



Sample transaction: Step 5/7 – Deliver Transaction



Orderer delivers to committing peers

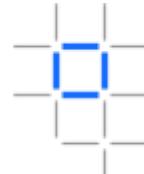
Ordering service collects transactions into proposed blocks for distribution to committing peers. Peers can deliver to other peers in a hierarchy (not shown)

Different ordering algorithms available:

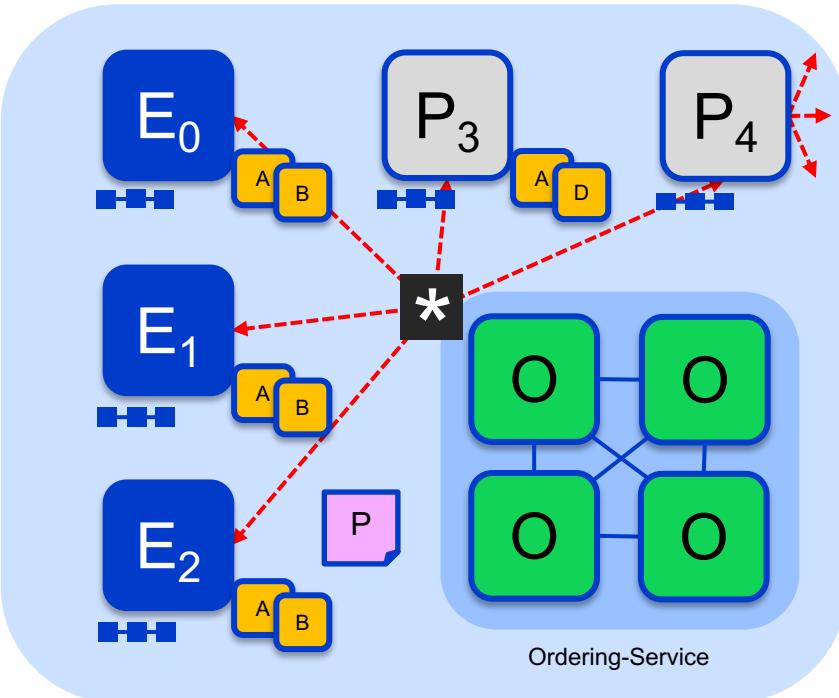
- SOLO (Single node, development)
- Kafka (Crash fault tolerance)

Key:

Endorser		Ledger
Committing Peer		Application
Ordering Node		
Smart Contract (Chaincode)		Endorsement Policy



Sample transaction: Step 5/7 – Deliver Transaction



Orderer delivers to committing peers

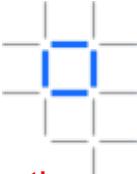
Ordering service collects transactions into proposed blocks for distribution to committing peers. Peers can deliver to other peers in a hierarchy (not shown)

Different ordering algorithms available:

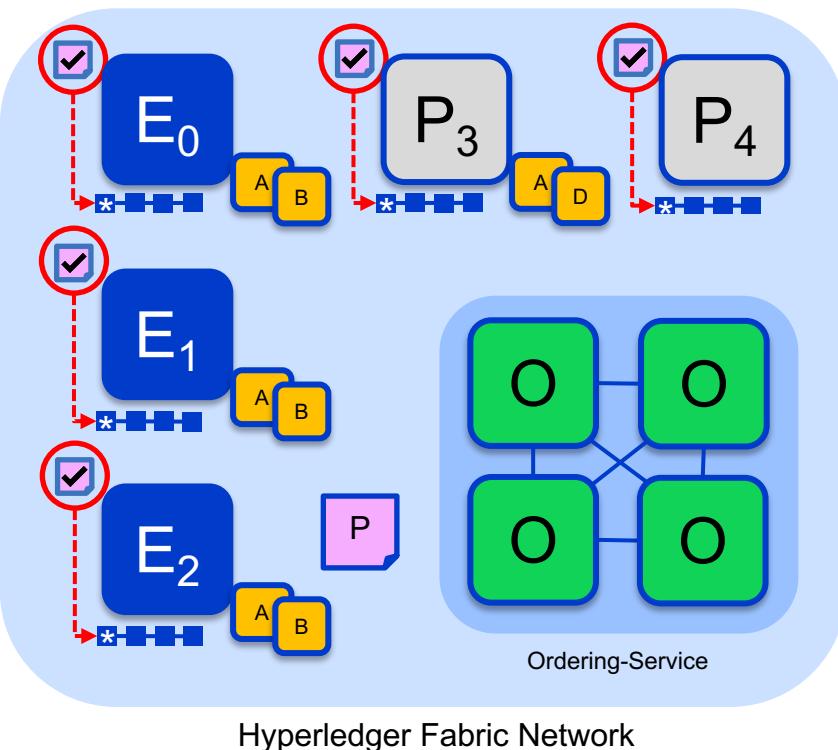
- SOLO (Single node, development)
- Kafka (Crash fault tolerance)

Key:

Endorser		Ledger
Committing Peer		Application
Ordering Node		
Smart Contract (Chaincode)		Endorsement Policy



Sample transaction: Step 6/7 – Validate Transaction



Committing peers validate transactions

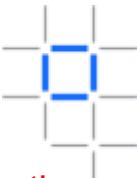
Every committing peer validates against the endorsement policy. Also check RW sets are still valid for current world state

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

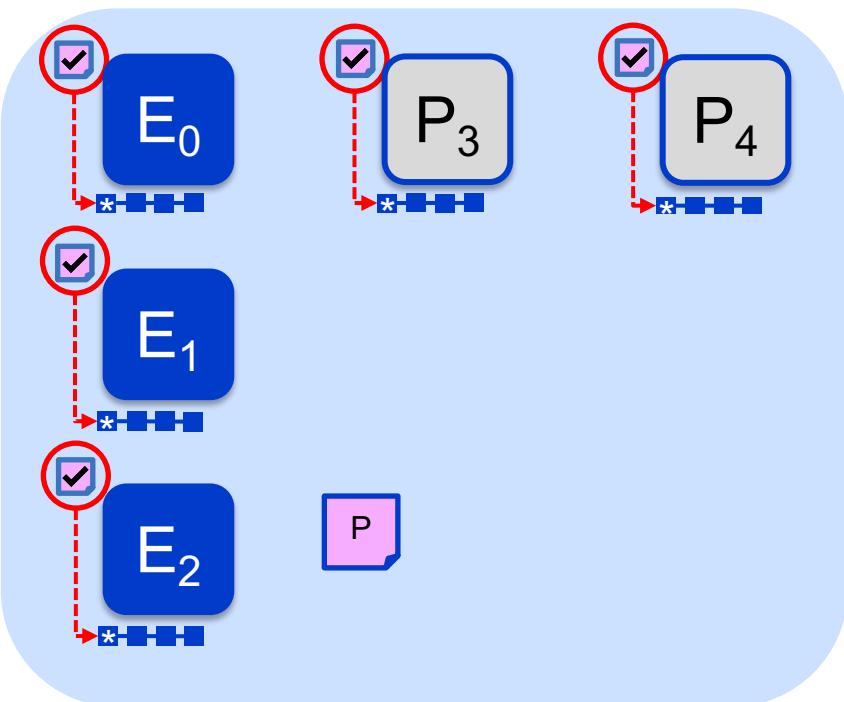
Invalid transactions are also retained on the ledger but do not update world state

Key:

Endorser		Ledger
Committing Peer		Application
Ordering Node		
Smart Contract (Chaincode)		Endorsement Policy



Sample transaction: Step 6/7 – Validate Transaction



Committing peers validate transactions

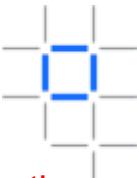
Every committing peer validates against the endorsement policy. Also check RW sets are still valid for current world state

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

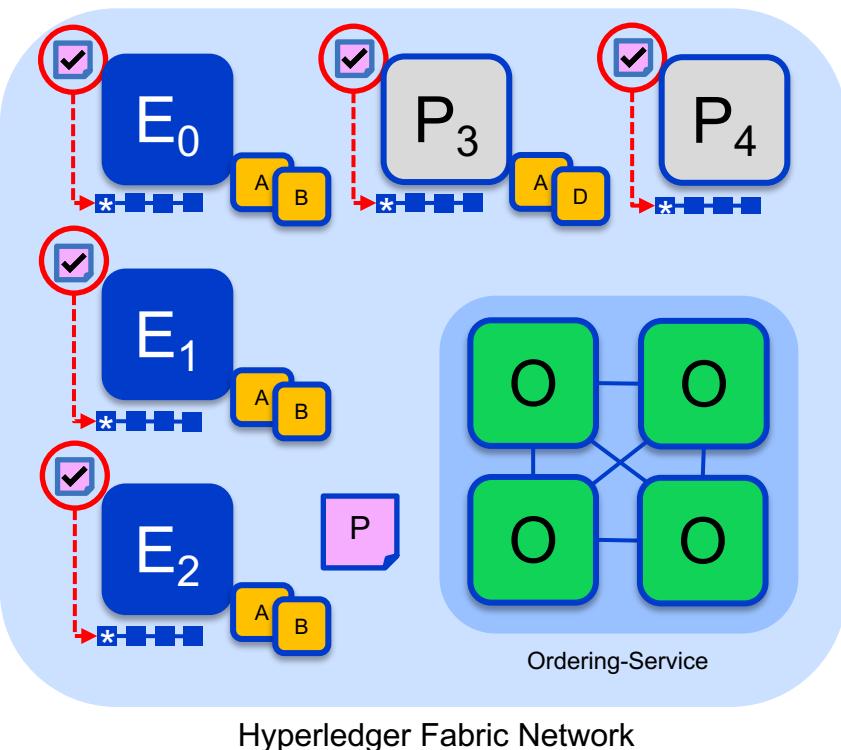
Invalid transactions are also retained on the ledger but do not update world state

Key:

Endorser		Ledger
Committing Peer		Application
Ordering Node		
Smart Contract (Chaincode)		Endorsement Policy



Sample transaction: Step 6/7 – Validate Transaction



Committing peers validate transactions

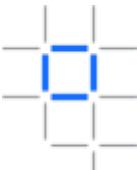
Every committing peer validates against the endorsement policy. Also check RW sets are still valid for current world state

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:

Endorser		Ledger
Committing Peer		Application
Ordering Node		
Smart Contract (Chaincode)		Endorsement Policy

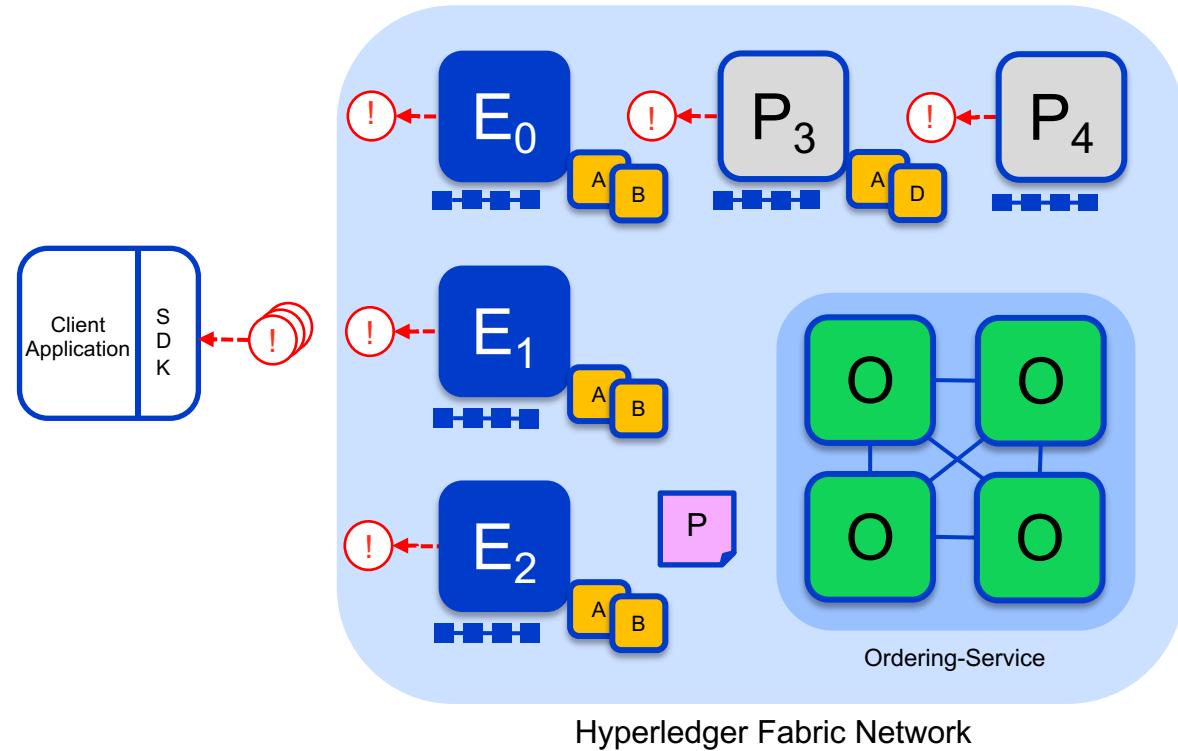


Sample transaction: Step 7/7 – Notify Transaction

Committing peers notify applications

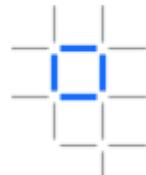
Applications can register to be notified when transactions succeed or fail, and when blocks are added to the ledger

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

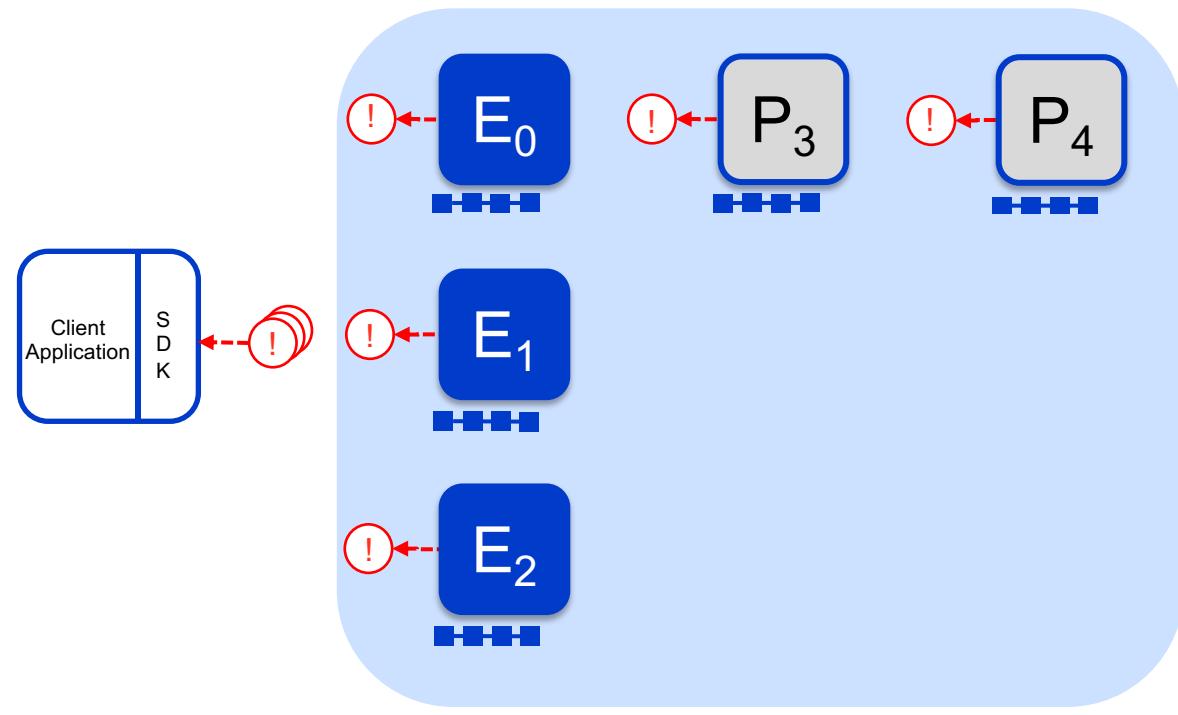


Key:

Endorser		Ledger
Committing Peer		Application
Ordering Node		
Smart Contract (Chaincode)		Endorsement Policy



Sample transaction: Step 7/7 – Notify Transaction



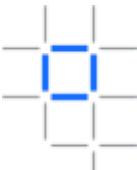
Committing peers notify applications

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

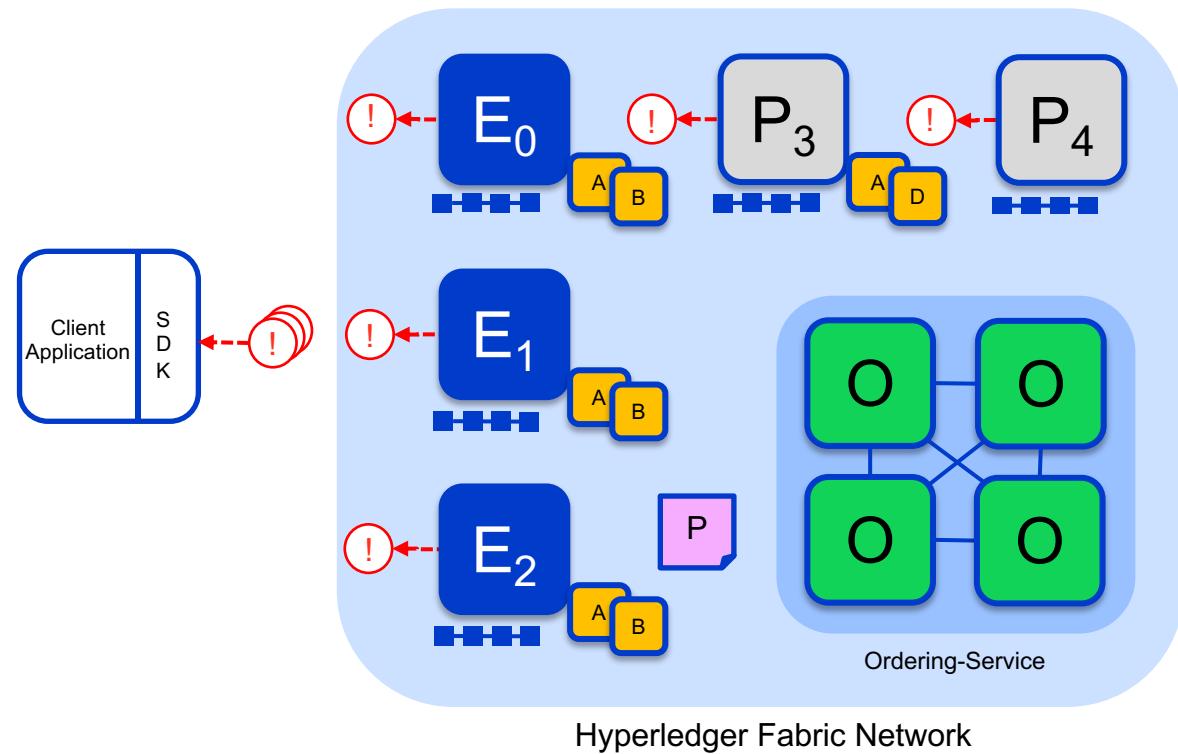
Applications will be notified by each peer to which they are connected

Key:

Endorser		Ledger
Committing Peer		Application
Ordering Node		
Smart Contract (Chaincode)		Endorsement Policy



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:

Endorser		Ledger
Committing Peer		Application
Ordering Node		
Smart Contract (Chaincode)		Endorsement Policy

Thank you

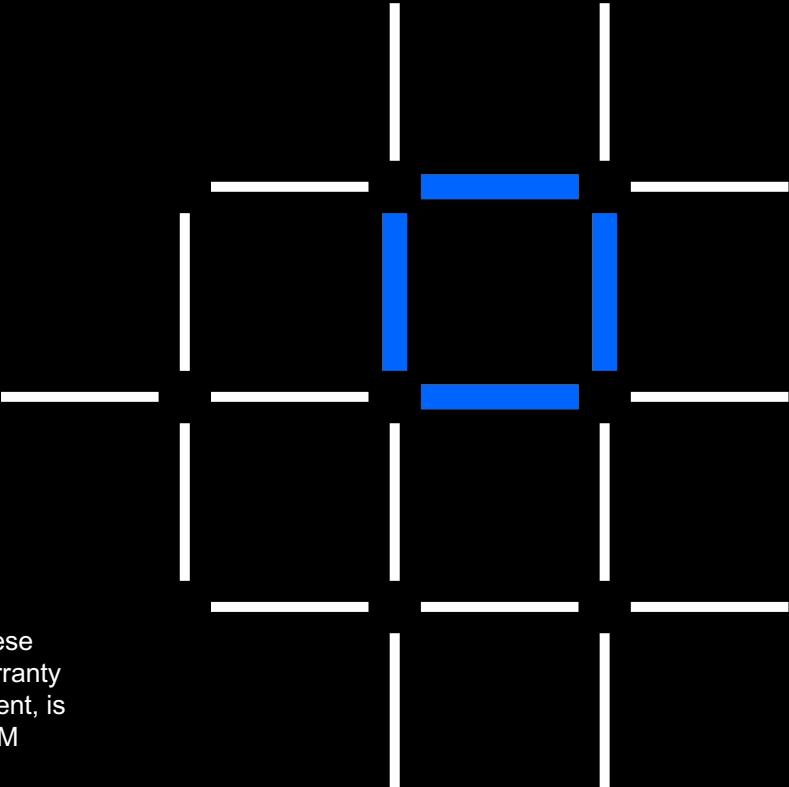
Barry Silliman
silliman@us.ibm.com

IBM Blockchain

www.ibm.com/blockchain

developer.ibm.com/blockchain

www.hyperledger.org



© Copyright IBM Corporation 2017. All rights reserved. The information contained in these materials is provided for informational purposes only, and is provided AS IS without warranty of any kind, express or implied. Any statement of direction represents IBM's current intent, is subject to change or withdrawal, and represents only goals and objectives. IBM, the IBM logo, and other IBM products and services are trademarks of the International Business Machines Corporation, in the United States, other countries or both. Other company, product, or service names may be trademarks or service marks of others.



IBM