Blockchain Explored and Architected

A Technical Deep-Dive on Hyperledger Fabric V1

IBM **Blockchain**

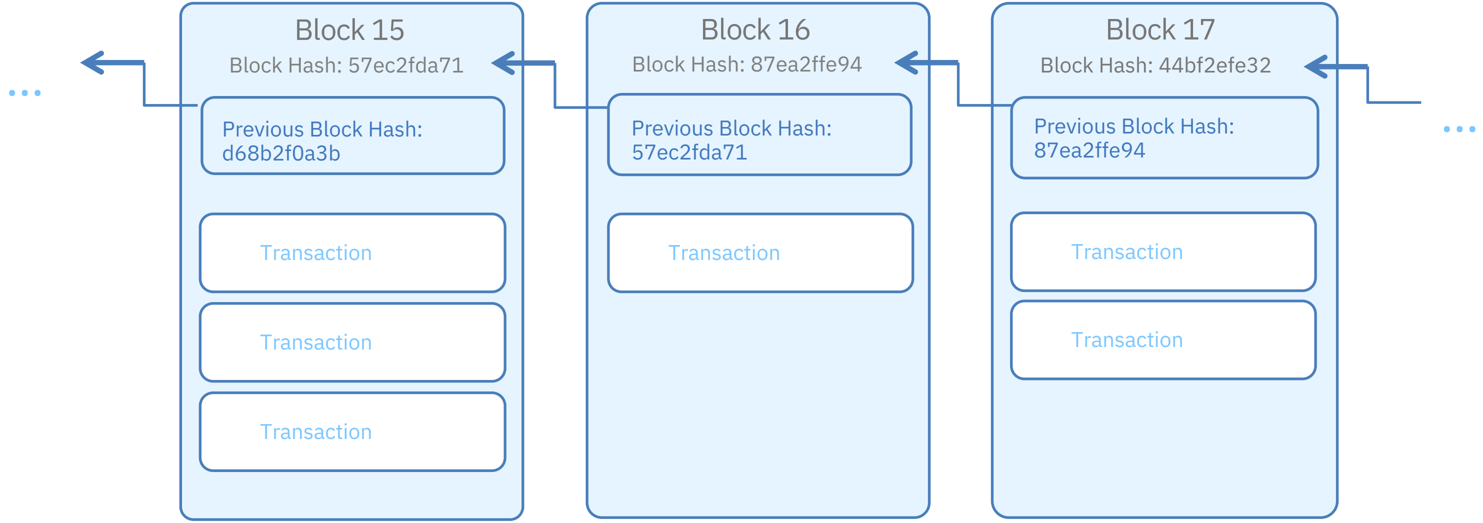
Blockchain education series

Explained Solutions Composed Architected **Explored** Next Steps

V4.0, 23 August 2017

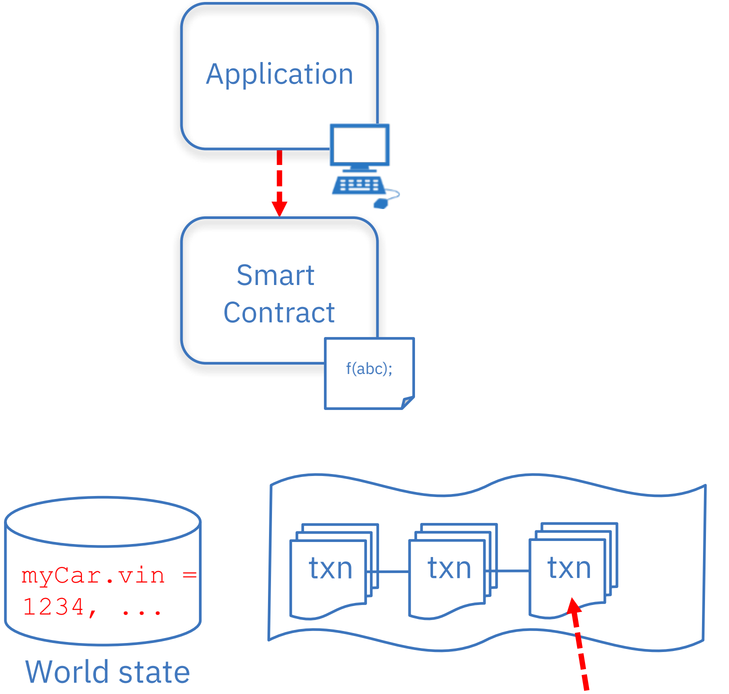
Blockchain Concepts

Block detail (simplified)



- A blockchain is made up of a series of blocks with new blocks always added to the end
- Each block contains zero or more transactions and some additional metadata
- Blocks achieve immutability by including the result of a hash function of the previous block
- The first block is known as the “genesis” block

Smart Contracts



“Invoke, myContract, setOwner, myCar, Matt”

Transaction input - sent from application

```
invoke(myContract, setOwner,
        myCar, Matt)
…
```

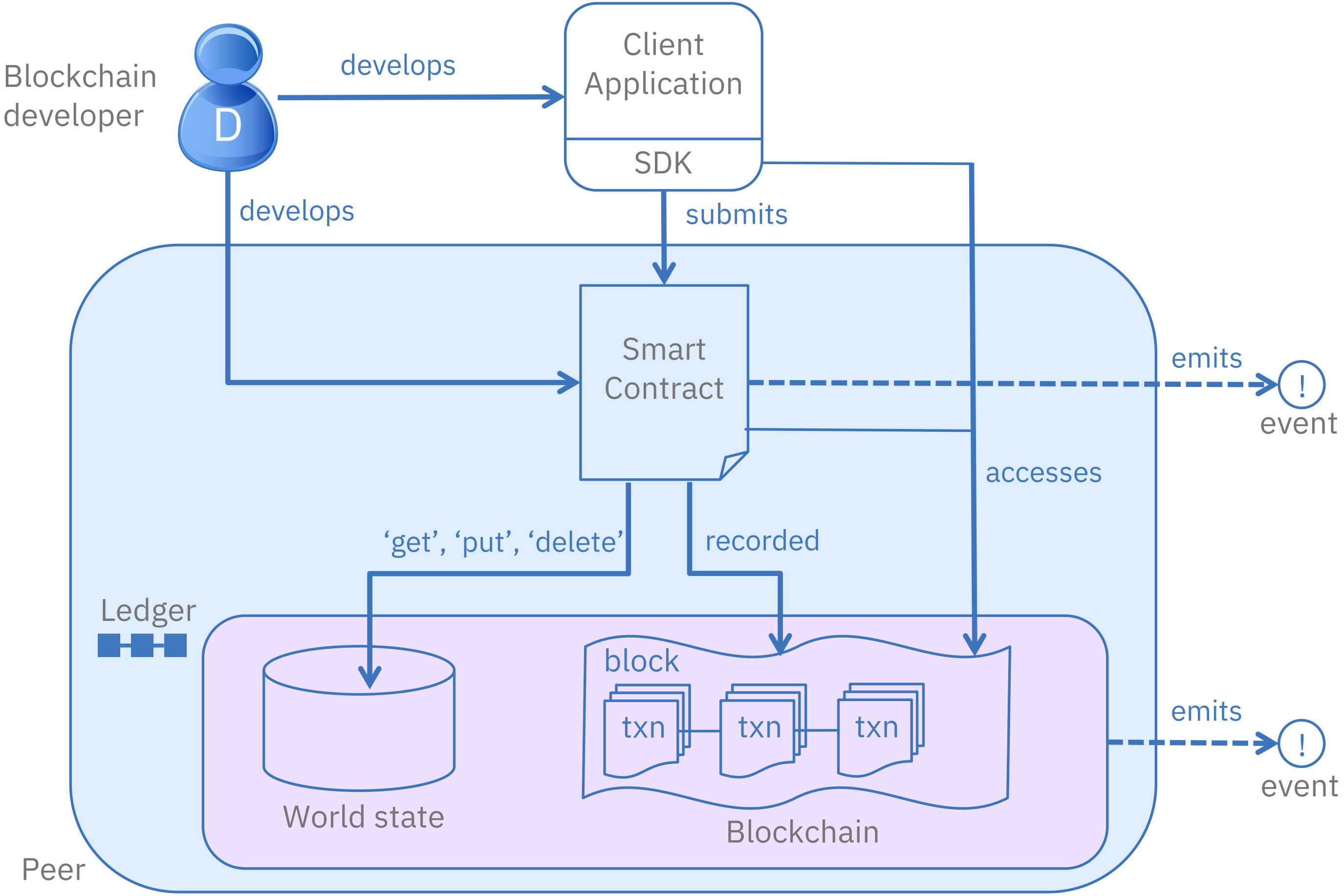
Smart contract implementation

```
setOwner(Car, newOwner) {
   set Car.owner = newOwner
}
```

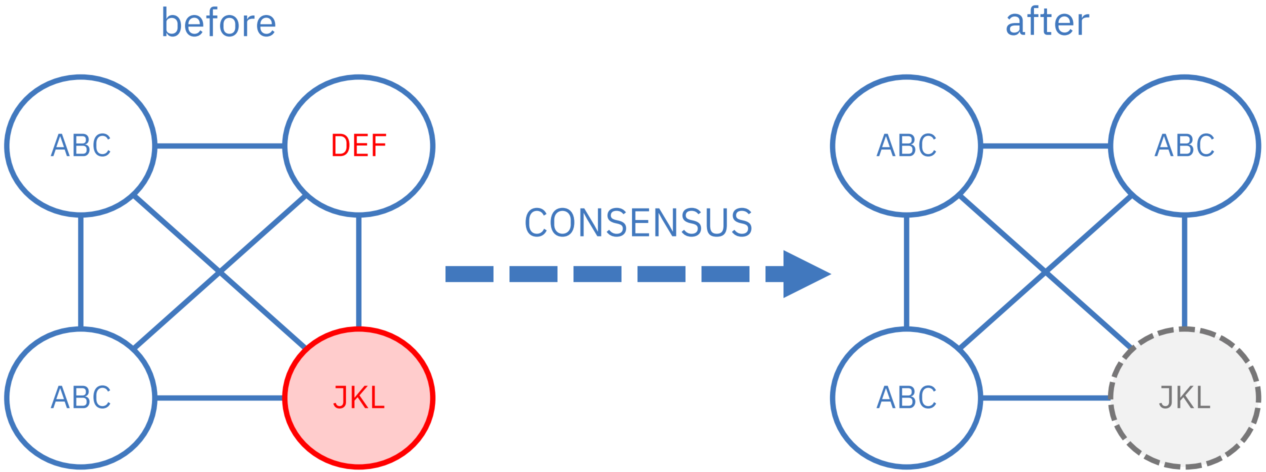
World state: new contents

```
myCar.vin = 1234
myCar.owner = Matt
myCar.make = Audi
…
```

Blockchain and Smart Contracts Put Together



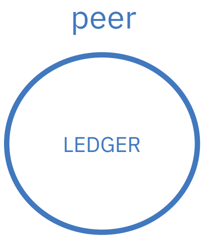
Consensus: The process of maintaining a consistent ledger



Keep all peers up-to-date

Fix any peers in error

Ignoring all malicious nodes



Public vs. private blockchains

Public blockchains

- For example, Bitcoin
- Transactions are viewable by anyone
- Participant identity is more difficult to control

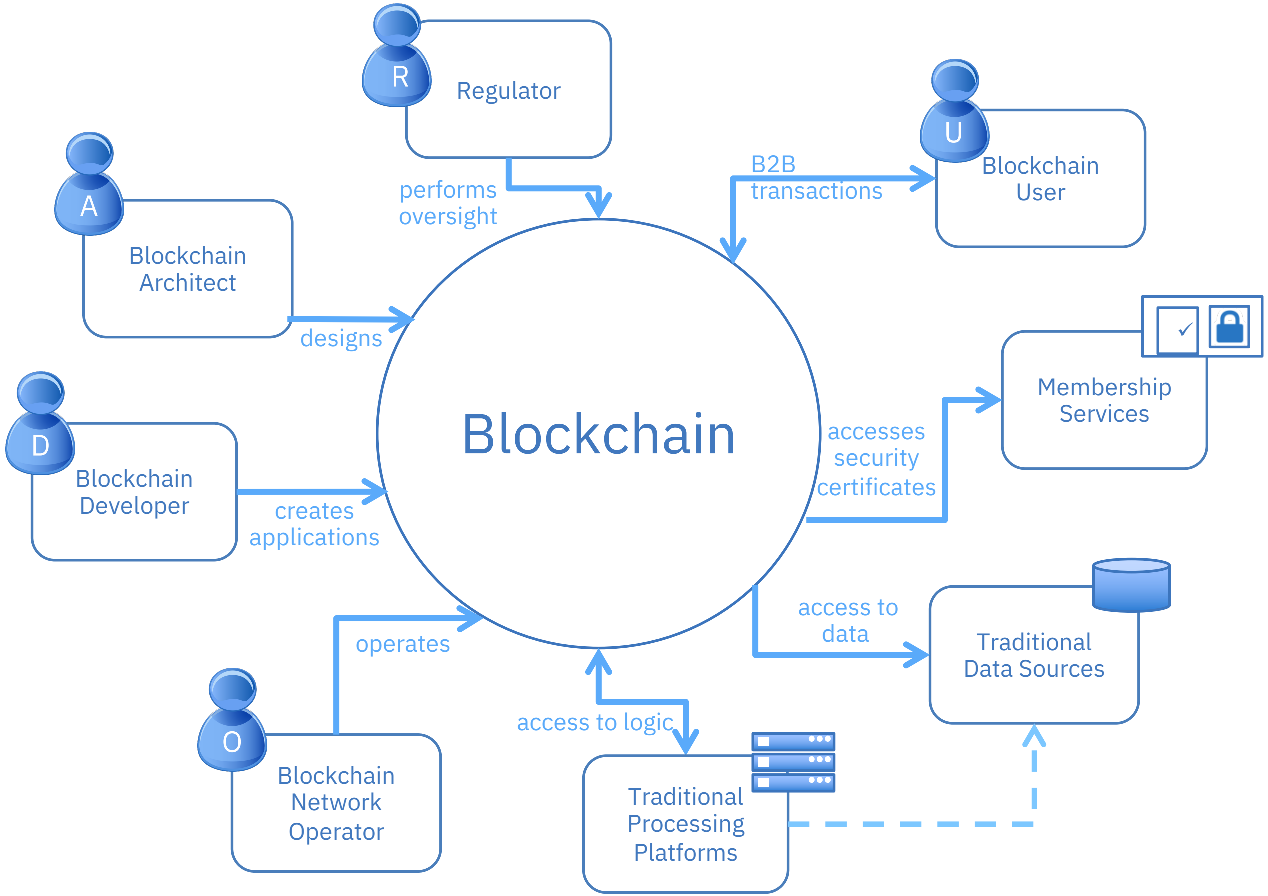
Private blockchains

- For example, Hyperledger Fabric
- Network members are known but transactions are secret

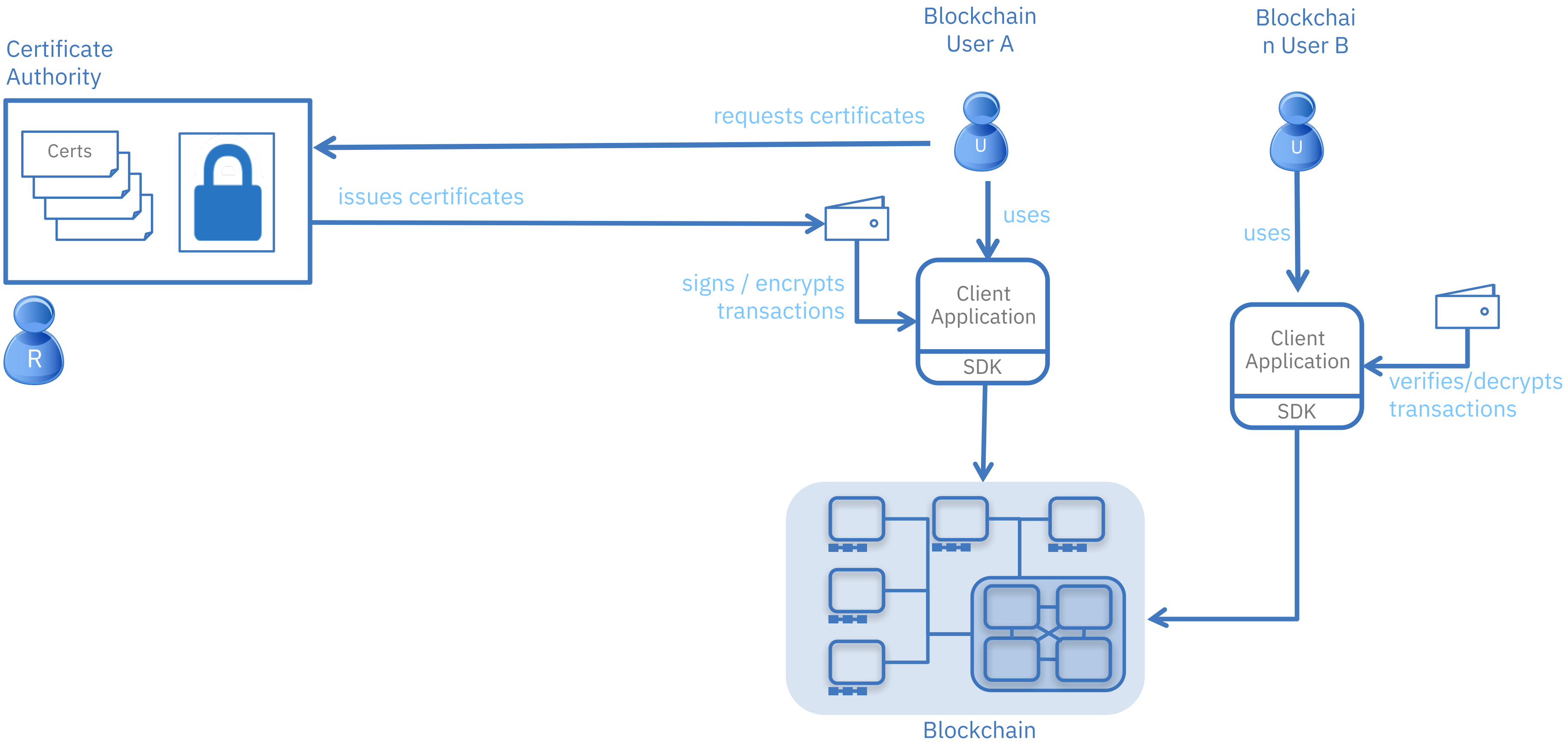
- Some use-cases require anonymity, others require privacy
  - Some may require a mixture of the two, depending on the characteristics of each participant

- Most business use-cases require private, permissioned blockchains
  - Network members know who they're dealing with (required for KYC, AML etc.)
  - Transactions are (usually) confidential between the participants concerned
  - Membership is controlled

Actors in a private blockchain solution



Privacy



Certificate
Authority

Certs

R

requests certificates

issues certificates

Blockchain
User A

U

uses

signs / encrypts
transactions

Client
Application

SDK

Blockchai
n User B

U

uses

Client
Application

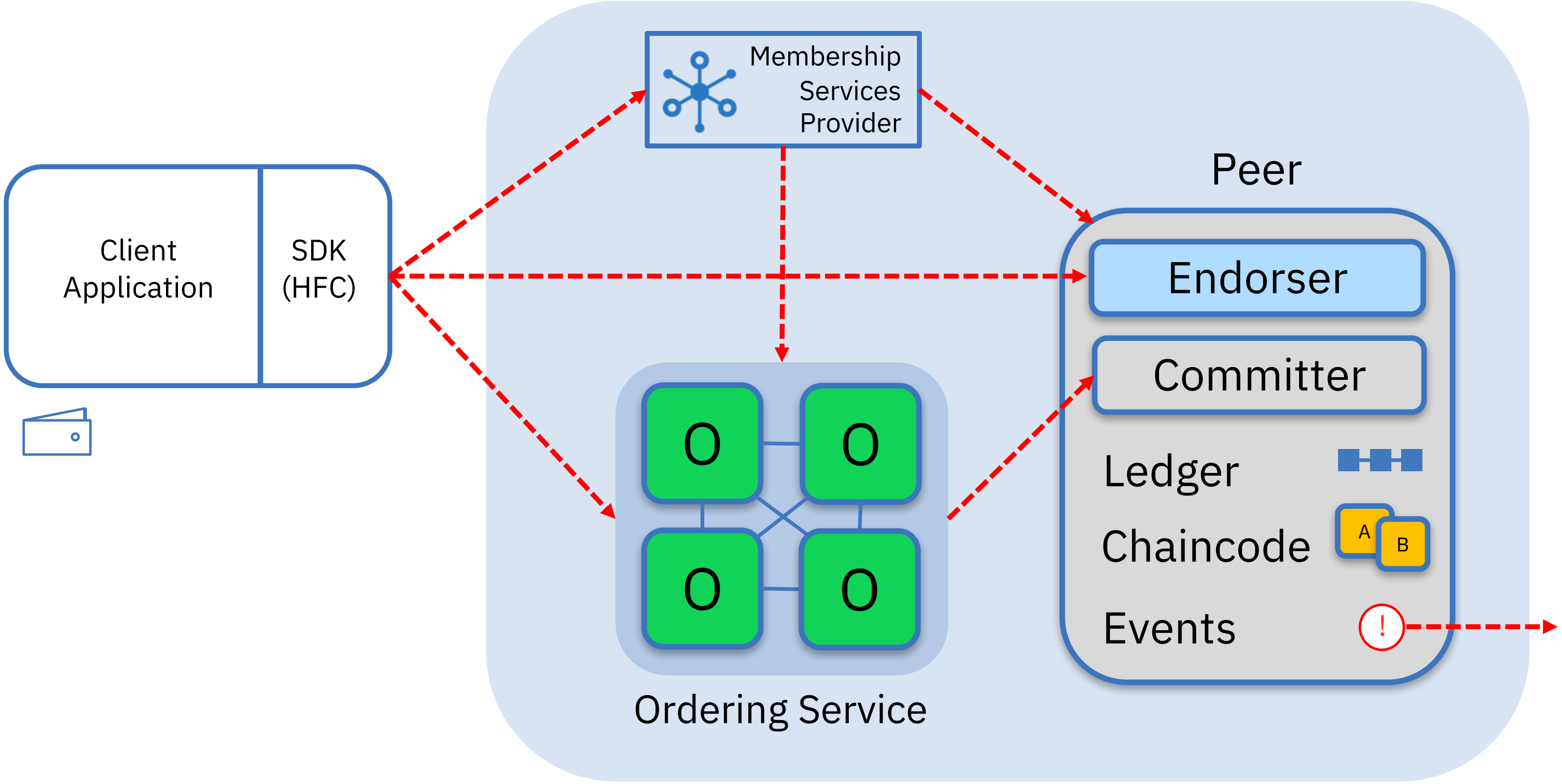
SDK

verifies/decrypts
transactions

Blockchain

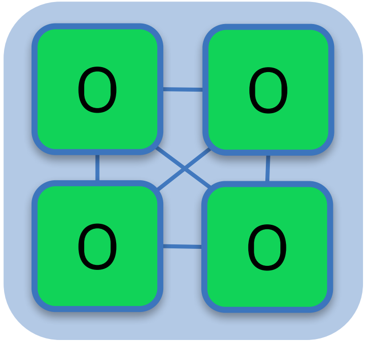
Hyperledger Fabric Architecture

Hyperledger Fabric V1 Architecture



Ordering Service

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



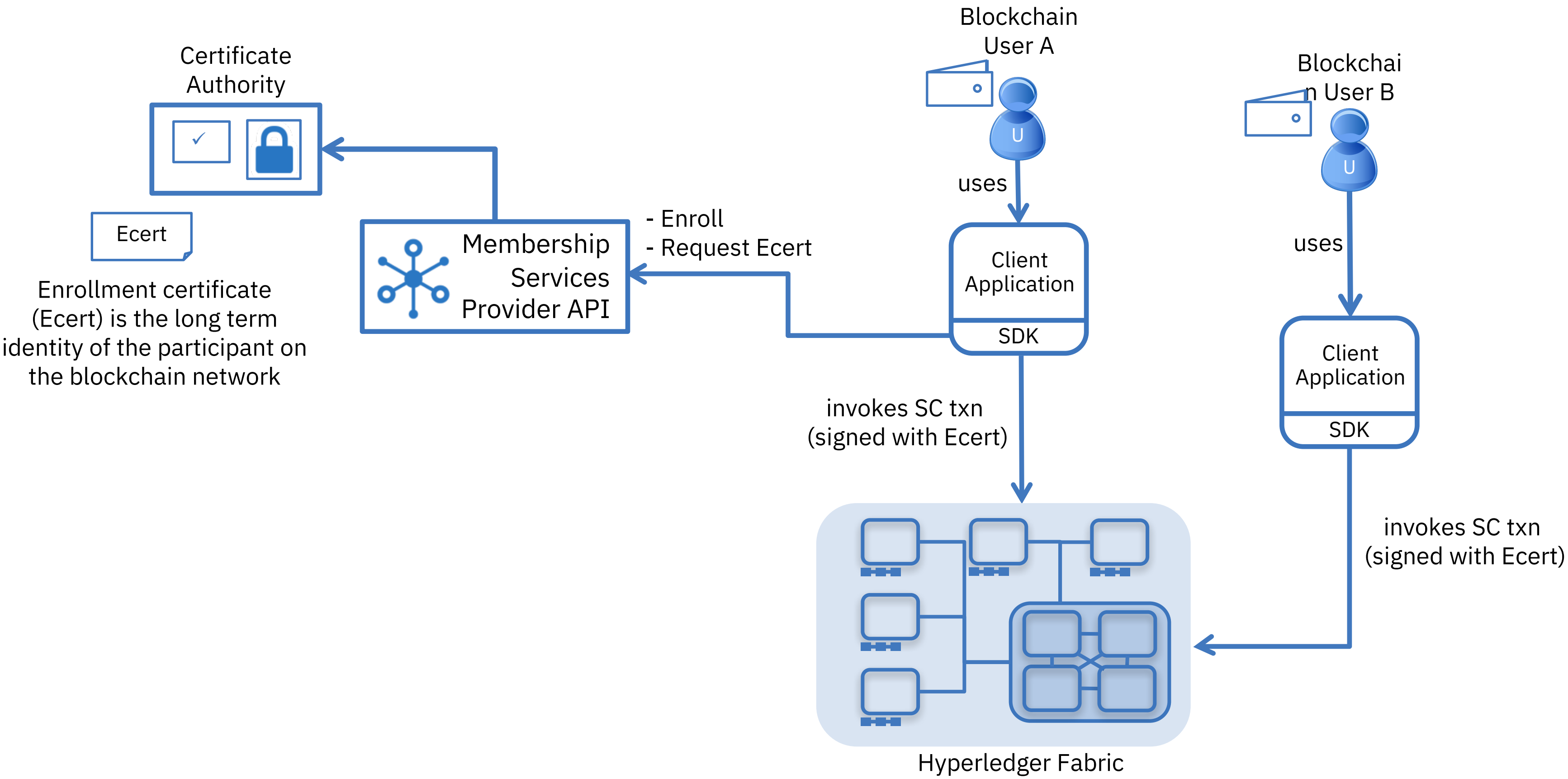
Ordering-Service

Different configuration options for the ordering service include:

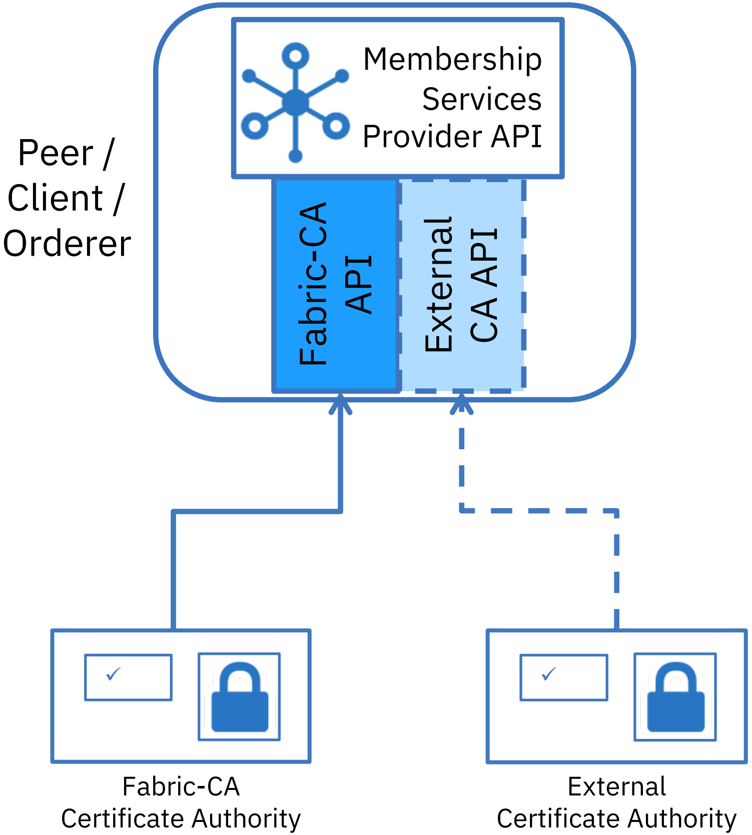
– SOLO
  - Single node for development

– Kafka : Crash fault tolerant consensus
  - 3 nodes minimum
  - Odd number of nodes recommended

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 SC txn (signed with Ecert)
Blockchai n User B
U
uses
Client Application
SDK
invokes SC txn (signed with Ecert)
Hyperledger Fabric

Membership Services Provider API



Membership Services Provider API

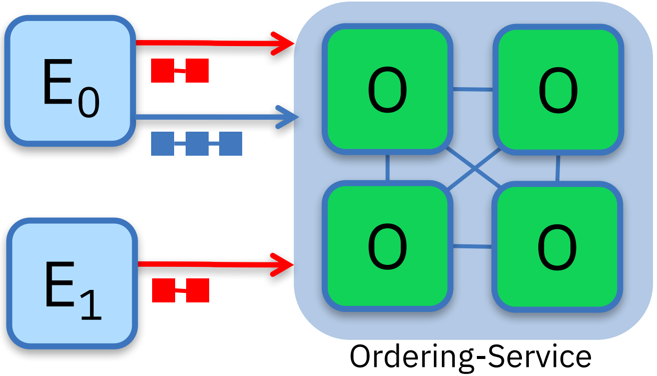
- Pluggable interface supporting a range of credential architectures
- Default implementation calls Fabric-CA.
- Governs identity for Peers and Users.
- Provides:
  - User authentication
  - User credential validation
  - Signature generation and verification
  - Optional credential issuance
- Additional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

Nodes and roles

⬜	Committing Peer: Maintains ledger and state. Commits transactions. May hold smart contract (chaincode).
🟦	Endorsing Peer: Specialized committing peer that receives a transaction proposal for endorsement, responds granting or denying endorsement. Must hold smart contract
🟩	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. Does not hold ledger.

Channels

Separate channels isolate transactions on different ledgers

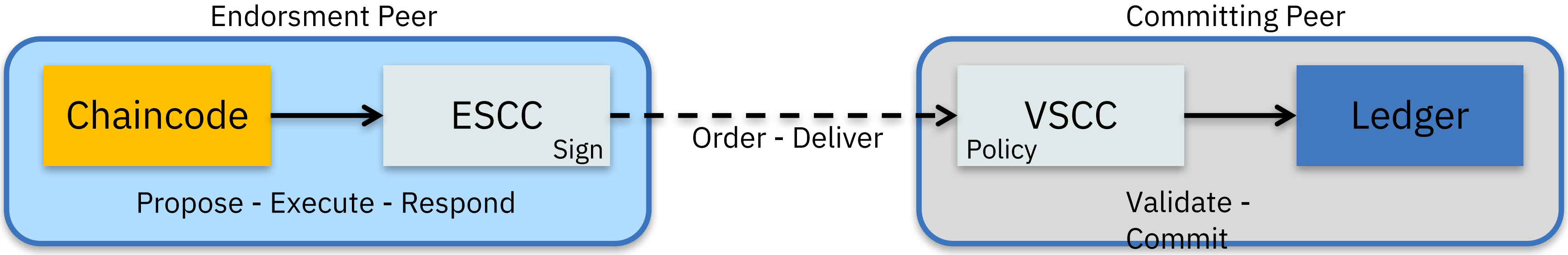


- Chaincode is installed on peers that need to access the worldstate
- Chaincode is instantiated on specific channels for specific peers
- Ledgers exist in the scope of a channel
  - Ledgers can be shared across an entire network of peers
  - Ledgers can be included only on a specific set of participants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

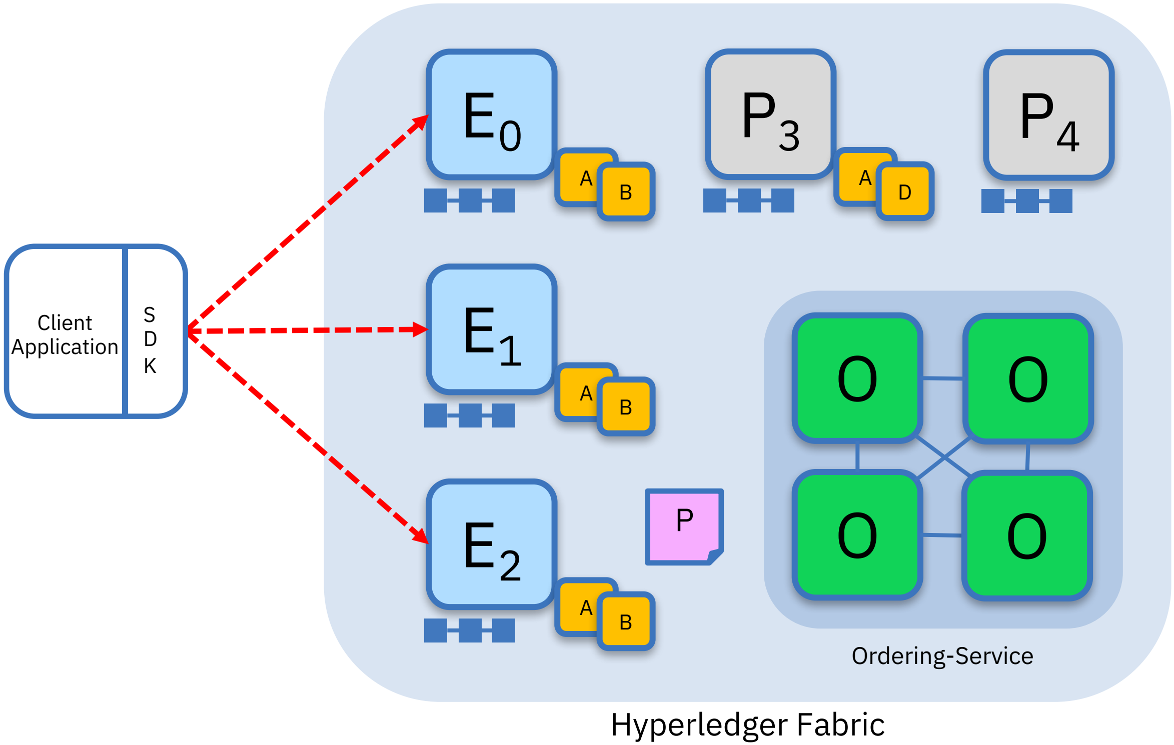
Endorsement Policies

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



Sample transaction: Step 1/7 – Propose transaction



Application proposes transaction

Endorsement policy:

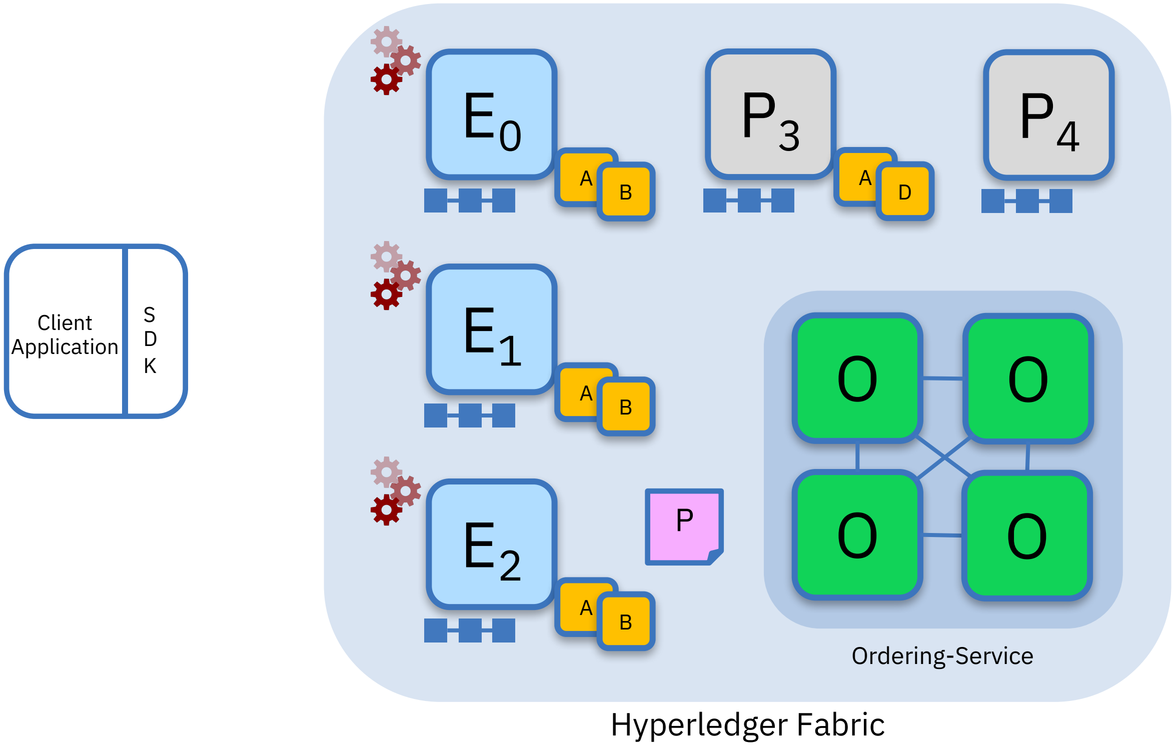
- “E_0, E_1 and E_2 must sign”
- (P_3, P_4 are not part of the policy)

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

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

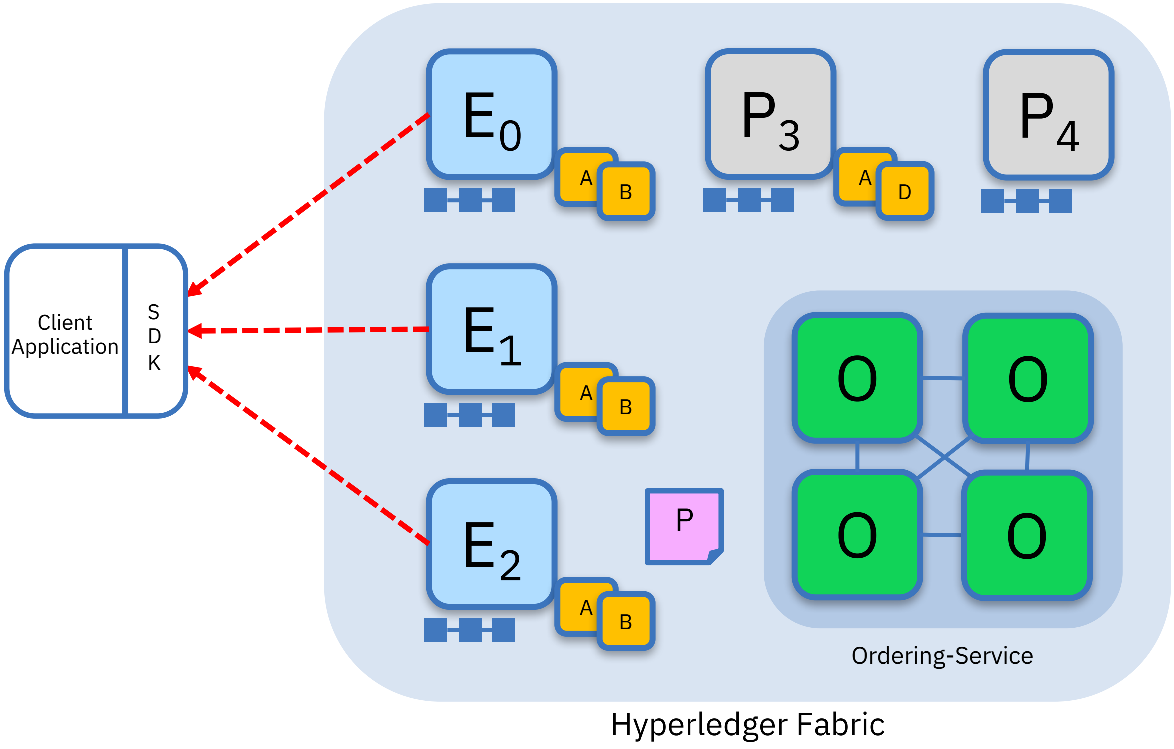
Each execution will capture the set of Read and Written data, called RW sets, which will now flow in the fabric.

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



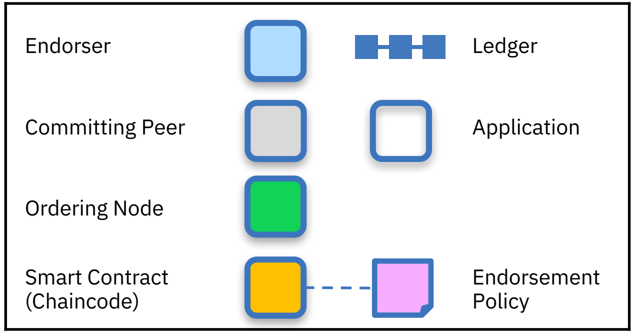
Application receives responses

RW sets are asynchronously returned to application

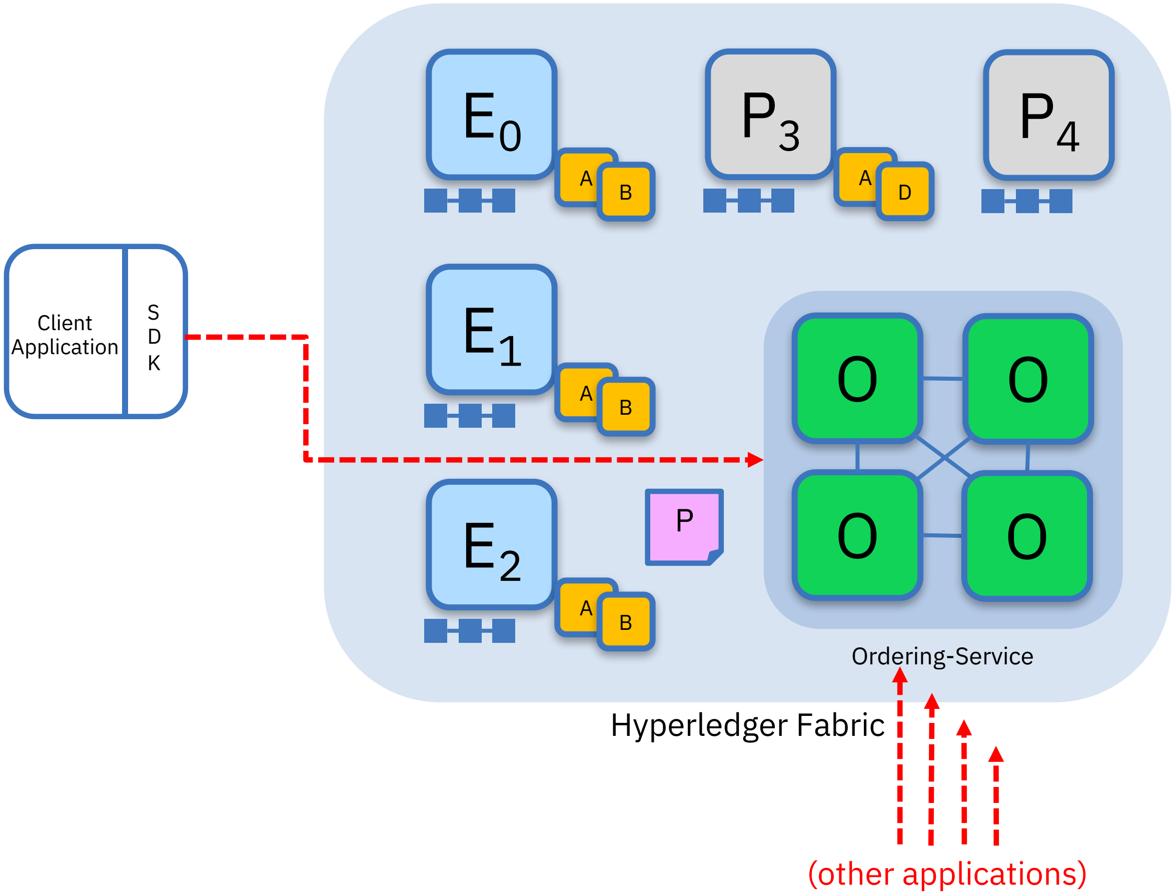
The RW sets are signed by each endorser, and also includes each record version number

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

Key:



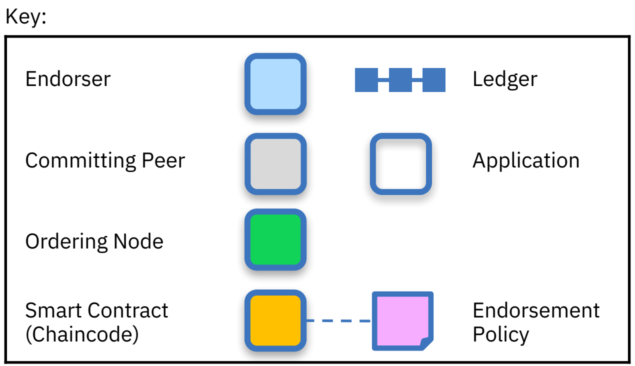
Sample transaction: Step 4/7 – Order Transaction



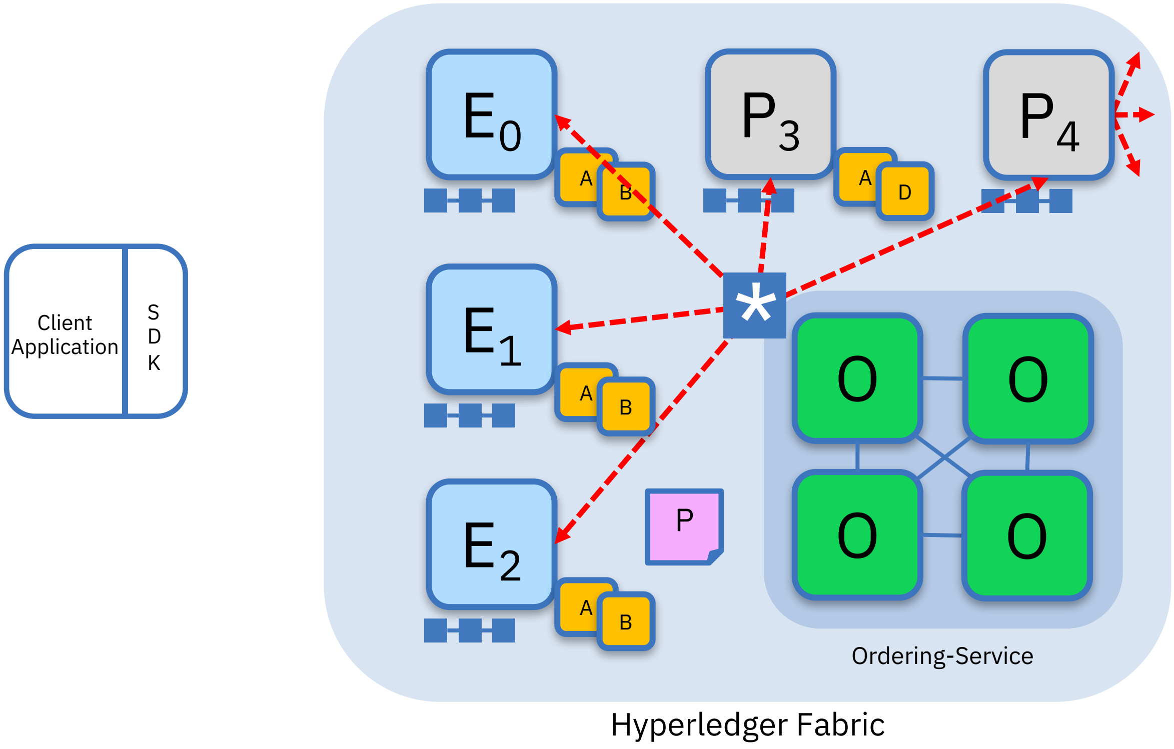
Application submits responses for ordering

Application submits responses as a transaction to be ordered.

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



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)

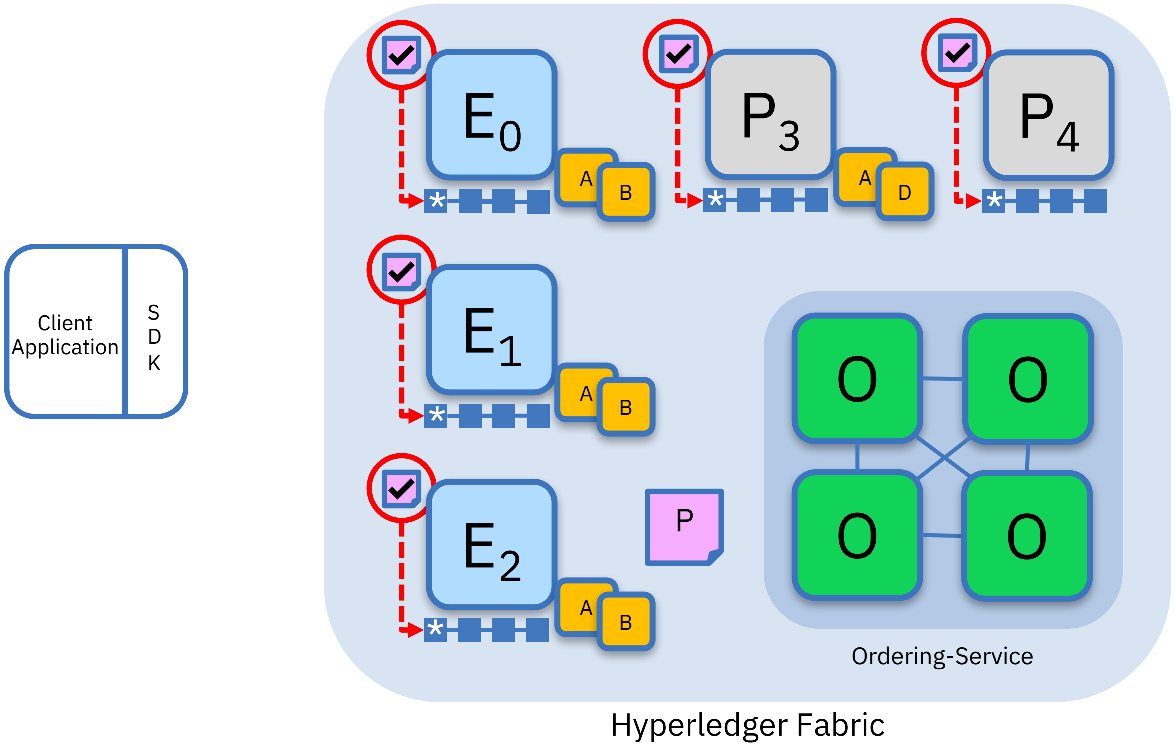
Different ordering algorithms available:

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

Key:

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

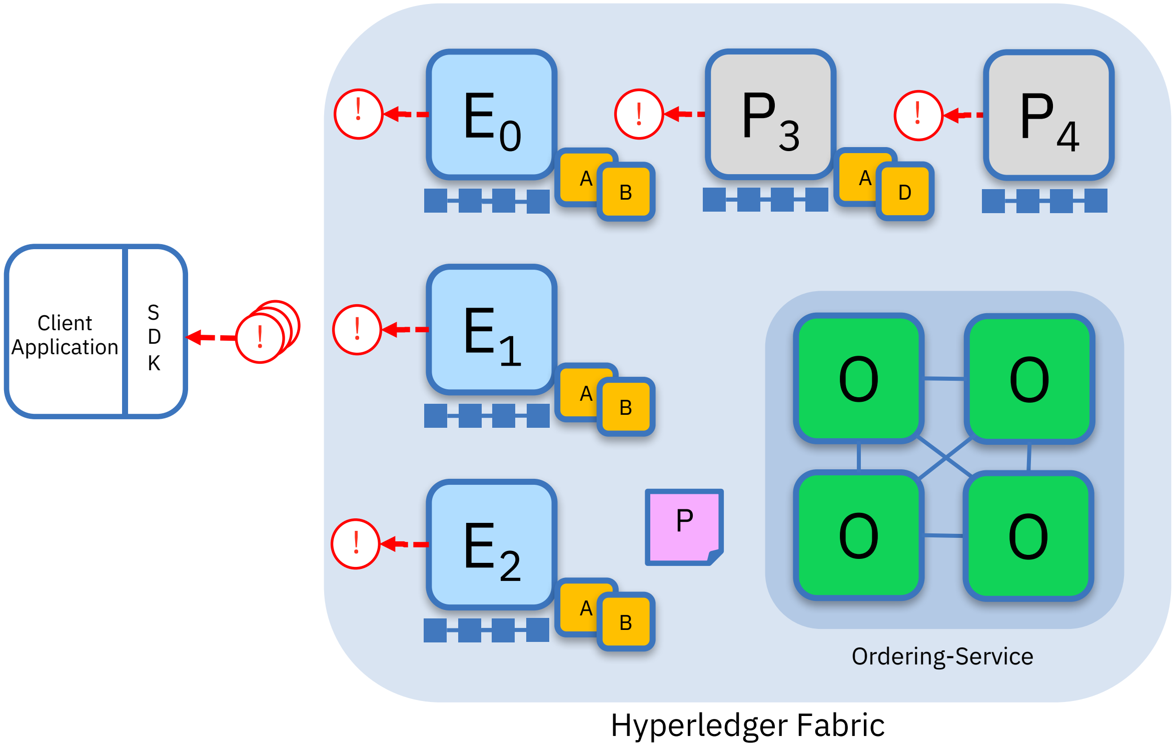
Validated transactions are applied to the world state and retained on the ledger

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

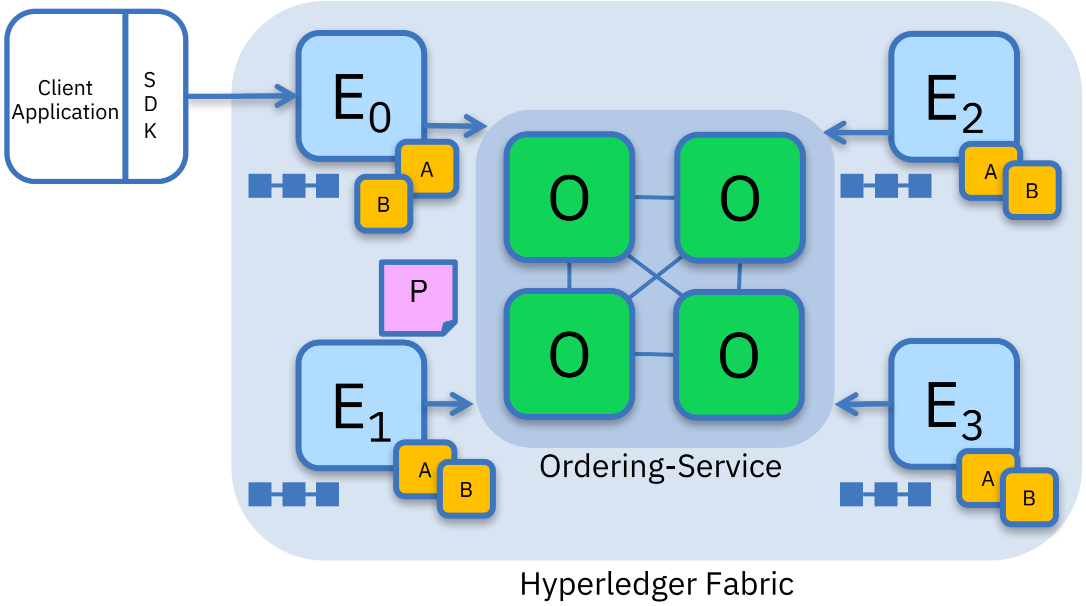
Applications can register to be notified when transactions succeed or fail, and when blocks are added to the ledger

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

Key:

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

Single Channel Network

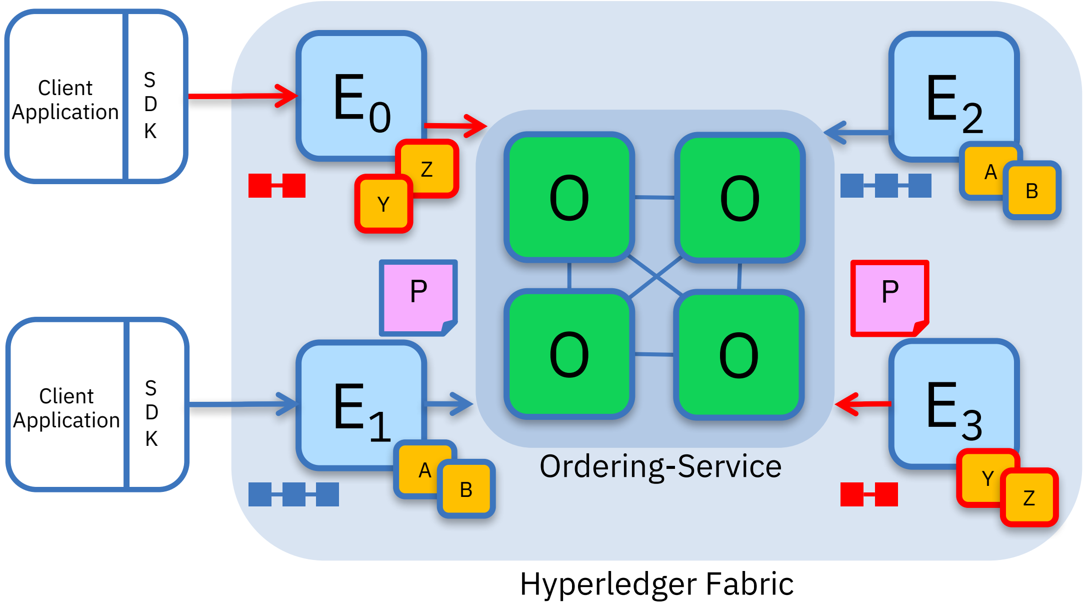


- Similar to v0.6 PBFT model
- All peers connect to the same system channel (blue).
- All peers have the same chaincode and maintain the same ledger
- Endorsement by peers E_0, E_1, E_2 and E_3

Key:

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

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

Private Blockchain Design Considerations

Business considerations

- As a B2B system, blockchain adds a number of aspects that are not typical in other projects:
  - Who pays for the development and operation of the network?
  - Where are the blockchain peers hosted?
  - When and how do new participants join the network?
  - What are the rules of confidentiality in the network?
  - Who is liable for bugs in (for example) shared smart contracts?
  - For private networks, what are the trusted forms of identity?

- Remember that each business network participant may have different requirements (e.g. trust)
  - Evaluate the incentives of potential participants to work out a viable business model
    - Mutual benefit → shared cost (e.g. sharing reference information)
    - Asymmetric benefit → money as leveler (e.g. pay for access to KYC)

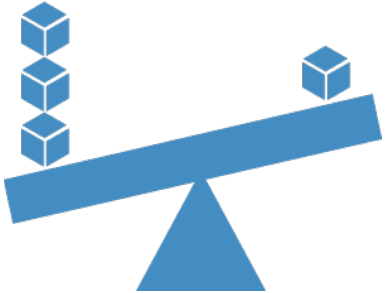
Trade-offs between non-functional requirements

- Performance
  - The amount of data being shared
  - Number and location of peers
  - Latency and throughput
  - Batching characteristics

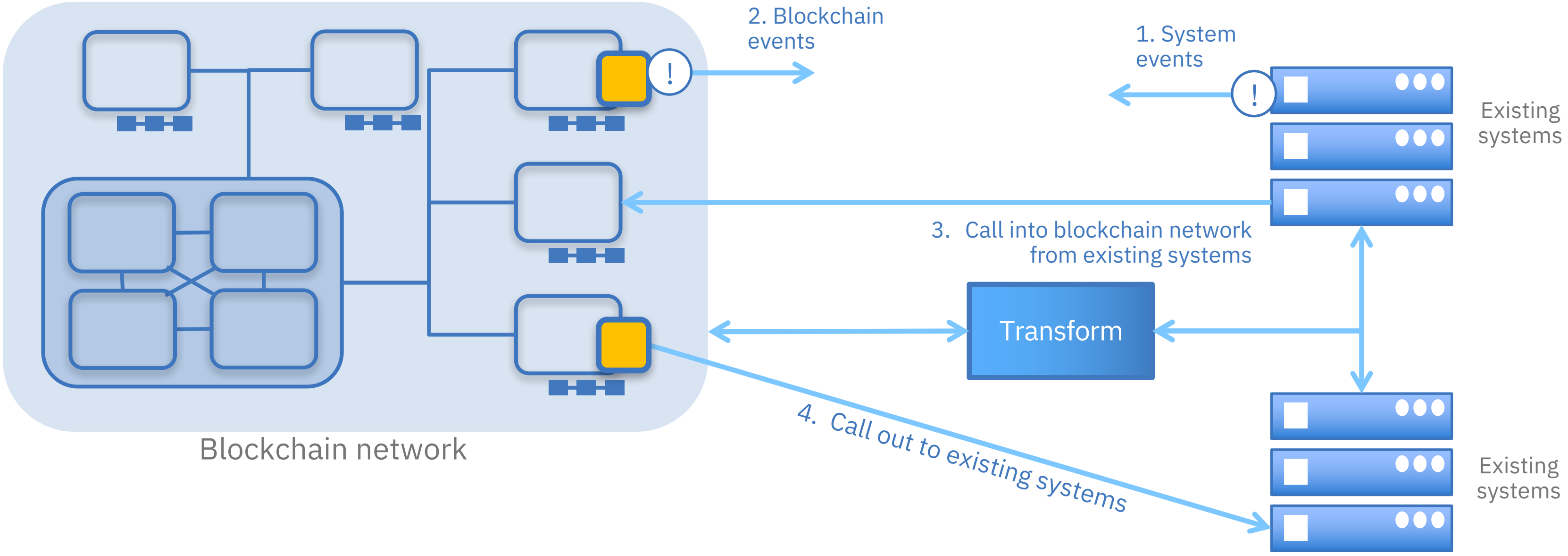
- Security
  - Type of data being shared, and with whom
  - How is identity achieved
  - Confidentiality of transaction queries
  - Who verifies (endorses) transactions

- Resiliency
  - Resource failure
  - Malicious activity
  - Non-determinism

Consider the trade-offs between performance, security and resiliency!



Integrating with existing systems – possibilities



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 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()

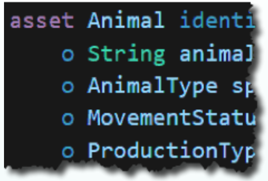
getExchangeRate()

getDateTime()

getTemperature()

incrementValue
inExternalSystem(...)

What Skills are Required to Build a Blockchain



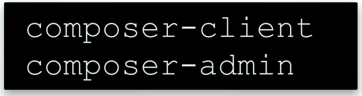
```
asset Animal identi
     o String animal
     o AnimalType sp
     o MovementStatu
     o ProductionTyp
```

Data modelling

JS

JavaScript business logic

Web playground



```
composer-client
composer-admin
```

npm

Client libraries

Editor support

```
$ composer
```

CLI utilities

YO

Code generation

Powered by

LoopBack

Node.js Framework

Swagger

Existing systems and data

Thank you

IBM **Blockchain**

www.ibm.com/blockchain

developer.ibm.com/blockchain

www.hyperledger.org