HYPER LEDGER FABRIC

A Permisioned Block Chain

Dr.N.RukmaRekha
Universityof Hyderabad
rukmarekha@uohyd.ac.in

Blockchain Defined

- A **blockchain** is a system for maintaining **distributed ledgers** in a way that allows a group of entities who **don't** fully trust each other to **agree** on the updates to the ledger using a **peer-to-peer** protocol and transact with each other without a **central authority** or an offline reconciliation process.

- Blockchains have **no single point** of failure, so entities can appear, disappear or malfunction **without** affecting the group as a whole.

- Blockchain is a design pattern made famous by its use in Bitcoin. But its uses go far beyond.

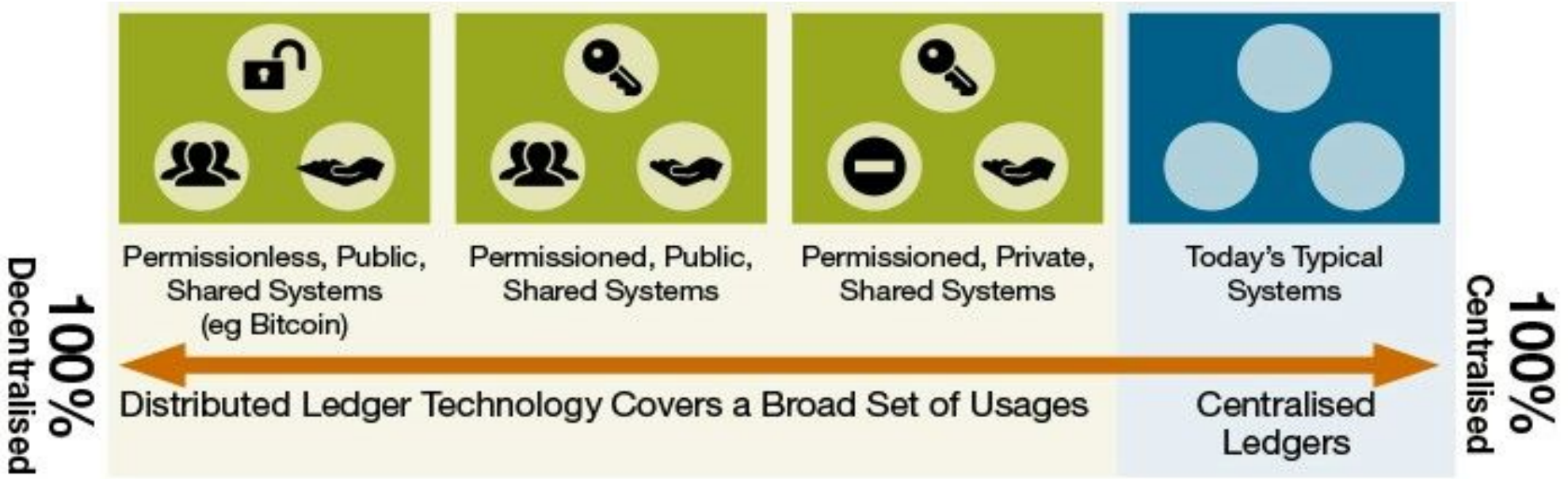
- Enterprises are adopting Blockchain to a very broad range of business applications

Key Concepts of Blockchain for Business

- **Shared Ledger**
  - Append only distibuted system of record shared across business network.
- **SmartContracts**
  - Bussiness terms embedded in transaction database & executed with transactions
- **Security**
  - Ensuring approprate visibility, transactions are secure, authenticated & verifiable
- **Consensus**
  - All parties agree to network verified transaction

Benefits of Blockchain for Business

- Reduces Time
  - Transaction time from days to near instantaneous
- Removes cost
  - Overheads and cost intermediaries
- Reduces Risk
  - Tampering, frauds & cyber crime
- Enables New Business Models
  - IoT integration into supply chain



- Censorship-resistant
- Scale to large number of nodes
- One global blockchain

- Privacy
- Scale in transaction throughput
- Many interacting blockchains

Figure source: "Distributed Ledger Technology: Beyond Blockchain", A report by UK Govt Chief Scientific Adviser

Permission-less vs Permissioned Blockchains

	Permission-less	**Permissioned**
Access	Open read/write access to database (Anyone)	Permissioned read/write access to database (By invite only)
Scale	Scale to a large number of nodes, but not in transaction throughput	Scale in terms of transaction throughput, but not to a large number of nodes
Consensus	Proof of work/ proof of stake	Closed membership consensus algorithms
Identity	Anonymous/pseudonymous	Identities of nodes are known, but transaction identities can be private/anonymous/pseudonymous
Asset	Native assets	Any asset/data/state
Currency	Required(incentivization)	Optional
Examples	Bitcoin, Ethereum,Stellar	Hyperledger Fabric, R3 Corda
Complexity	Linear	Combinatorial
Throughput	Low to Medium	High

HyperLedger Fabric

- Open source collaborative effort to advance cross-industry blockchain technologies
- Hosted by The Linux Foundation, fastest-growing project in LF history
- Global collaboration spanning finance, banking, IoT, supply chains, healthcare, manufacturing, technology and more.

Infrastructure

Technical, Legal, Marketing, Organizational

Ecosystems that accelerate open development and commercial adoption

CloudFoundry | Node.js | **Hyperledger** | Open Container Initiative

Frameworks

Meaningfully differentiated approaches to business blockchain frameworks developed by a growing community of communities

Hyperledger **Indy** | Hyperledger **Fabric** | Hyperledger **Iroha** | Hyperledger **Sawtooth** | Hyperledger **Burrow**

Modules

Typically built for one framework, and through common license and community of communities approach, ported to other frameworks

Hyperledger **Composer** | Hyperledger **Explorer** | Hyperledger **Cello**

https://medium.com/swlh/hyperledger-chapter-2-hyperledger-frameworks-modules-cabf50e12105

Hyperledger members

Premier

General

Associate

Source: https://www.hyperledger.org/about/members
Updated 21 August 2017

HyperLedger Fabric- Contd.

- Founded early 2016 as part of Linux Foundation's Hyperledger project
- Includes a ledger, uses smart contracts, and is a system by which participants manage their transactions.
- It is **permissioned**
  - New members of network enroll through a **Membership Service Provider**
- Offers several pluggable options
- Store data in multiple formats, switchable consensus mechanisms
- Offers the ability to create **channels**
  - Allowing a group of participants to create a separate ledger of transactions

What does Hyperledger Fabric contain?

- Shared Ledger and Smart Contracts

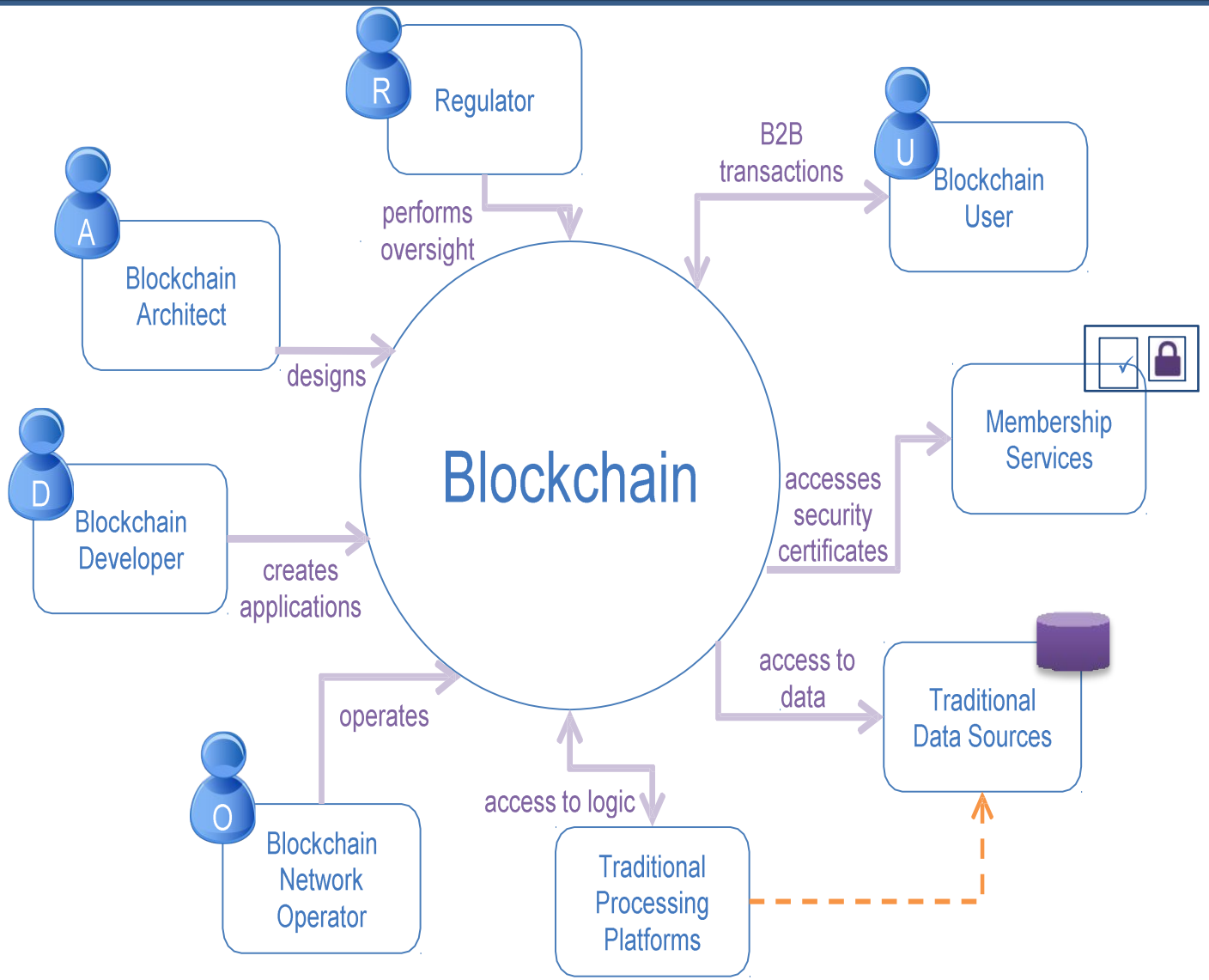
- *Ledger* comprising two components: **world state** & **transaction log**
  - World state describes state of the ledger at a given point in time (database)
  - Transaction log records all transactions which have resulted in the current value of the world state (update history)

- *Smart contracts* are written in **chaincode** and invoked by an external application
  - Mostly interacts with the database component (querying world state)

What does Hyperledger Fabric offer? Key features

- Assets — enable the exchange of almost anything with monetary value over the network, from whole foods to antique cars to currency futures.
- Chaincode — Chaincode execution is partitioned from transaction ordering, limiting the required levels of trust and verification across node types, and optimizing network scalability and performance.
- Ledger Features — encodes the entire transaction history for each channel, and includes SQL-like query capability for efficient auditing and dispute resolution.
- Privacy — **Channels** enable private and confidential multi-lateral transactions that are usually required by competing businesses and regulated industries that exchange assets on a common network.
- Security & Membership Services — provides a trusted blockchain network, where participants know that all transactions can be detected and traced by authorized regulators and auditors.
- Consensus — A unique approach that enables the flexibility and scalability needed for the enterprise.

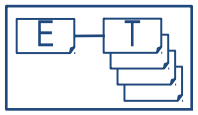
Actors in a Blockchain Solution



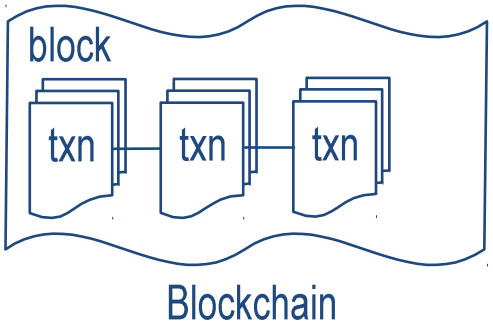
Actors in a Blockchain Solution

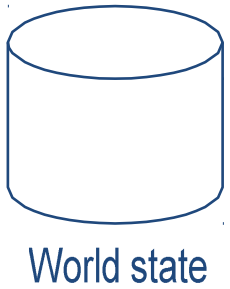
Blockchain Architect		Responsible for the architecture and design of the blockchain solution
Blockchain User		The business user, operating in a business network. This role interacts with the Blockchain using an application. They are not aware of the Blockchain.
Blockchain Regulator		The overall authority in a business network. Specifically, regulators may require broad access to the ledger's contents.
Blockchain Developer		The developer of applications and smart contracts that interact with the Blockchain and are used by Blockchain users.
Blockchain Operator		Manages and monitors the Blockchain network. Each business in the network has a Blockchain Network operator.
Membership Services		Manages the different types of certificates required to run a permissioned Blockchain.
Traditional Processing Platform		An existing computer system which may be used by the Blockchain to augment processing. This system may also need to initiate requests into the Blockchain.
Traditional Data Sources		An existing data system which may provide data to influence the behavior of smart contracts.

Components in a Blockchain Solution

Ledger		A ledger is a channel's chain and current state data which is maintained by each peer on the channel.
Smart Contract		Software running on a ledger, to encode assets and the transaction instructions (business logic) for modifying the assets.
Peer Network		A broader term overarching the entire transactional flow, which serves to generate an agreement on the order and to confirm the correctness of the set of transactions constituting a block.
Membership		Membership Services authenticates, authorizes, and manages identities on a permissioned blockchain network.
Events		Creates notifications of significant operations on the blockchain (e.g. a new block), as well as notifications related to smart contracts.
Systems Management		Provides the ability to create, change and monitor blockchain components
Wallet		Securely manages a user's security credentials
Systems Integration		Responsible for integrating Blockchain bi-directionally with external systems. Not part of blockchain, but used with it.

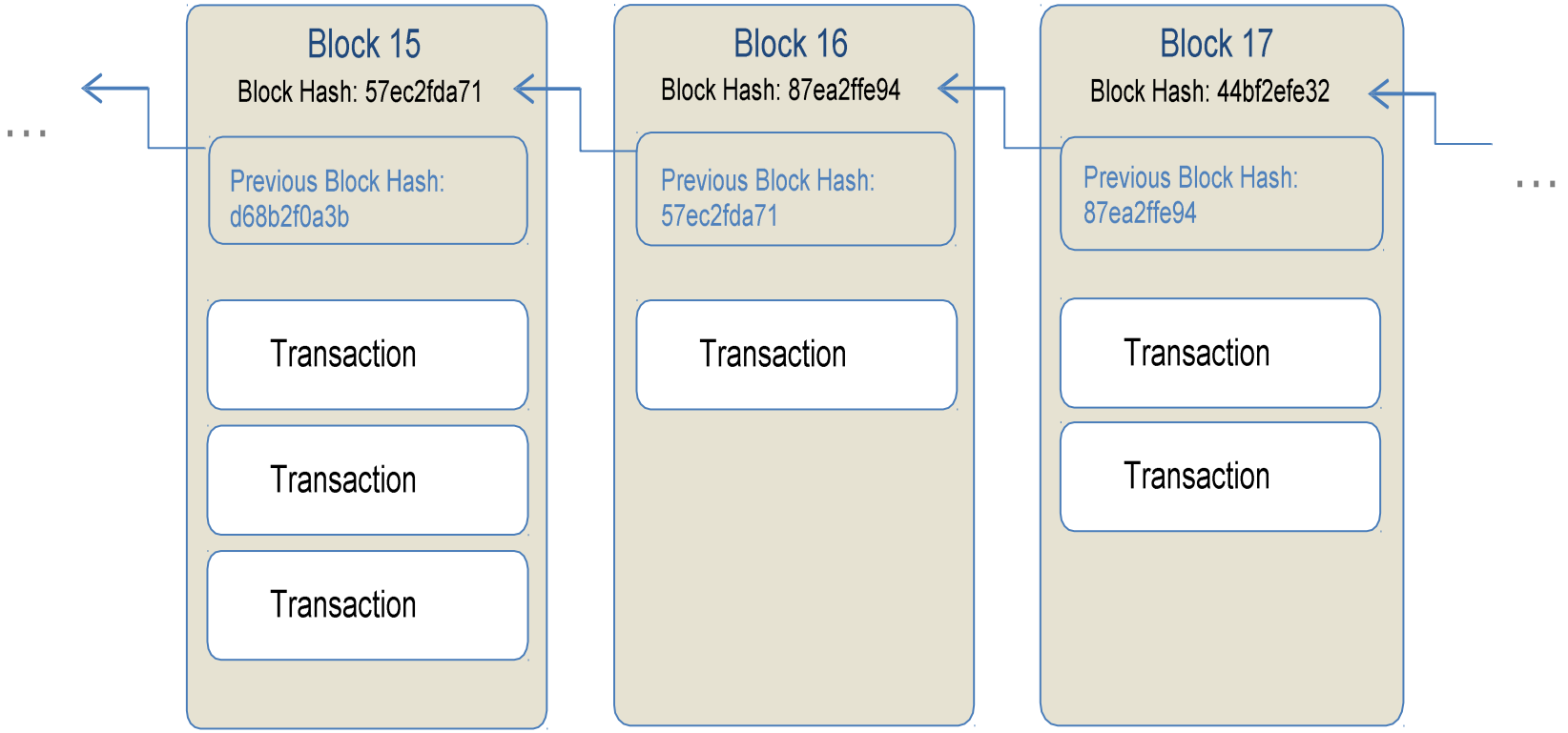
A ledger often consists of two data structures





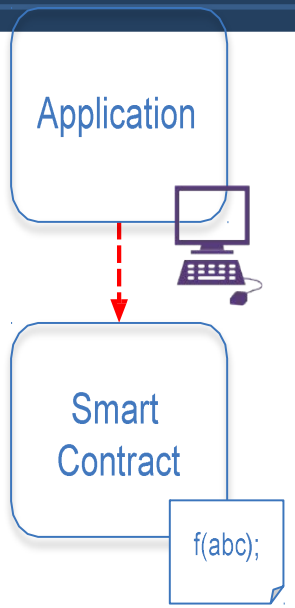
- Blockchain
  - A linked list of blocks (a hashchain)
  - Each block describes a set of transactions (e.g. the inputs to a smart contract invocation, output, identities/certs)
  - Immutable – blocks cannot be tampered
- World State
  - Stores the most recent state of smart contracts / output of transactions
  - Stored in a traditional database (e.g. key-value store)
  - Data elements can been added, modified, deleted, all recorded as transactions on blockchain

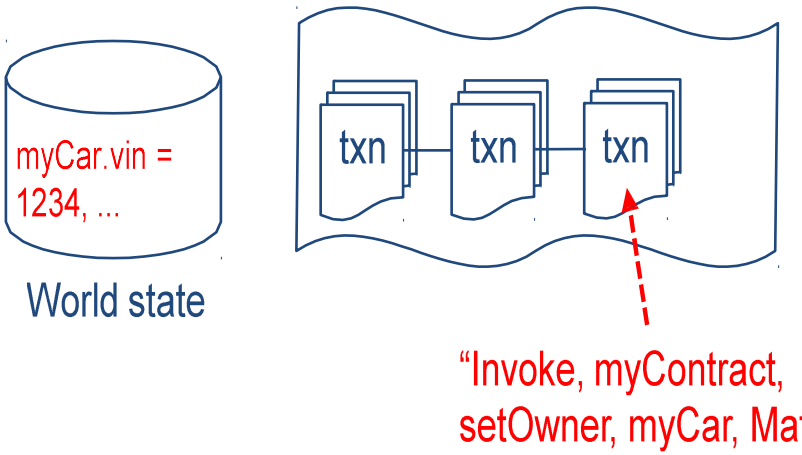
Block Detail (Simplified)



- 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

Ledger Example: A Change of Ownership Transaction





Transaction input - sent from application

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

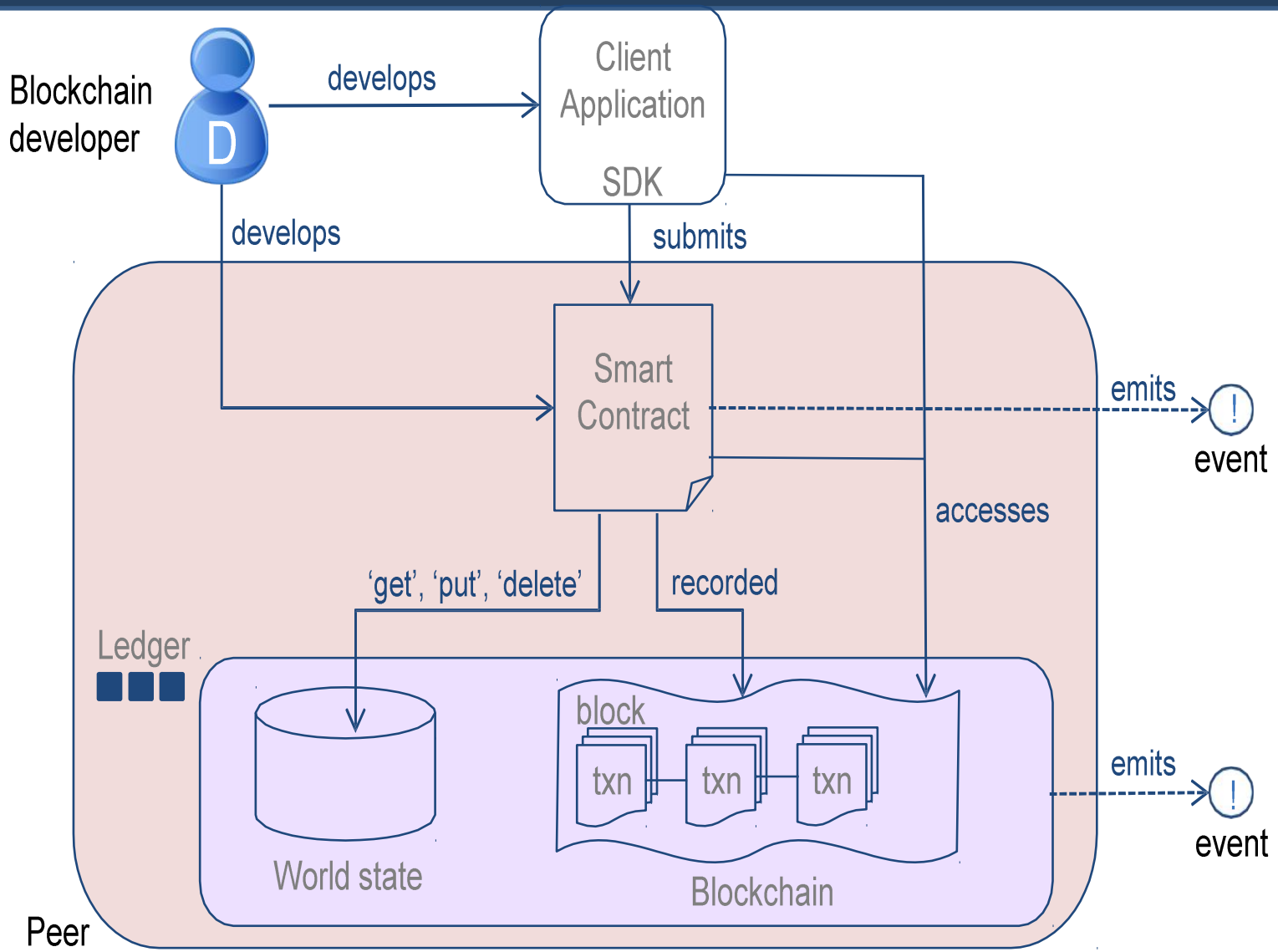
Smart contract implementation

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

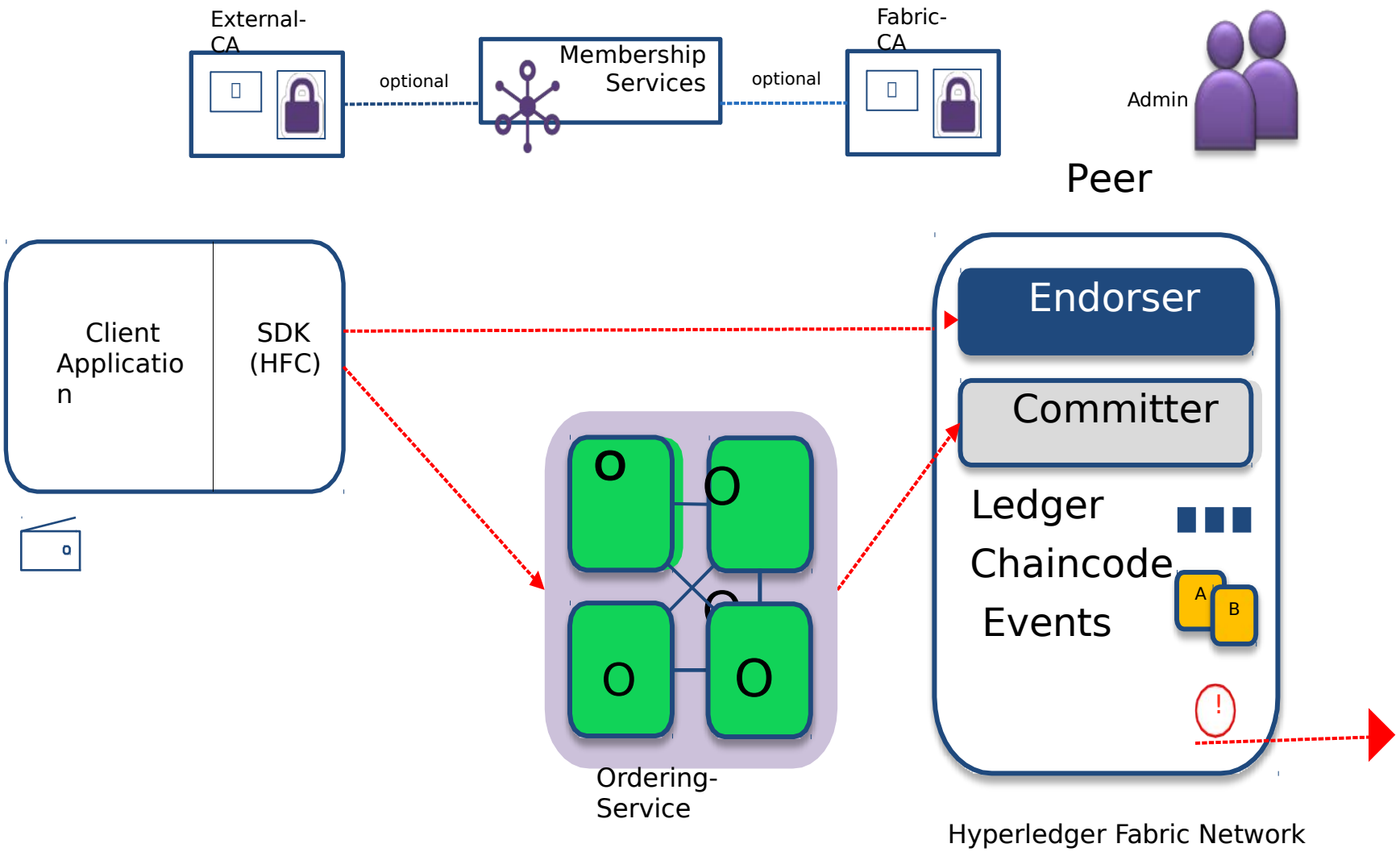
World state: new contents

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

How Applications Interact with the Ledger

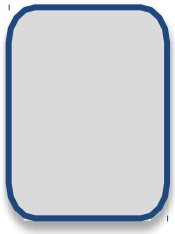
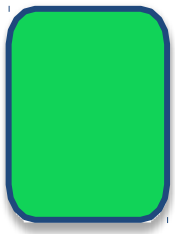


Hyperledger Fabric V1 Architecture



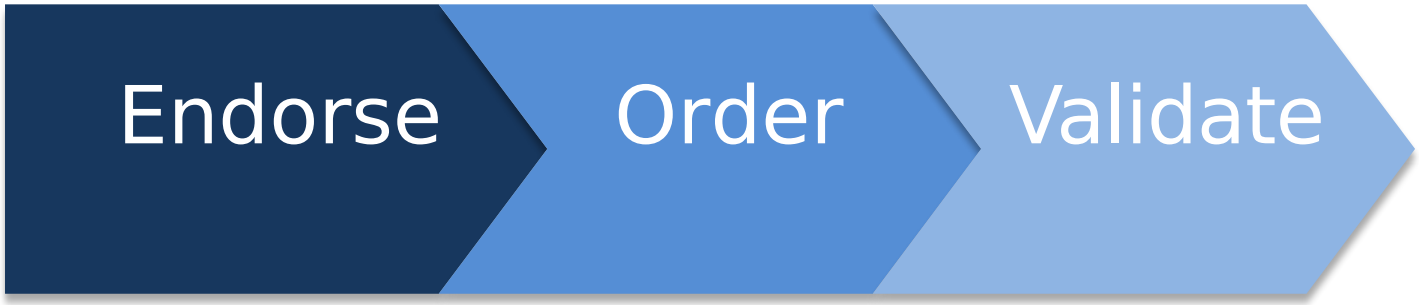
Validated ledger and PeerLedger checkpointing (pruning)--v4

Nodes and Roles

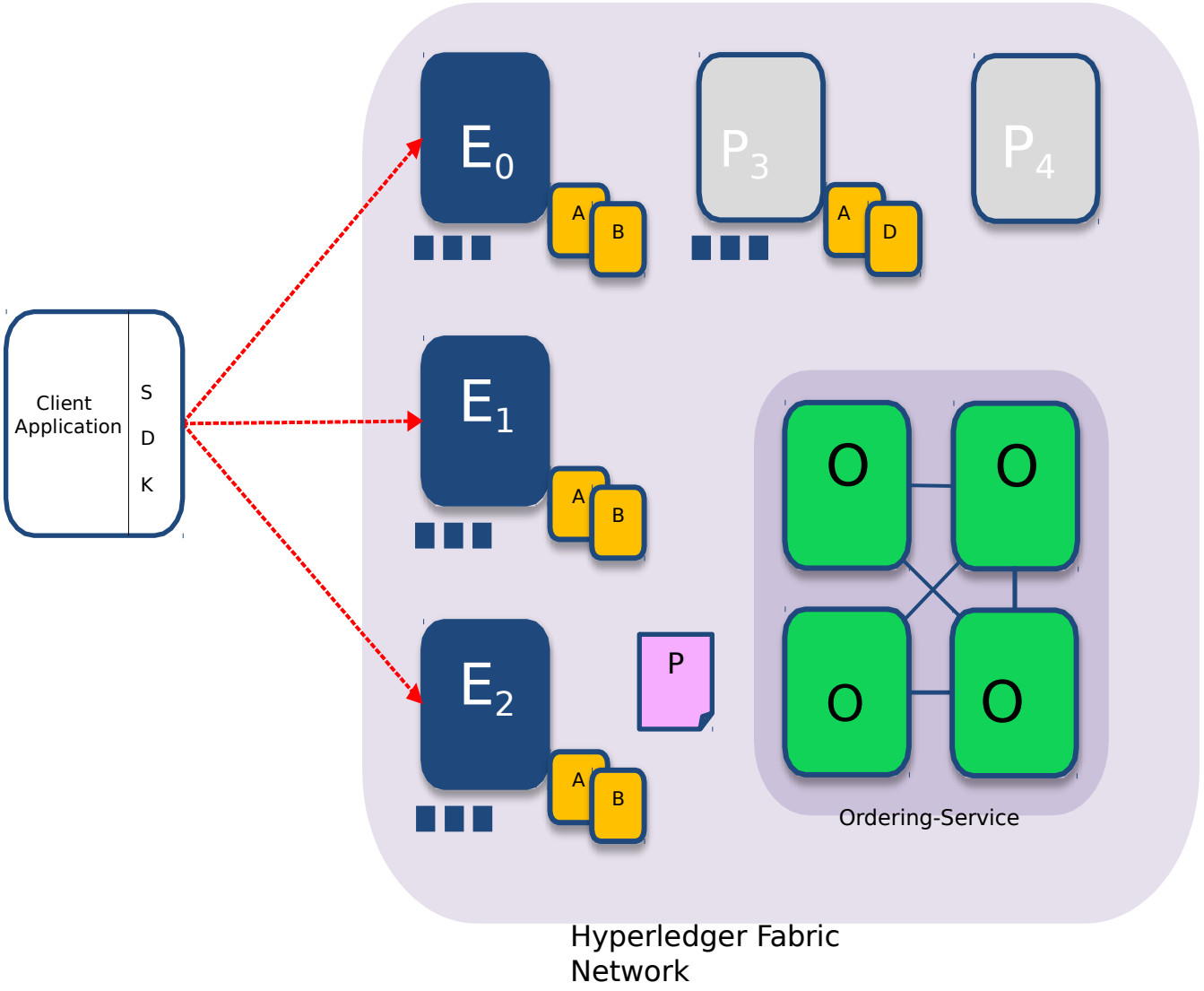
	**Committing Peer:** Maintains ledger and state. Commits transactions. May hold smart contract (chaincode).
	**Endorsing Peer:** Specialized committing peer that receives a transaction proposal for endorsement, responds granting or denying endorsement. Must hold smart contract
	**Ordering Node:** Approves 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.

Transaction Flow

Consensus is achieved using the following transaction flow:



Step 1/7: Propose Transaction



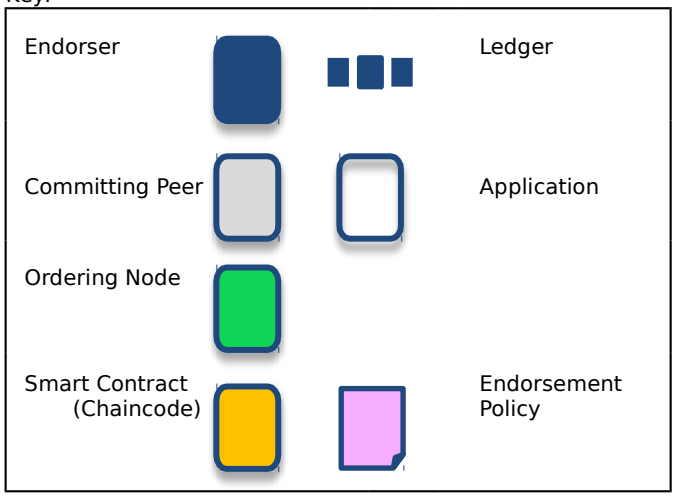
Hyperledger Fabric Network

Application proposes transaction

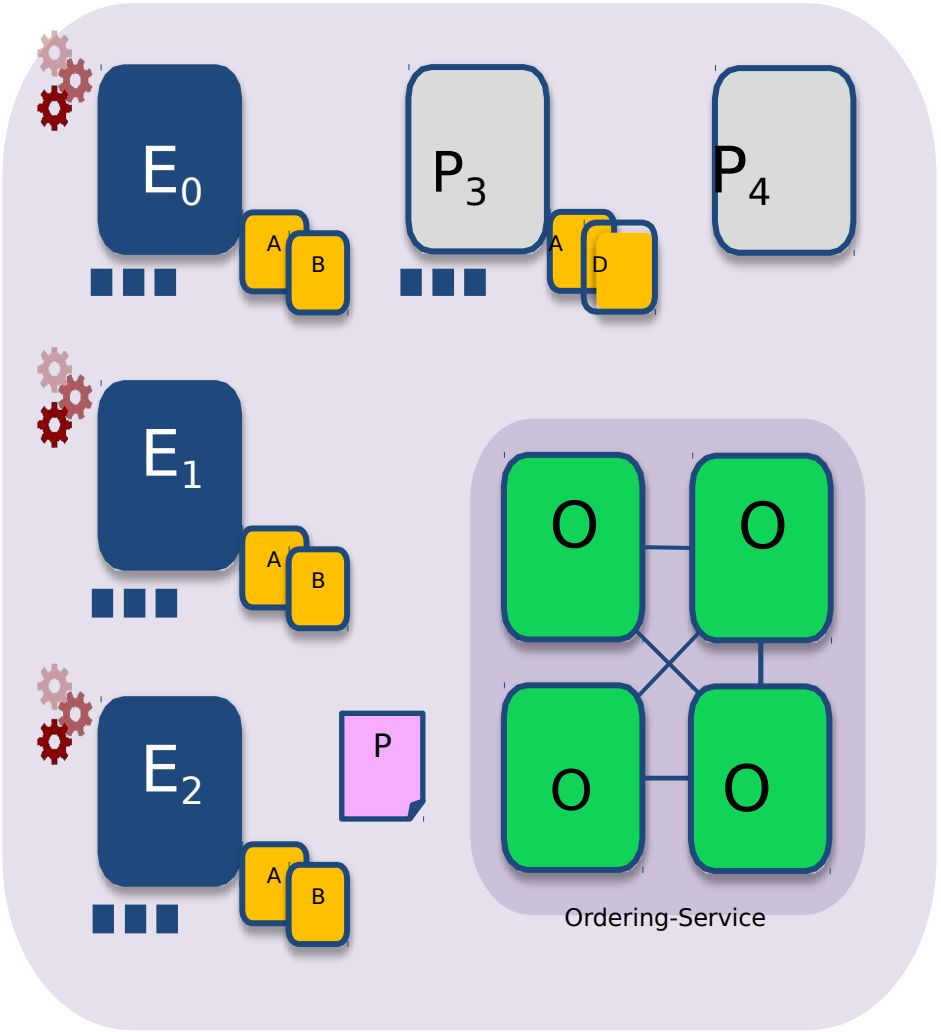
Endorsement policy:
- "E_0, E_1 and E_2 must sign"
- (P_3, P_4 are not part of the policy)

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

Key:



Step 2/7: Execute Proposed Transaction



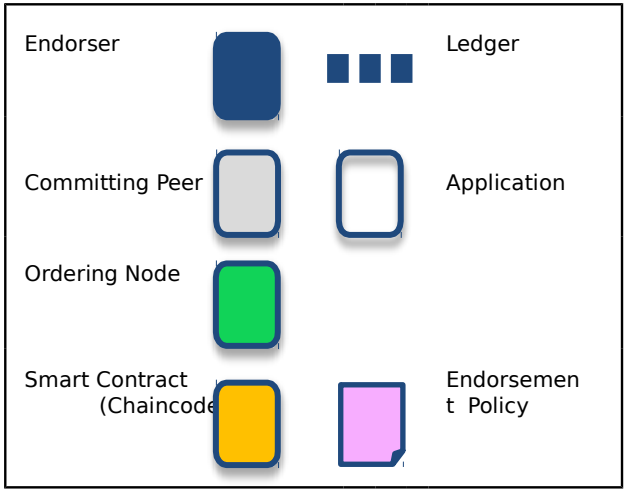
Hyperledger Fabric Network

Client Application | SDK

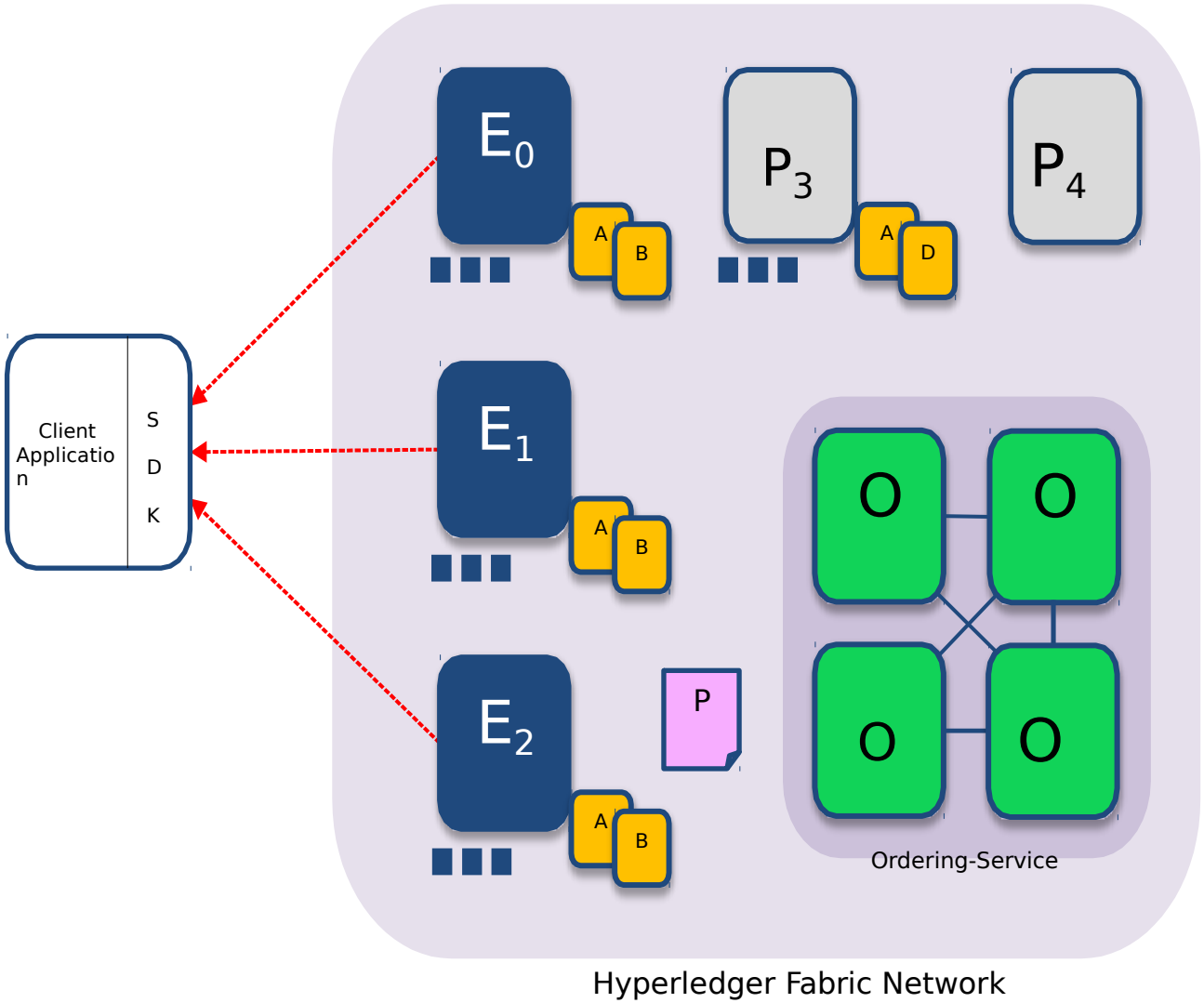
Endorsers Execute Proposals

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:



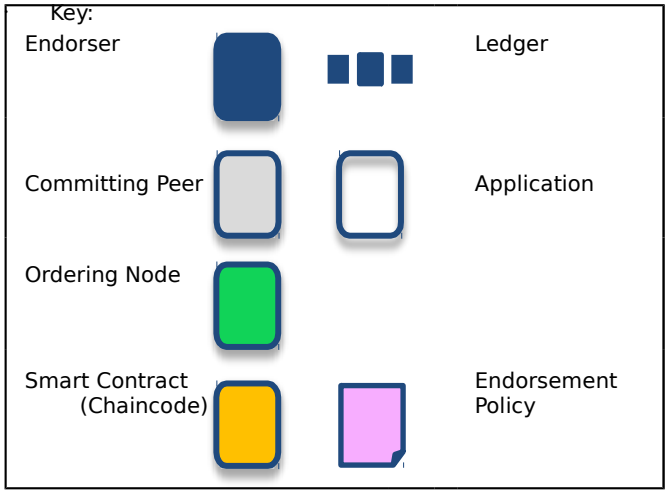
Step 3/7: Proposal Response



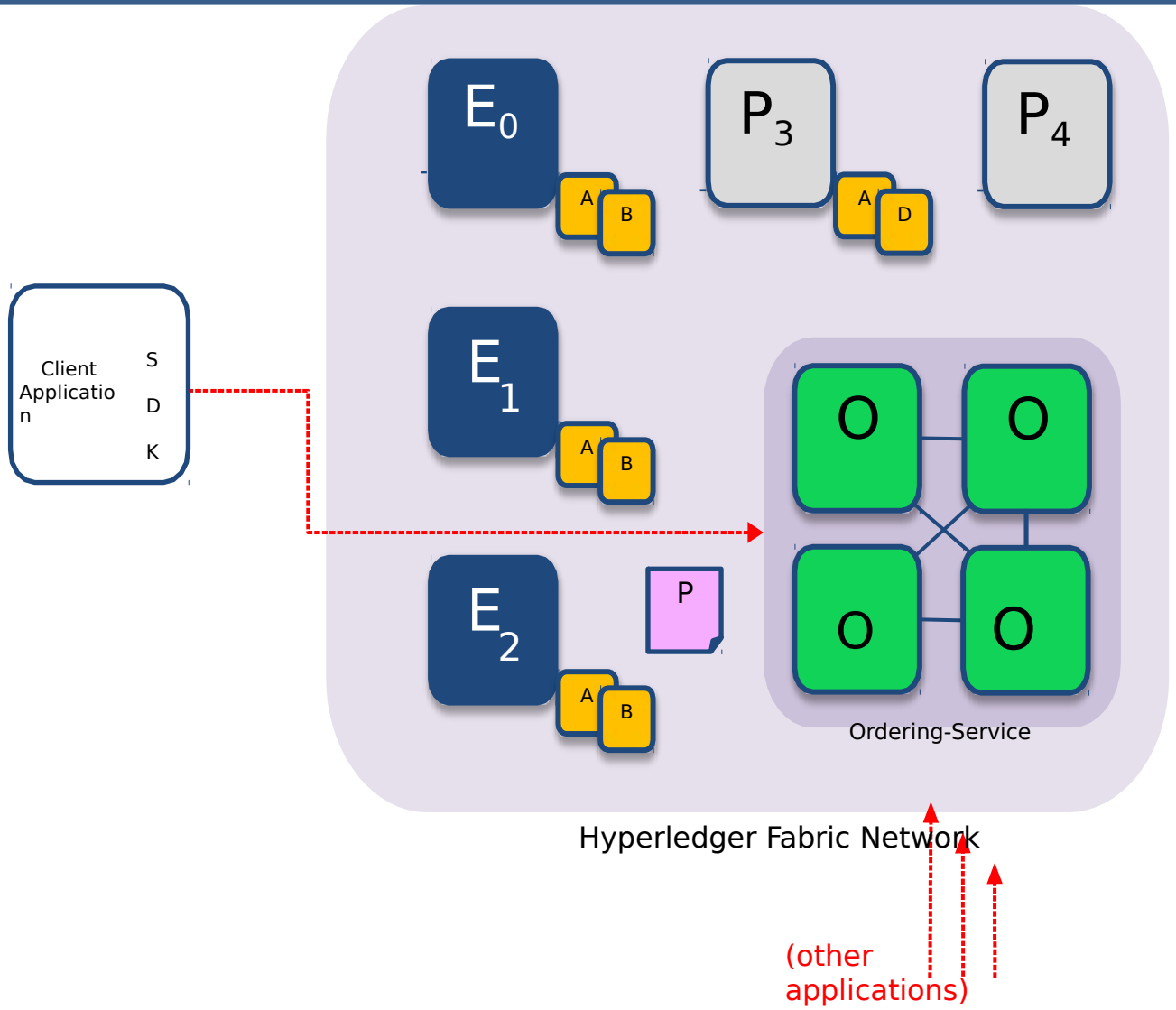
Hyperledger Fabric Network

Application receives responses

- Read-Write 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)



Step 4/7: Order Transaction

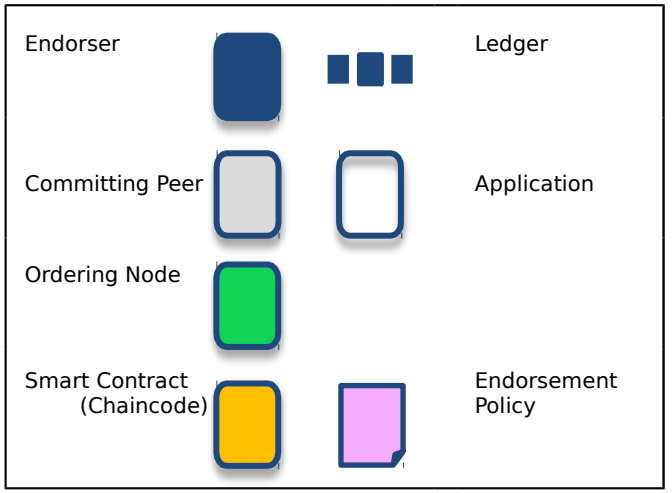


Responses submitted for ordering

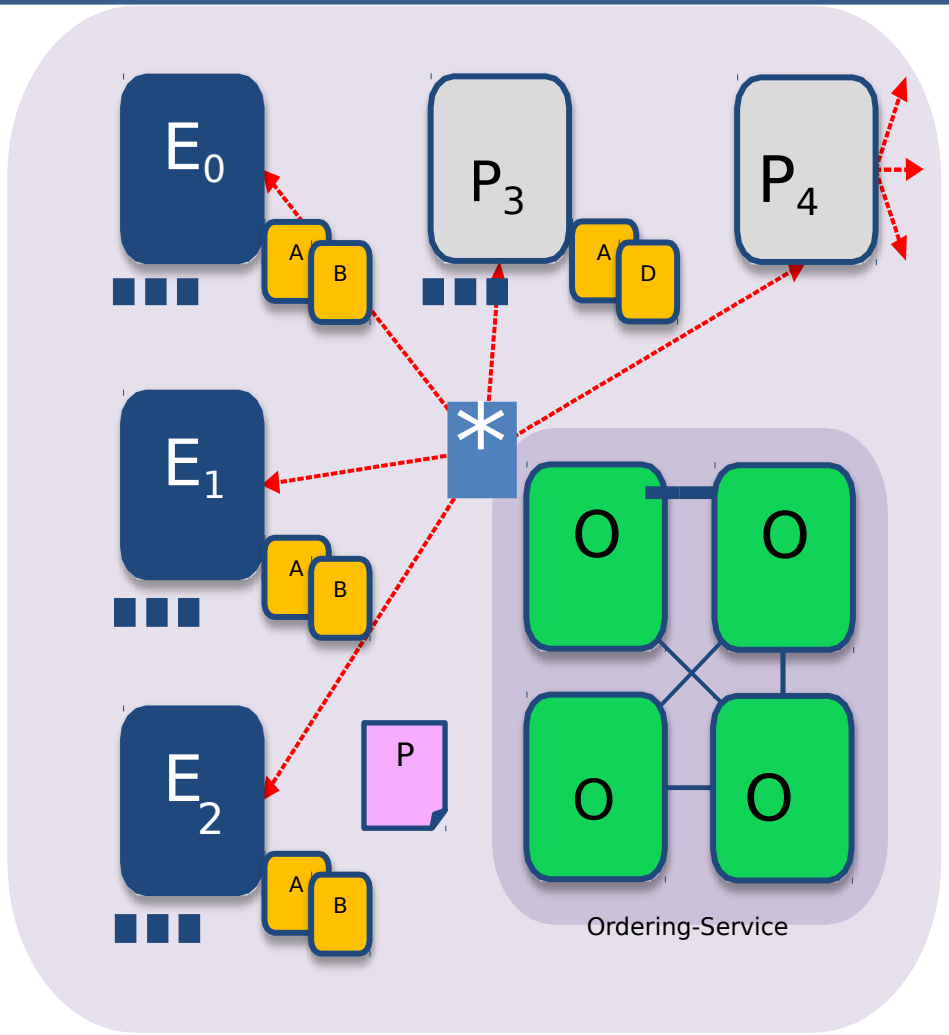
Application submits responses as a transaction to be ordered.

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

Key:



Step 5/7: Deliver Transaction



Hyperledger Fabric Network

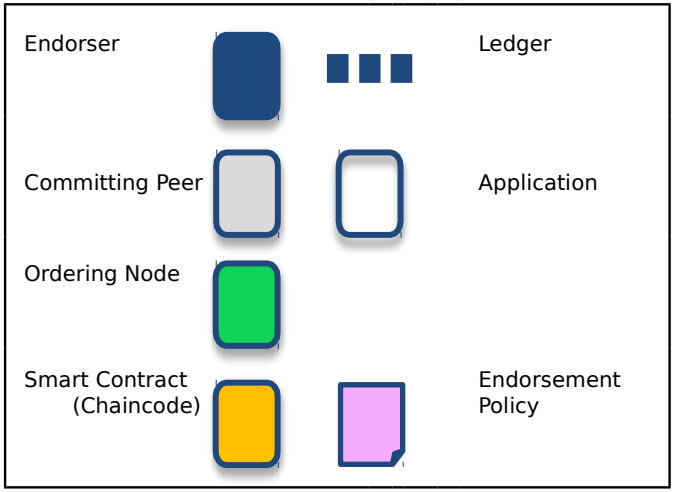
Client Application | S D K

Orderer delivers to committing peers

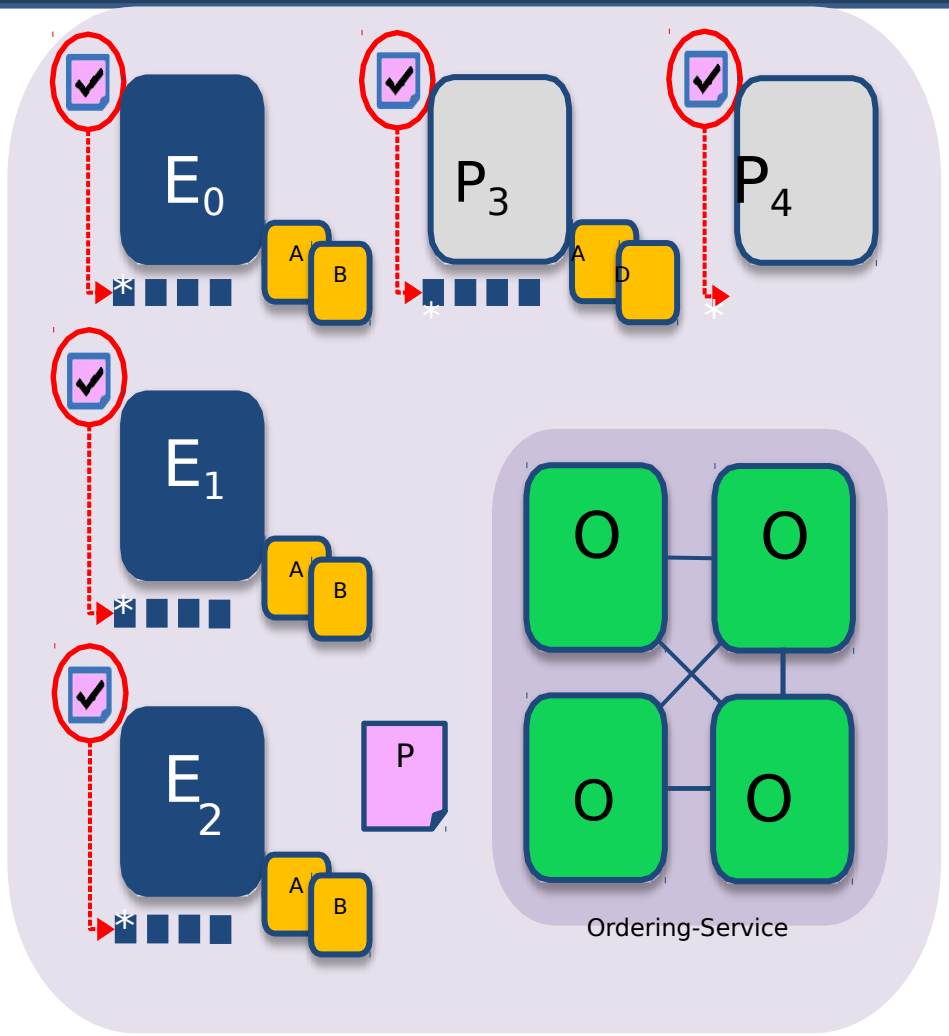
Ordering service collects transactions into proposed blocks for distribution to committing peers. Peers can deliver to other peers in a hierarchy (not shown) Different ordering algorithms available:

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

Key:



Step 6/7: Validate Transaction



Client Application

SDK

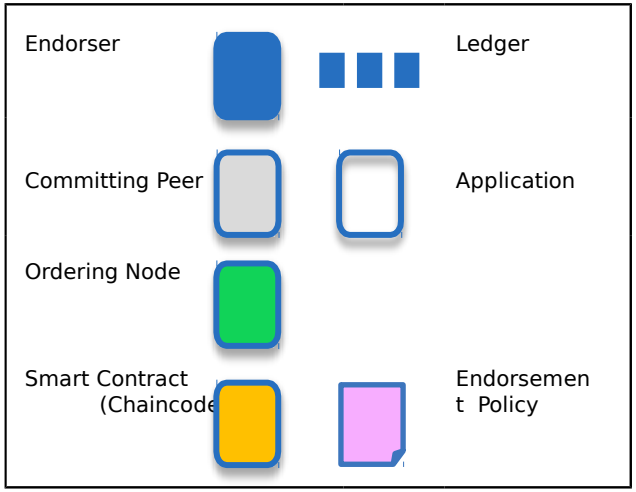
Hyperledger Fabric Network

Committing peers validate transactions

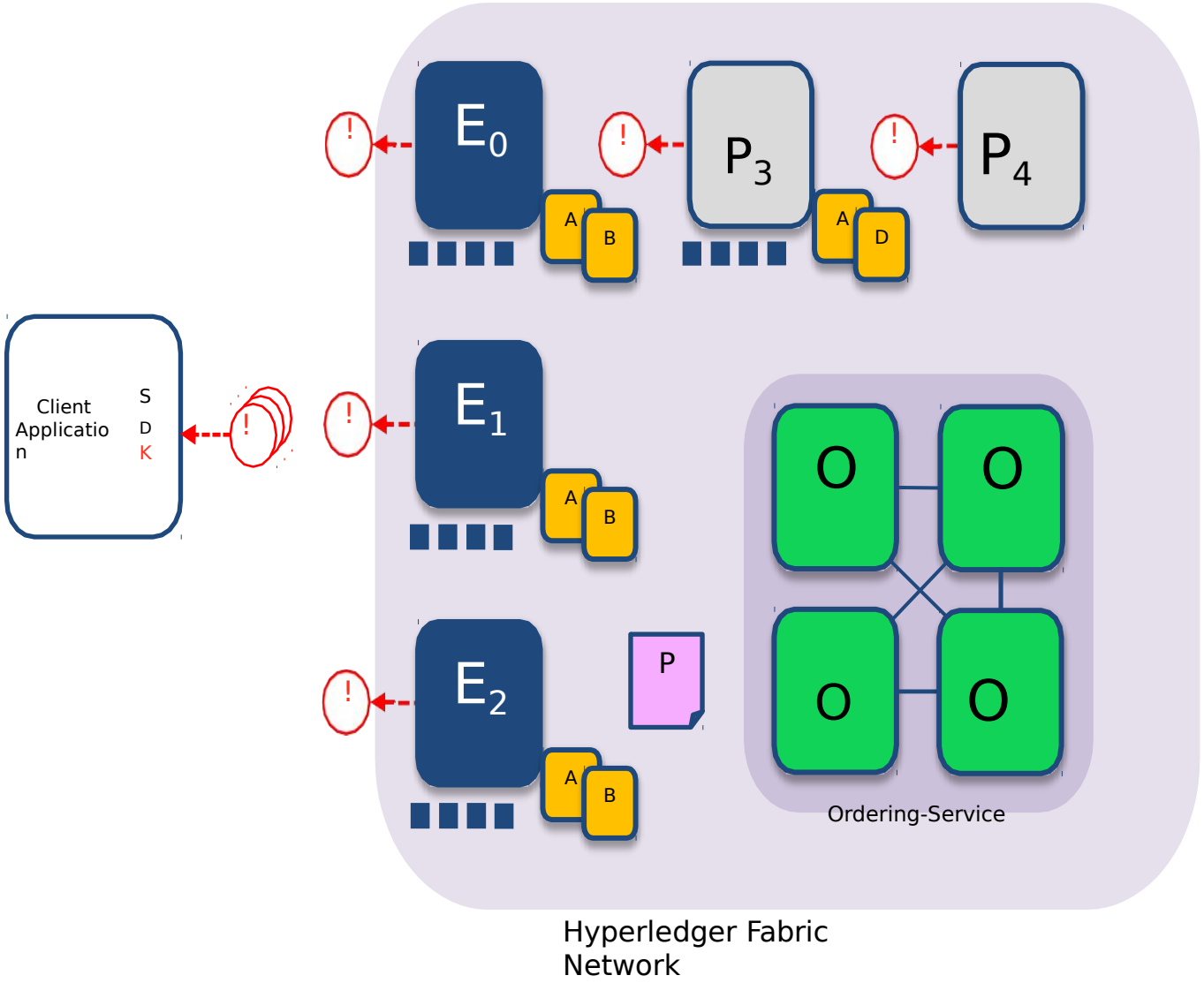
Every committing peer validates against the endorsement policy. Also check RW sets are still valid for current world state

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

Key:



Step 7/7: Notify Transaction



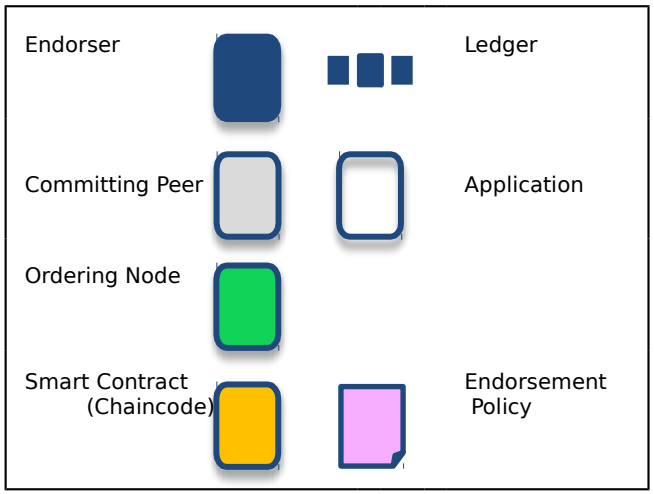
Hyperledger Fabric Network

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

Key:



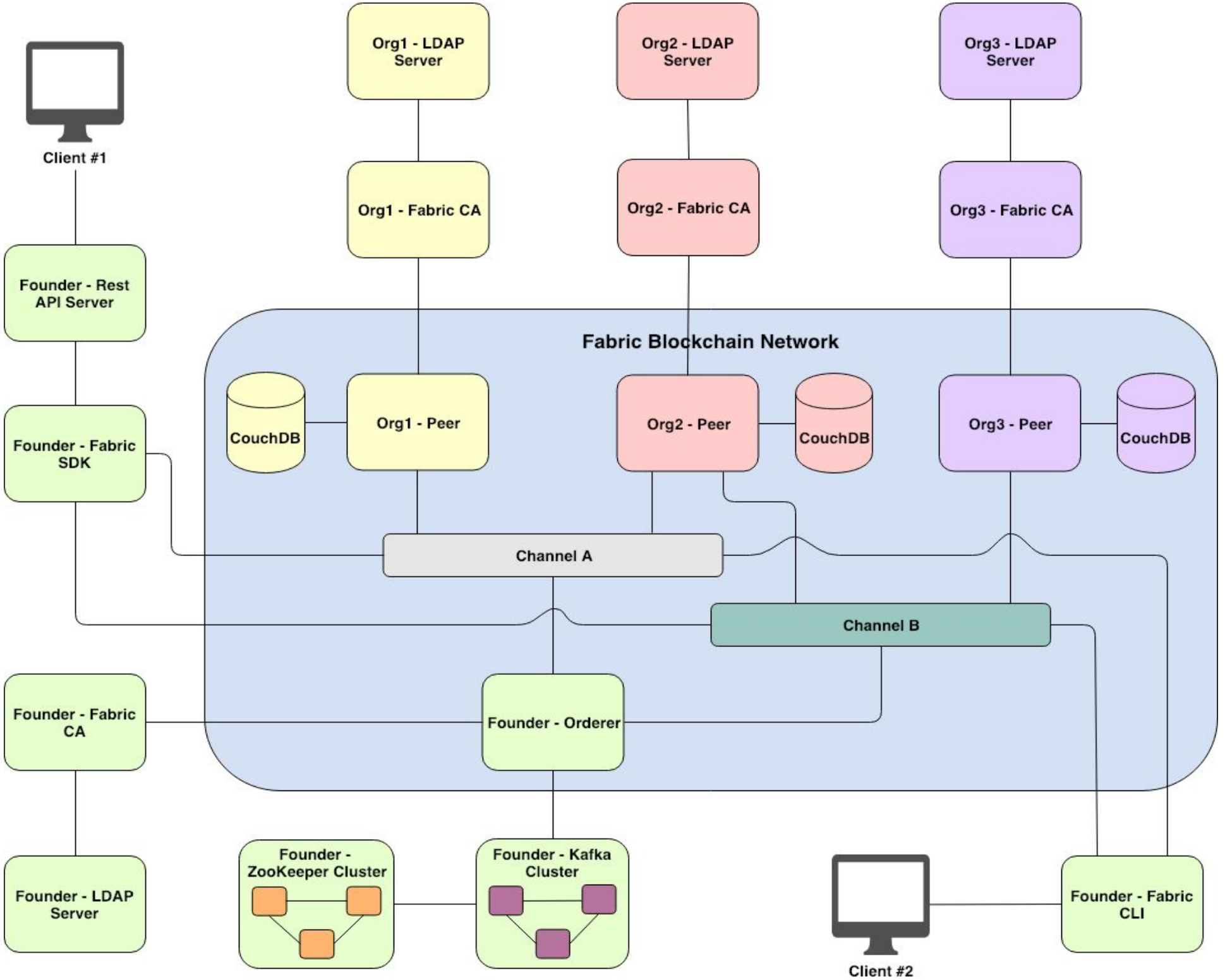
Validated ledger --v4

- Validated ledger (Vledger)
  - abstraction of a ledger that contains only valid and committed transactions
  - a hash chain derived from the ledger by filtering out invalid transactions.
  - every block of a validated ledger contains:
    - The hash of the previous vBlock.
    - vBlock number.
    - list of valid transactions in a corresponding block.
    - The hash of the corresponding block
  - All this information is concatenated and hashed by a peer, producing the hash of the vBlock in the validated ledger.

PeerLedger Checkpointing--V4

- Checkpointing is performed periodically by the peers every CHK blocks.
- To initiate a checkpoint, the peers broadcast to other peers message <CHECKPOINT,blocknohash,blockno,stateHash,peerSig>,
- A peer collects CHECKPOINT messages until it obtains enough correctly signed messages with matching blockno, blocknohash and stateHash to establish a valid checkpoint

- Upon establishing a valid checkpoint for block number blockno with blocknohash, a peer:
  - if blockno>latestValidCheckpoint.blockno, then a peer assigns latestValidCheckpoint=(blocknohash,blockno),
  - stores the set of respective peer signatures that constitute a valid checkpoint into the set latestValidCheckpointProof,
  - stores the state corresponding to stateHash to latestValidCheckpointedState,
  - (optionally) prunes its PeerLedger up to block number blockno (inclusive).


Client #1
Org1 - LDAP Server
Org2 - LDAP Server
Org3 - LDAP Server
Org1 - Fabric CA
Org2 - Fabric CA
Org3 - Fabric CA
Founder - Rest API Server
Fabric Blockchain Network
CouchDB
Org1 - Peer
Org2 - Peer
CouchDB
Org3 - Peer
CouchDB
Founder - Fabric SDK
Channel A
Channel B
Founder - Fabric CA
Founder - Orderer
Founder - LDAP Server
Founder - ZooKeeper Cluster
Founder - Kafka Cluster
Client #2
Founder - Fabric CLI

Key Benefits of the Transaction Flow

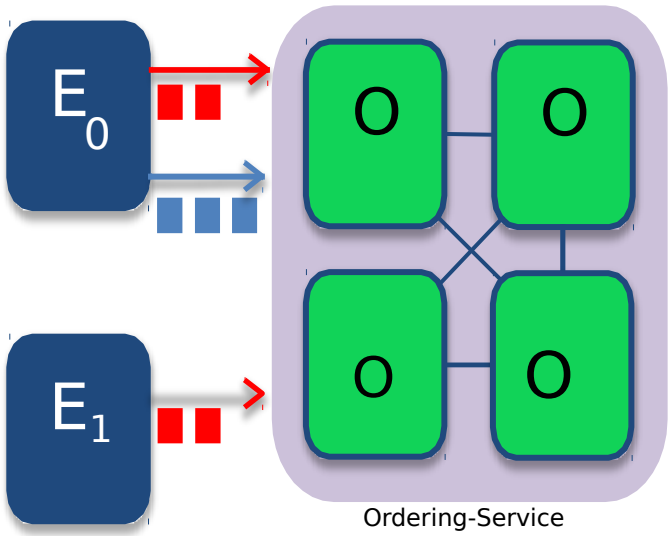
- Better reflect business processes by specifying who endorses transactions

- Eliminate non deterministic transactions

- Scale the number of participants and transaction throughput

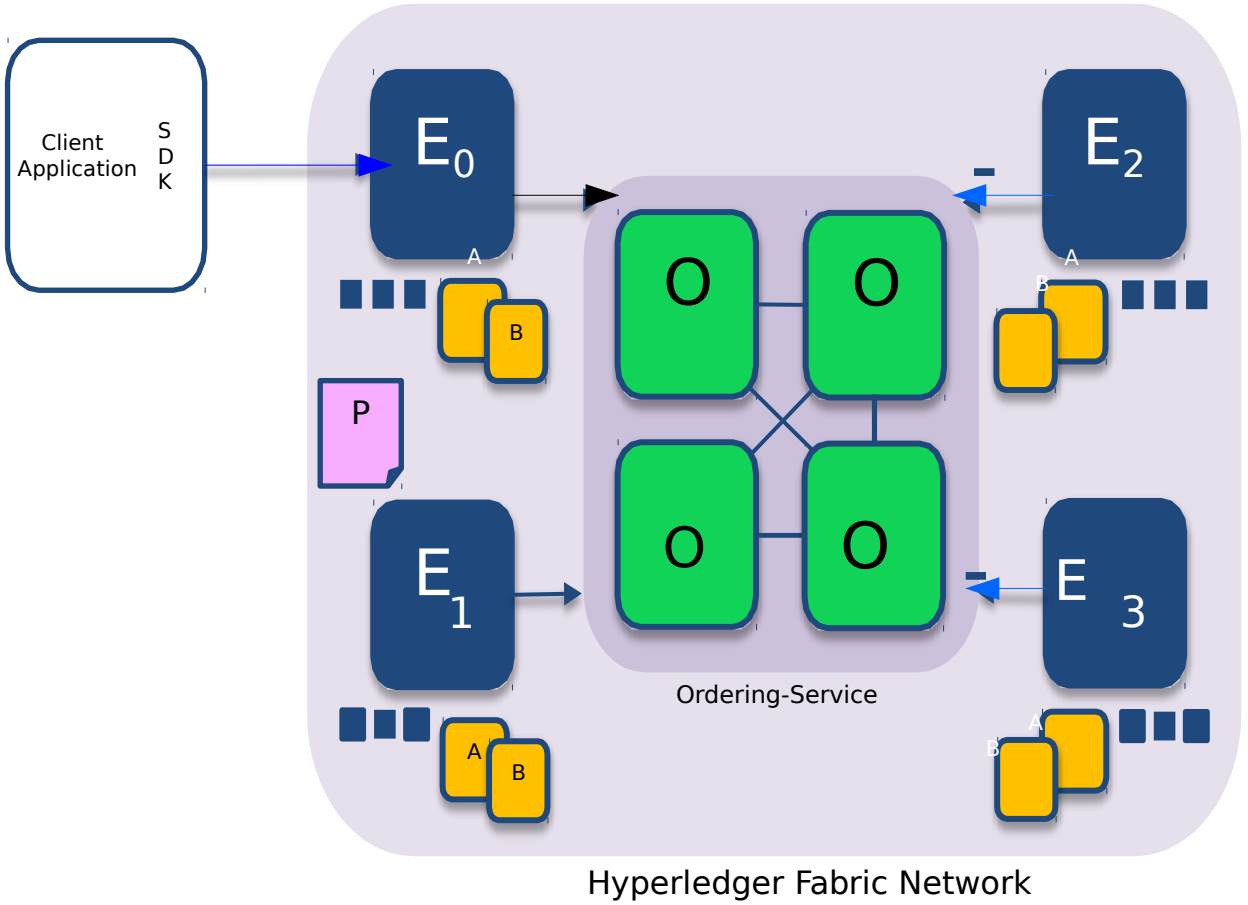
Channels

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
- Peers can participate in multiple channels
- Concurrent execution for performance and scalability

Single Channel Network

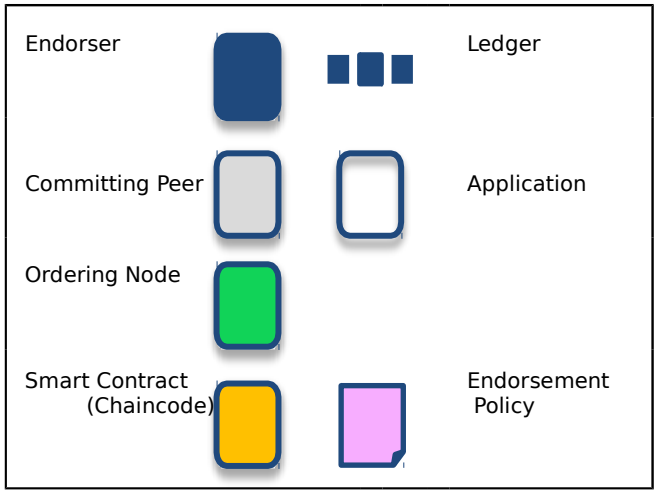


Hyperledger Fabric Network

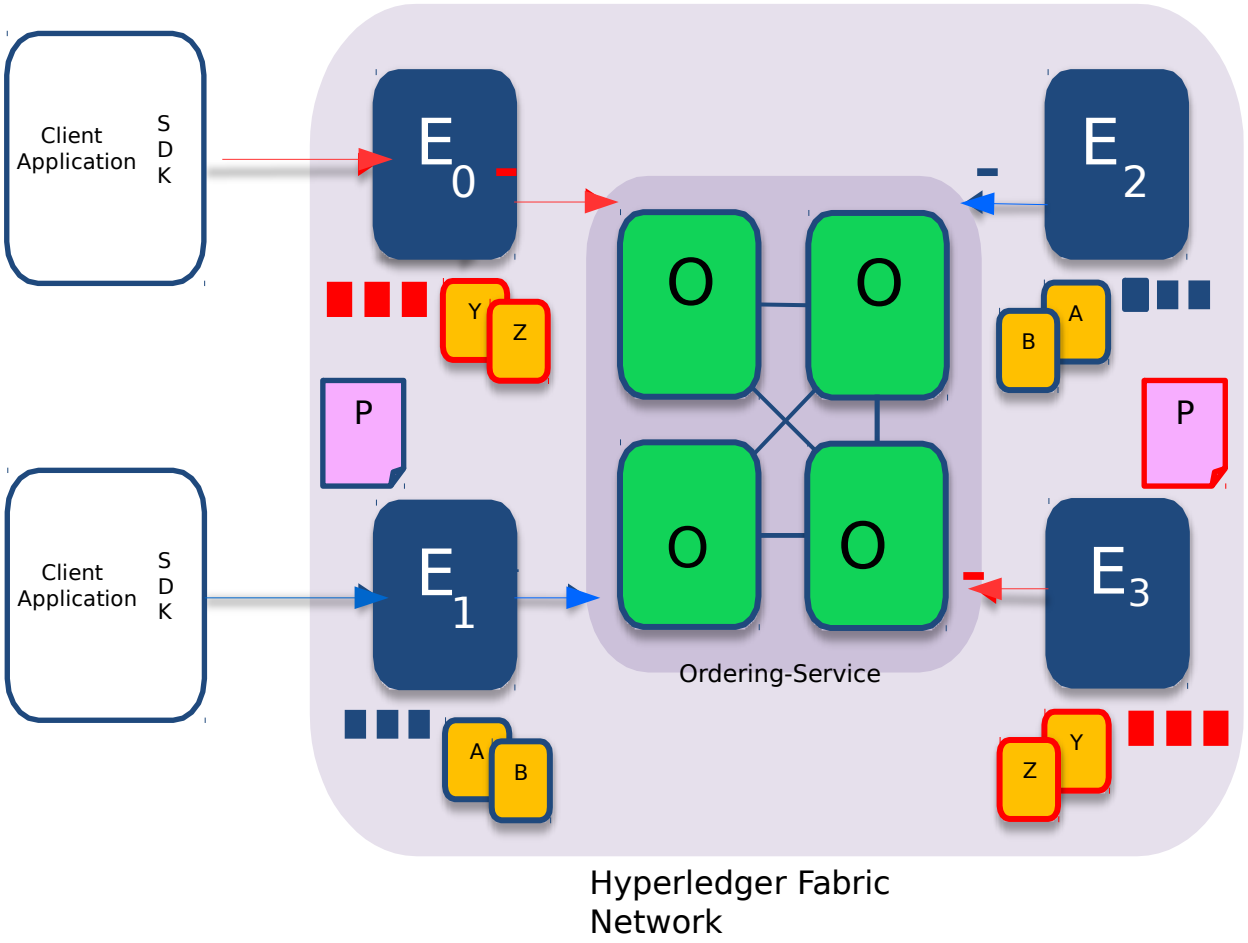
- All peers connect to the same system channel (blue).
- All peers have the same chaincode and maintain the same ledger

- Endorsement by peers E_0 E_1 E_2 and E_3

Key:

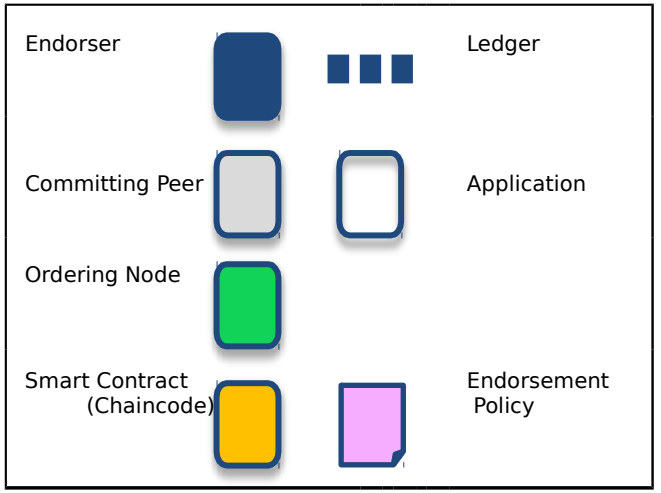


Multi-Channel Network



- 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

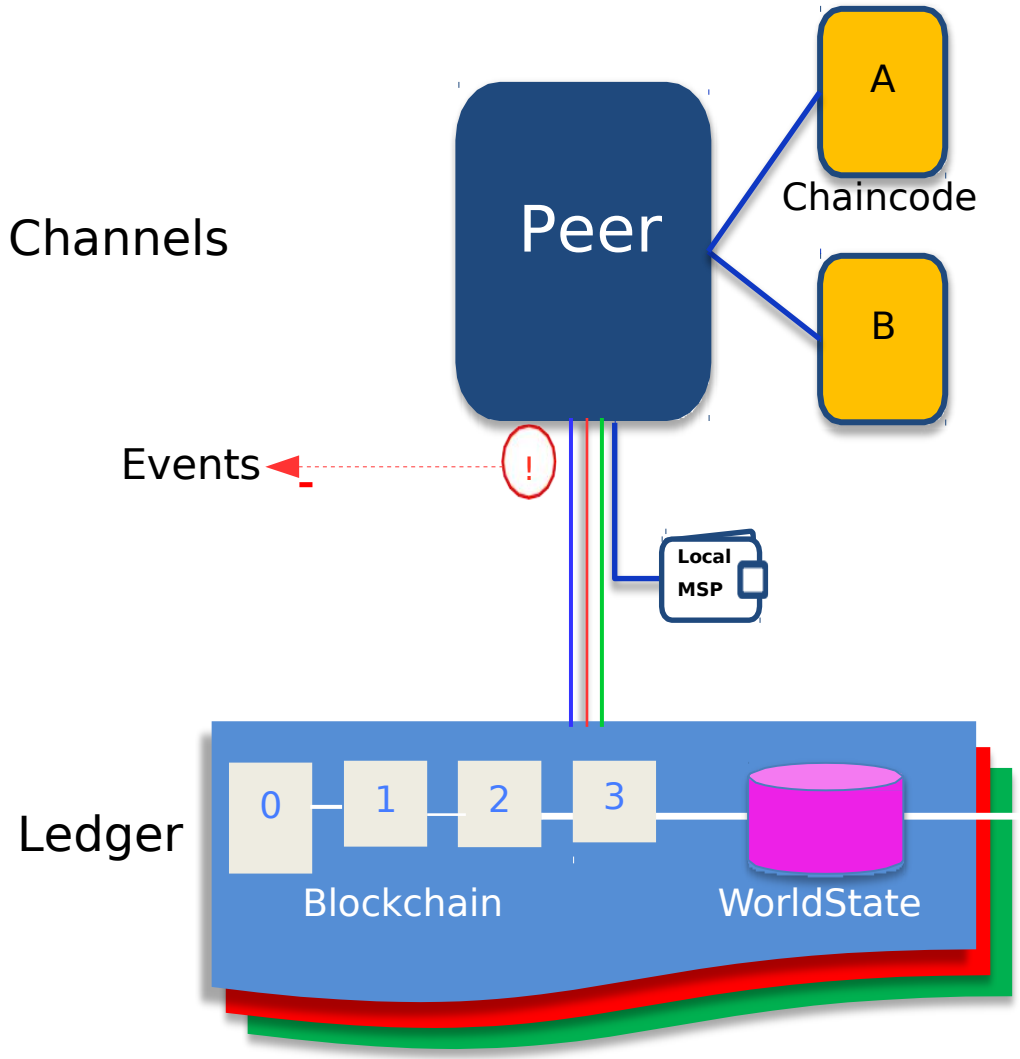
Key:



Fabric Peer

- Each peer:

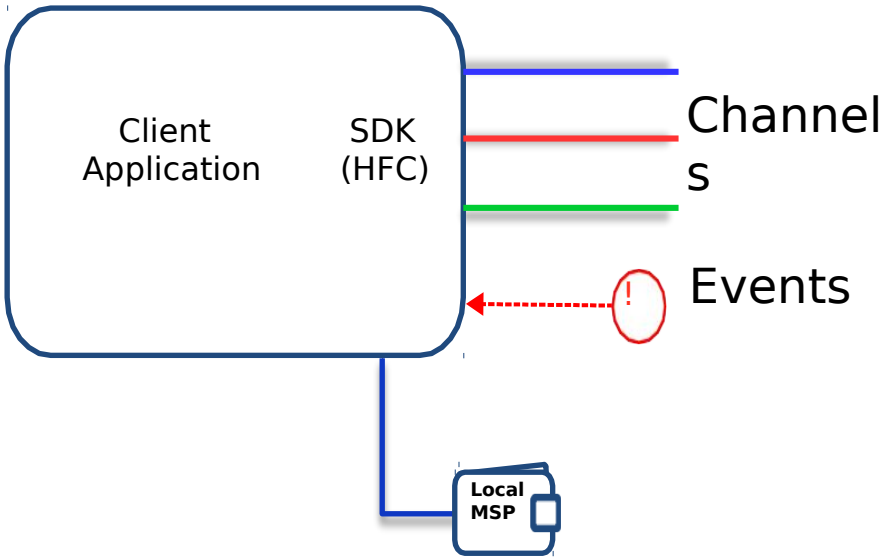
- Connects to one or more channels
- Maintains one or more ledgers for each channel
- Chaincodes are instantiated in separate docker containers
- Chaincodes are shared across channels (no state is stored in chaincode container)
- Local MSP (Membership Services Provider) provides crypto material
- Emits events to the client application



Client Application

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

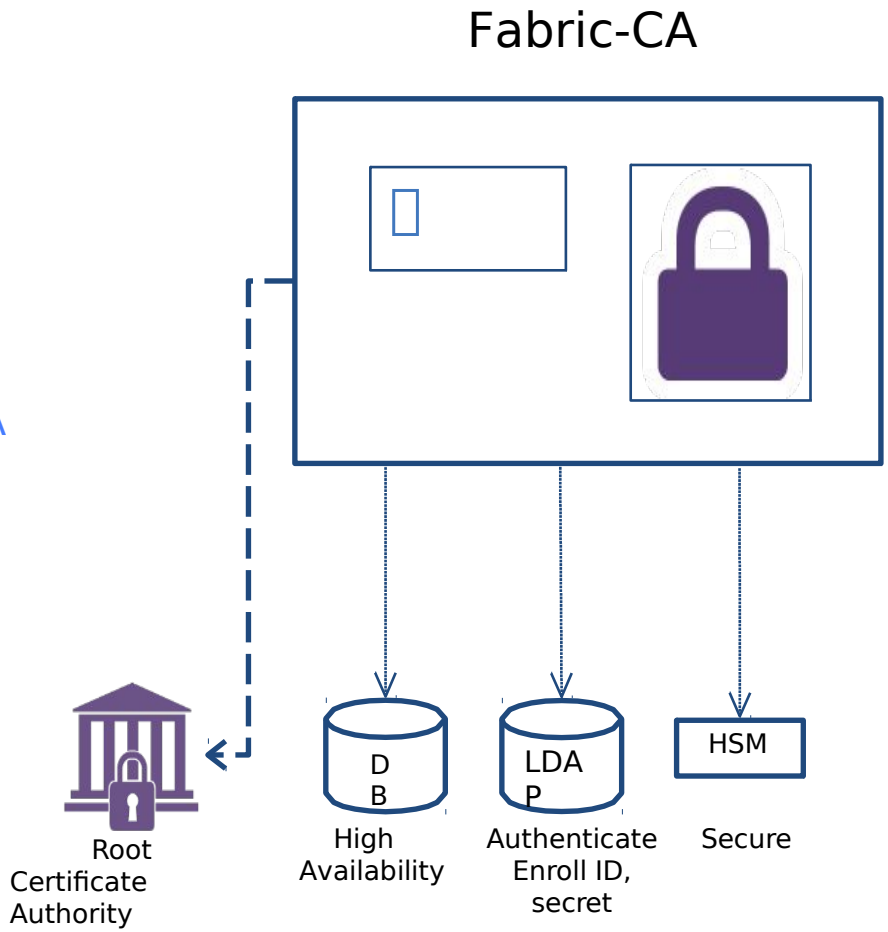
  - Client can be written in different languages (Node.js, Go, Java, Python?)



39

Fabric Certificate Authority

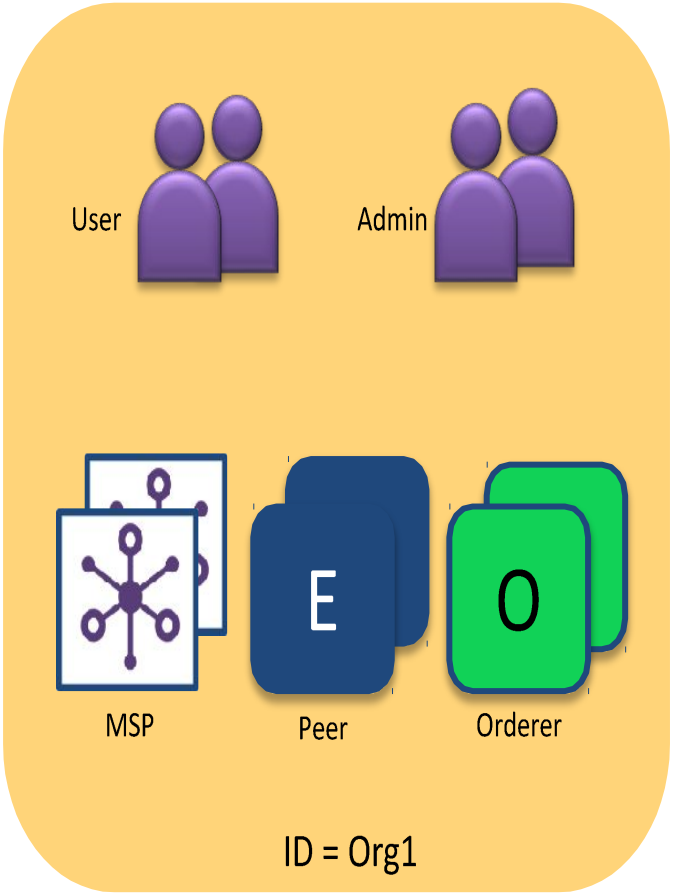
- Default (optional) Certificate Authority within Fabric network for issuing Ecerts (long-term identity)
- Supports clustering for HA characteristics
- Supports LDAP for user authentication
- Supports HSM for security
- Can be configured as an intermediate CA



Organisations

Organisations define boundaries within a Fabric Blockchain Network

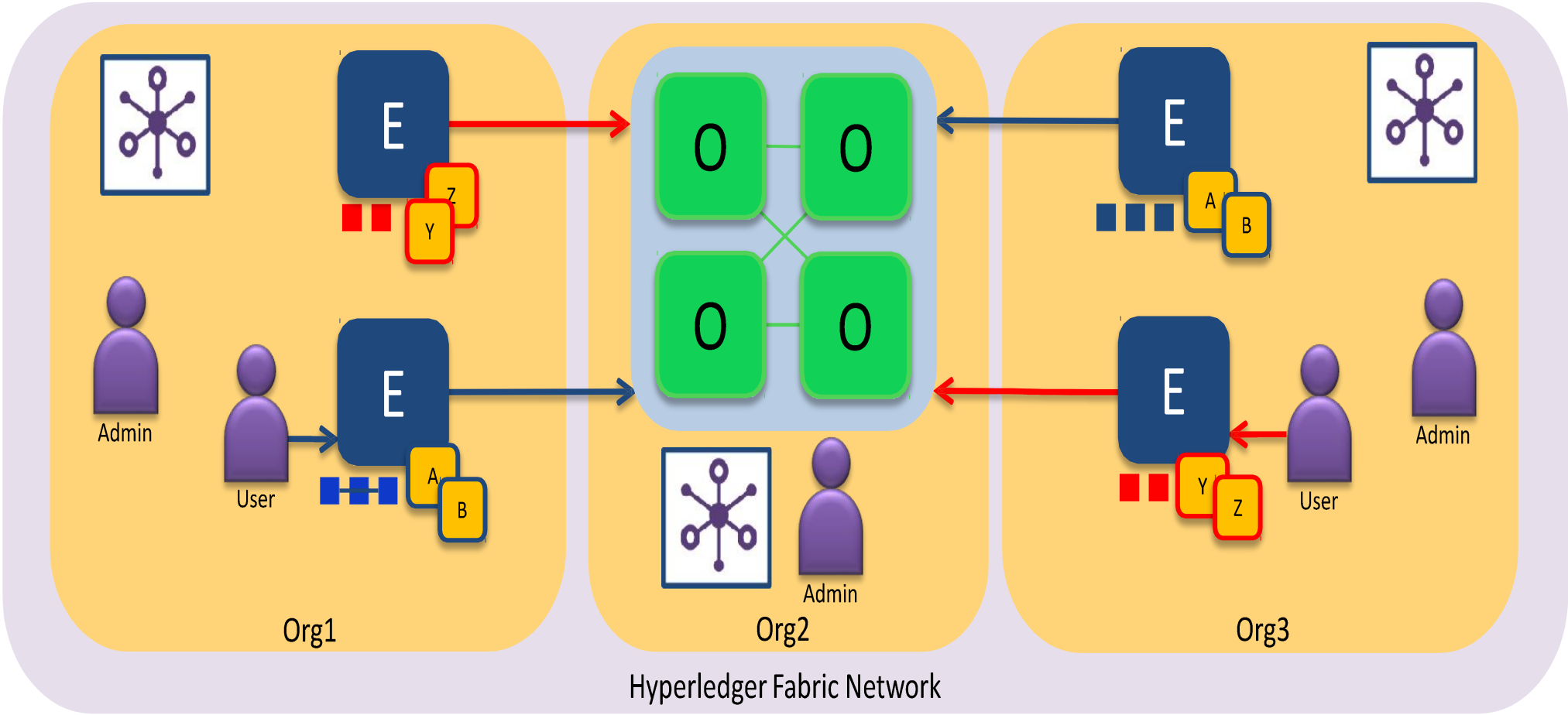
- Each organisation defines:
  - Membership Services Provider (MSP) for identities
  - Administrator(s)
  - Users
  - Peers
  - Orderers (optional)
- A network can include many organisations representing a consortium
- Each organisation has an ID



Consortium Network

An example consortium network of 3 organisations

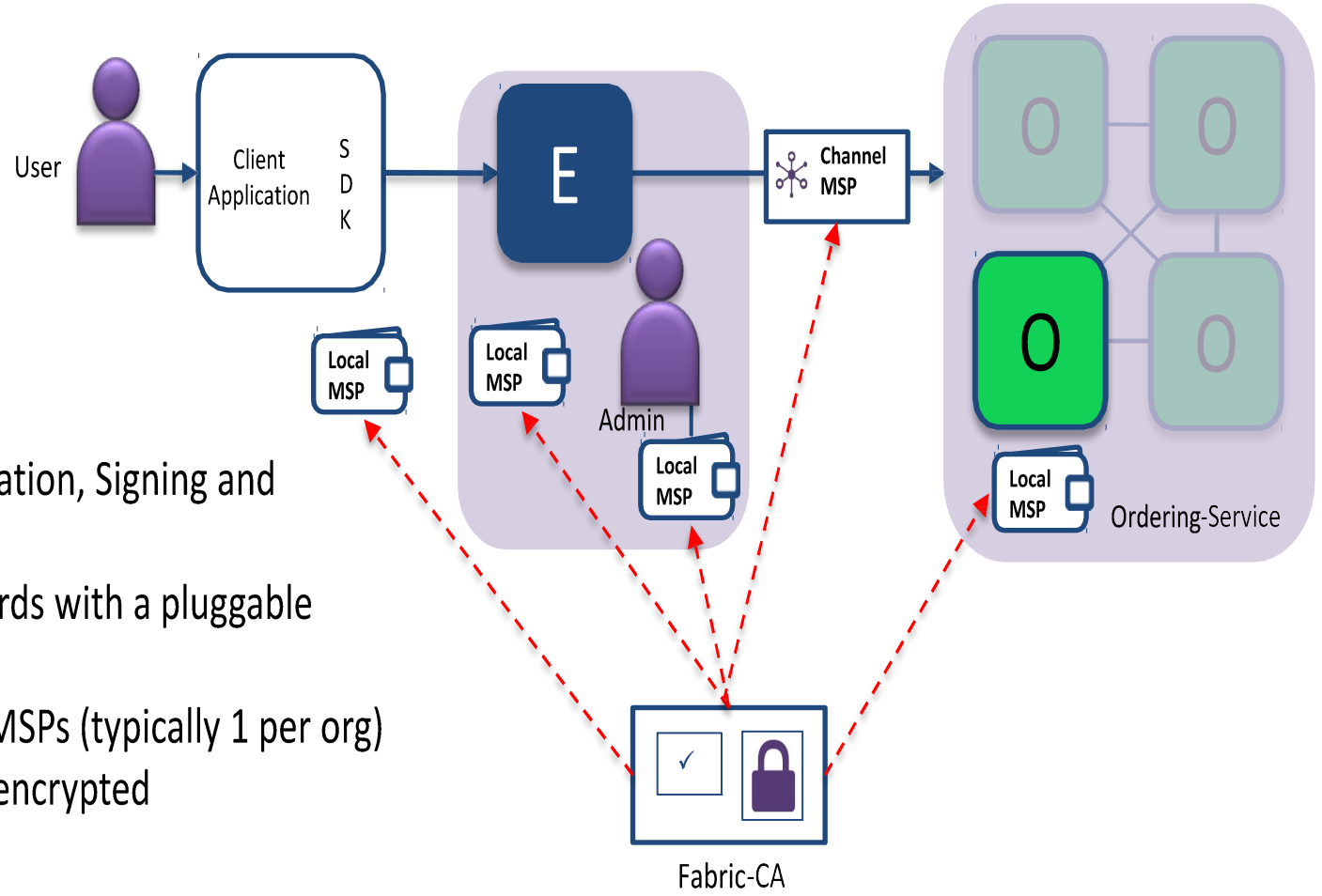
- Orgs 1 and 3 run peers
- Org 2 provides the ordering service only



Membership Service Provider (MSP) - Overview

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

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



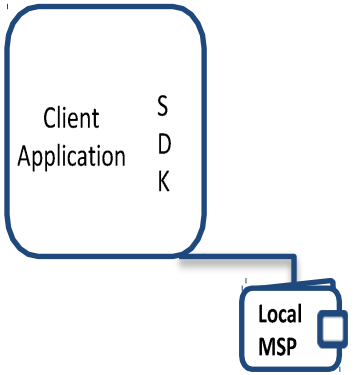
Transport Layer Security (TLS)

- Cryptographic protocols that provide communications security over a computer network
- Provides privacy and data integrity
- Symmetric cryptography is used to encrypt the data transmitted (privacy)
- Public-key cryptography is used to authenticate the identities of the communicating parties
- Include message integrity check to prevent loss or alteration of the data
- All component communication in Fabric secured using TLS (client-peer, peer-peer, peer-orderer, orderer-orderer)

User Identities

Each client application has a local MSP to store user identities

- Each local MSP includes:
  - **Keystore**
    - **Private key** for signing transactions
  - **Signcert**
    - **Public x.509 certificate**



- May also include TLS credentials
- Can be backed by a Hardware Security Module (HSM)

user@org1.example.com	
keystore	<private key>
signcert	user@org1.example.com-cert.pem

Admin Identities

Each Administrator has a local MSP to store their identity

- Each local MSP includes:
  - **Keystore**
    - **Private key** for signing transactions
  - **Signcert**
    - **Public x.509 certificate**

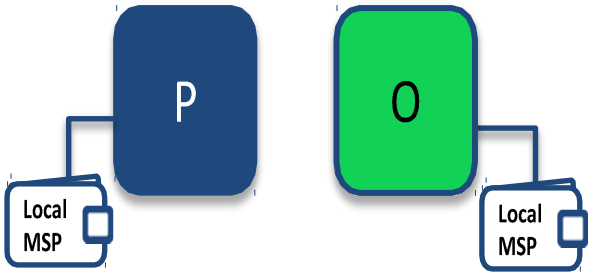
- May also include Transport Layer Security (TLS) credentials
- Can be backed by a Hardware Security Module (HSM)

admin@org1.example.com	
keystore	<private key>
signcert	admin@org1.example.com-cert.pem

Peer and Orderer Identities

Each peer and orderer has a local MSP

- Each local MSP includes:
  - **keystore**
    - **Private key** for signing transactions
  - **signcert**
    - **Public x.509 certificate**



- In addition Peer/Orderer MSPs identify authorized administrators:
  - **admincerts**
    - List of **administrator certificates**
  - **cacerts**
    - The **CA public cert** for verification
  - **crls**
    - List of **revoked certificates**

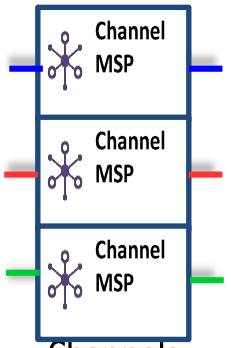
peer@org1.example.com	
admincerts	admin@org1.example.com-cert.pem
cacerts	ca.org1.example.com-cert.pem
keystore	<private key>
signcert	peer@org1.example.com-cert.pem
crls	<list of revoked admin certificates>

- Peers and Orderers also receive channel MSP info
- Can be backed by a Hardware Security Module (HSM)

Channel MSP Information

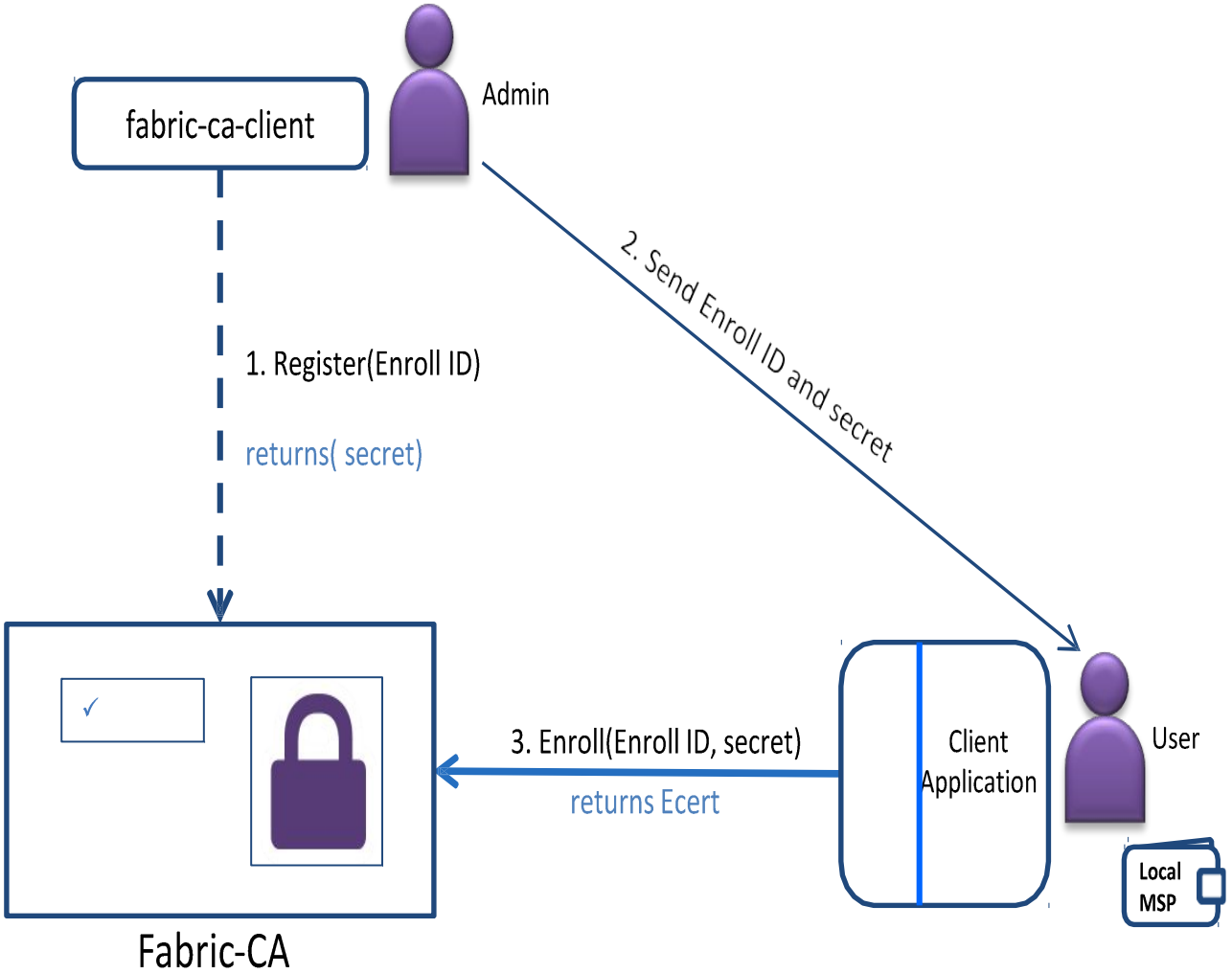
Channels include additional organisational MSP information

- Determines which orderers or peers can join the channel
- Determines client applications read or write access to the channel
- Stored in configuration blocks in the ledger
- Each channel MSP includes:
  - **admincerts**
    - Any public certificates for administrators
  - **cacerts**
    - The CA public certificate for this MSP
  - **crls**
    - List of revoked certificates
- Does not include any private keys for identity



ID = MSP1	
admincerts	admin.org1.example.com-cert.pem
cacerts	ca.org1.example.com-cert.pem
crls	<list of revoked admin certificates>

New User Registration and Enrollment



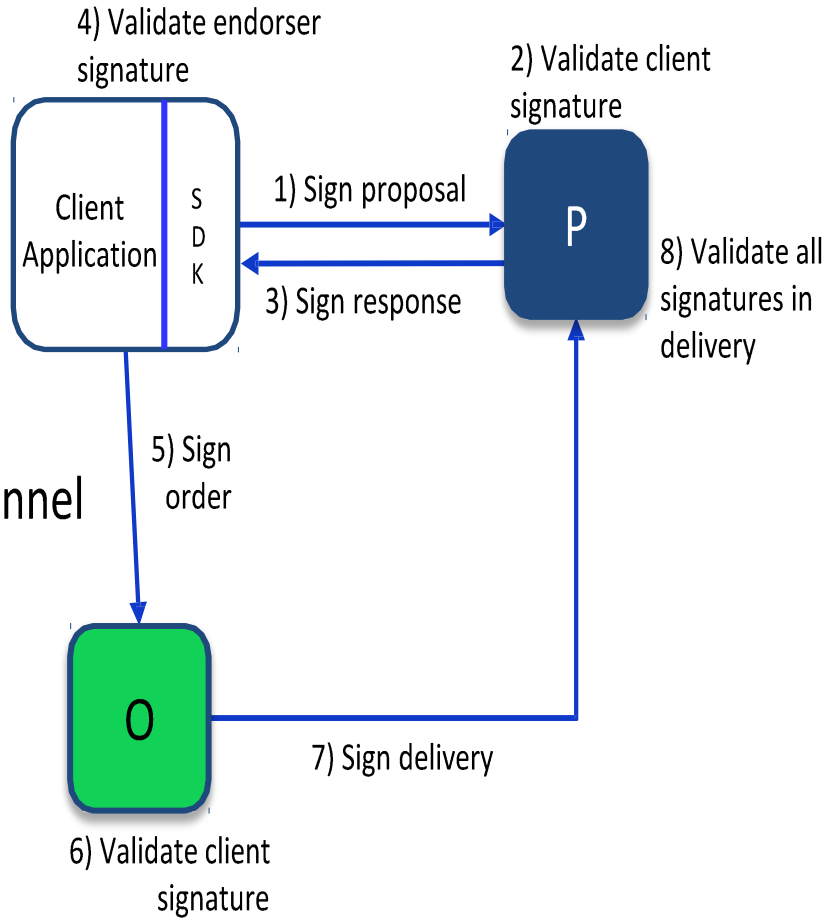
Registration and Enrollment

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

Transaction Signing

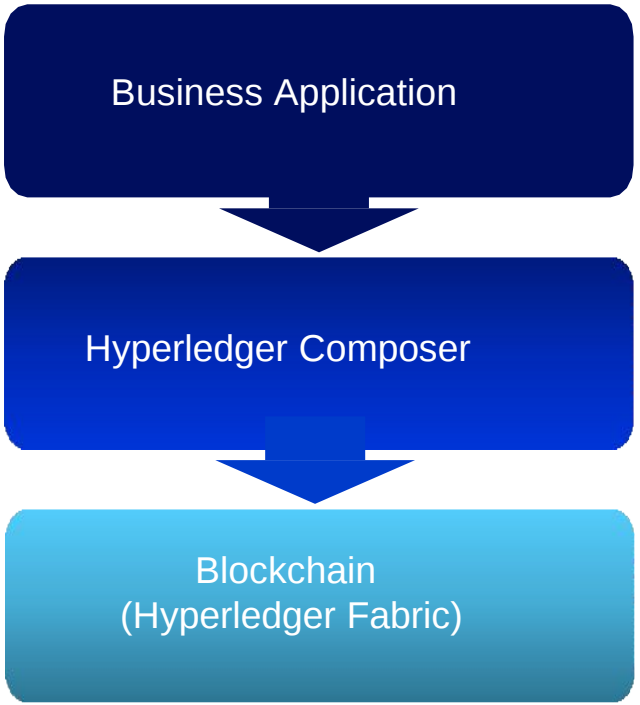
All transactions within a Hyperledger Fabric network are signed by permissioned actors, and those signatures are validated

- Actors sign transactions with their enrolment private key
  - Stored in their local MSP
- Components validate transactions and certificates
  - Root CA certificates and CRLs stored in local MSP
  - Root CA certificates and CRLs stored in Org MSP in channel



HyperLedger Composer

Hyperledger Composer: Accelerating Time to Value



- Emphasis on business-centric vocabulary for quick solution creation – model in terms of assets, participants and transactions
- Features
  - Model your business networks, test and expose via APIs
  - Applications invoke transactions to interact with business network
  - Integrate existing systems of record
- Fully open and part of Linux Foundation Hyperledger
- Try it in your web browser now:
  - http://composer-playground.mybluemix.net/

Goals of Hyperledger Composer

- **Increase understanding**
  - Bridges simply from business concepts to blockchain
- **Save time**
  - Develop blockchain applications more quickly and cheaply
- **Reduce risk**
  - Well tested, efficient design conforms to best practice
- **Increase flexibility**
  - Higher level abstraction makes it easier to iterate

What is Hyperledger Composer?

- A toolset and framework containing a modeling language

- A set of APIs to quickly define and deploy business networks and applications
  - Allow **participants** to send **transactions** that exchange **assets**.

- Quickly model your current business network
  - Define your (existing) assets and the transactions related to them
  - Define business logic a transactions executes

- Supports the existing Hyperledger Fabric infrastructure and runtime

Key Concepts of Hyperledger Composer

- **Assets** can represent almost anything in a business network
  - for example, a house for sale, the sale listing, the land, etc
  - must have a unique identifier, but other whatever properties you define
  - may be *related to* other assets or participants

- **Participants** are members of a business network
  - They may own assets and submit transactions
  - must have a unique identifier, but other whatever properties you define

- **Identities** and **ID cards** can be associate with Participants
  - ID cards are a combination of an identity, a connection profile, and metadata

- **Connection Profiles** are used to connect to a runtime
  - JSON document that lives in the user's home directory
  - Referenced by name when using the Composer APIs or the CLI tools

Key Concepts of Hyperledger Composer Contd..

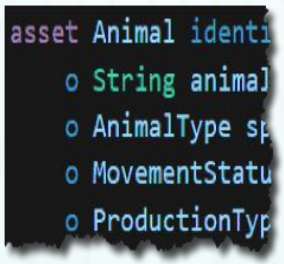
- **Transactions** are the mechanism by which participants interact with assets

- **Queries** are used to return data about the blockchain world-state
  - Queries are defined within a business network
  - Can include variable parameters for simple customization
  - Data can be easily extracted from the blockchain

- **Events** can be emitted by transaction processor functions
  - Notify external systems that something of importance has happened to the ledger

- **Access control rules** allow fine-grained control over
  - What participants have access to what assets and under what conditions

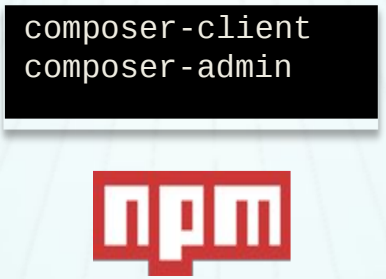
Extensive, Familiar, Open Development Toolset



Data modelling

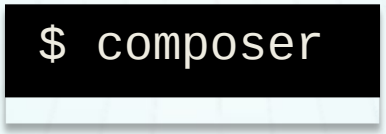
JavaScript
business logic

Web playground



Client libraries

Editor support
(Atom, Visual Studio)

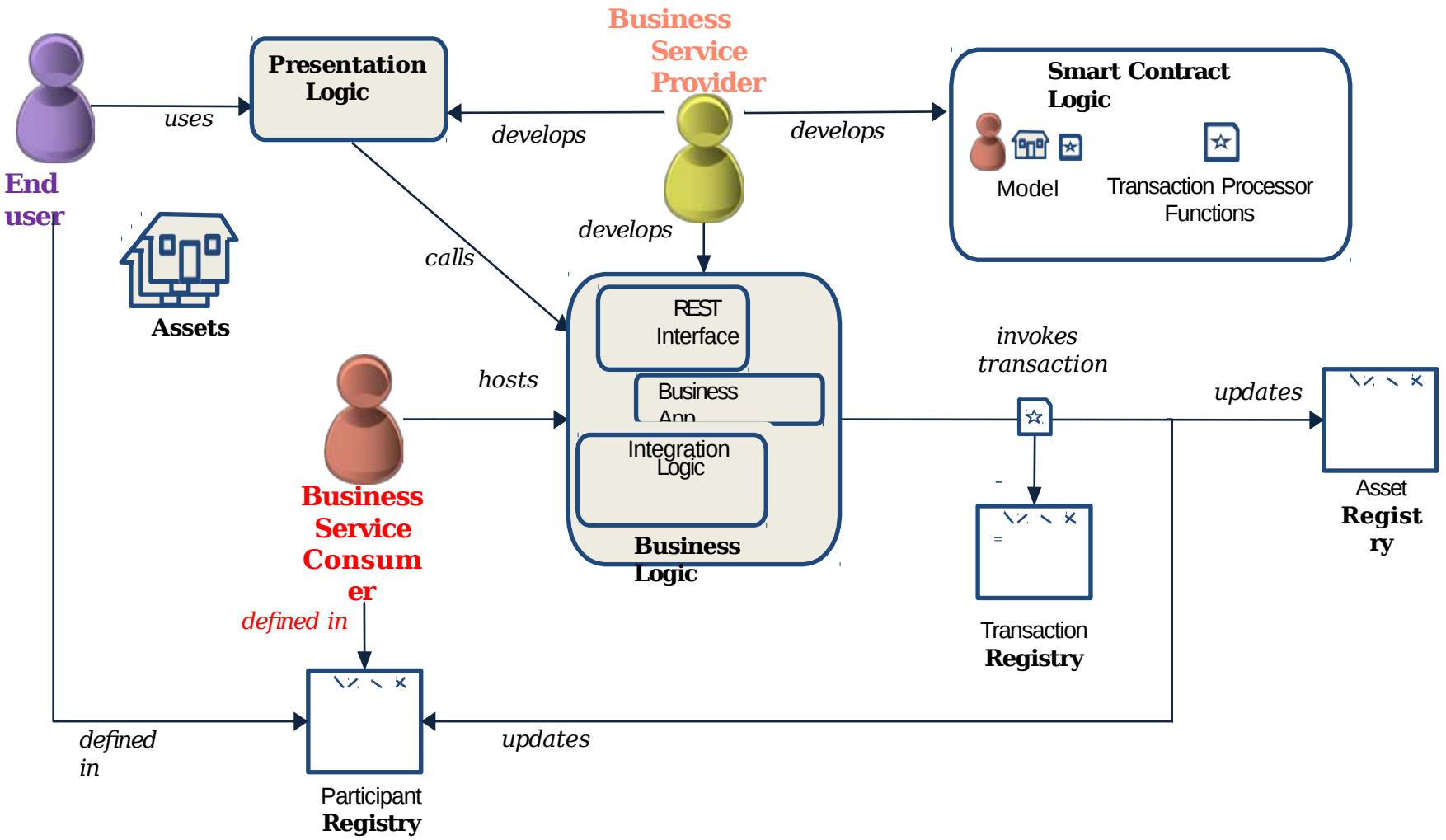


CLI utilities

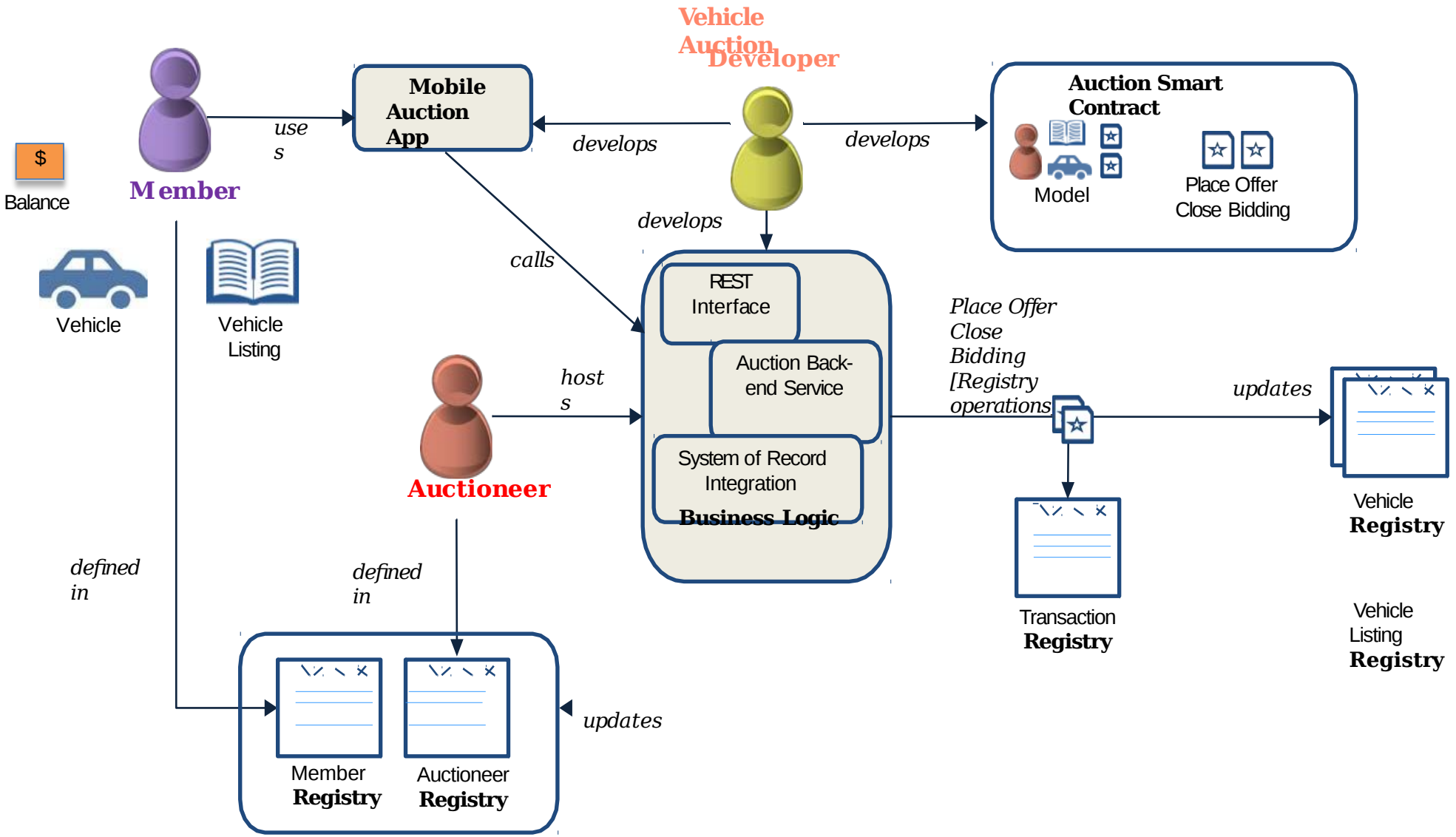
Code generation

Existing systems and
data

Key Concepts for the Business Service Provider



Example: Vehicle Auction Developer



Business Service Provider develops three components

Smart Contracts

- Implements the logic deployed to the blockchain
  - **Models** describe assets, participants & transactions – expressive modeling language includes relationships and validation rules
  - **Transaction processors** provide the JavaScript implementation of transactions
  - **ACLs** define privacy rules
  - May also define events and registry queries

Business Logic

- **Services** that interact with the registries
  - Create, delete, update, query and invoke smart contracts
  - Implemented inside business applications, integration logic and REST services
- Hosted by the Business Application Consumer

Presentation Logic

- Provides the **front-end** for the end-user
  - May be several of these applications
- Interacts with business logic via standard interfaces (e.g. REST)
- Composer can generate the REST interface from model and a sample application

60

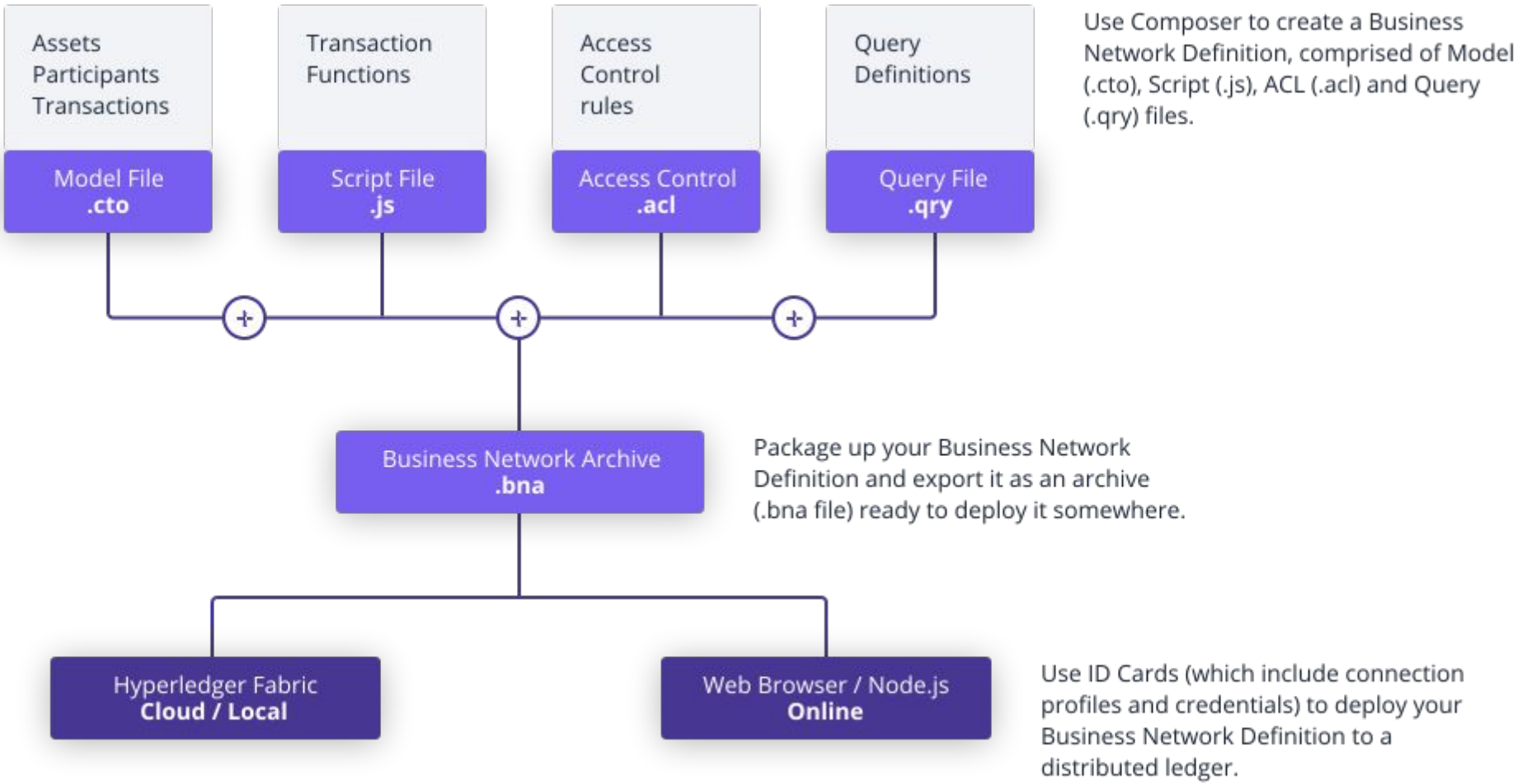


Image Courtesy: hyperledger.github.io/

Questions??