# Introducción a Hyperledger

Carlos Castro-Iragorri

# Blockchain: una conjunto de tecnologías

**Consensus**

PoW, PoS, POET, RaFT, BFT, PBFT

**Crypto/Security**

PKI, HASH, SHA-256, zk-SNARK, HE, ECC, EXDSA, SGX

**Ledger Concepts**

Mining, Blocks, Forks, Parents, Uncles, Merkle Trees

**Platform Concepts**

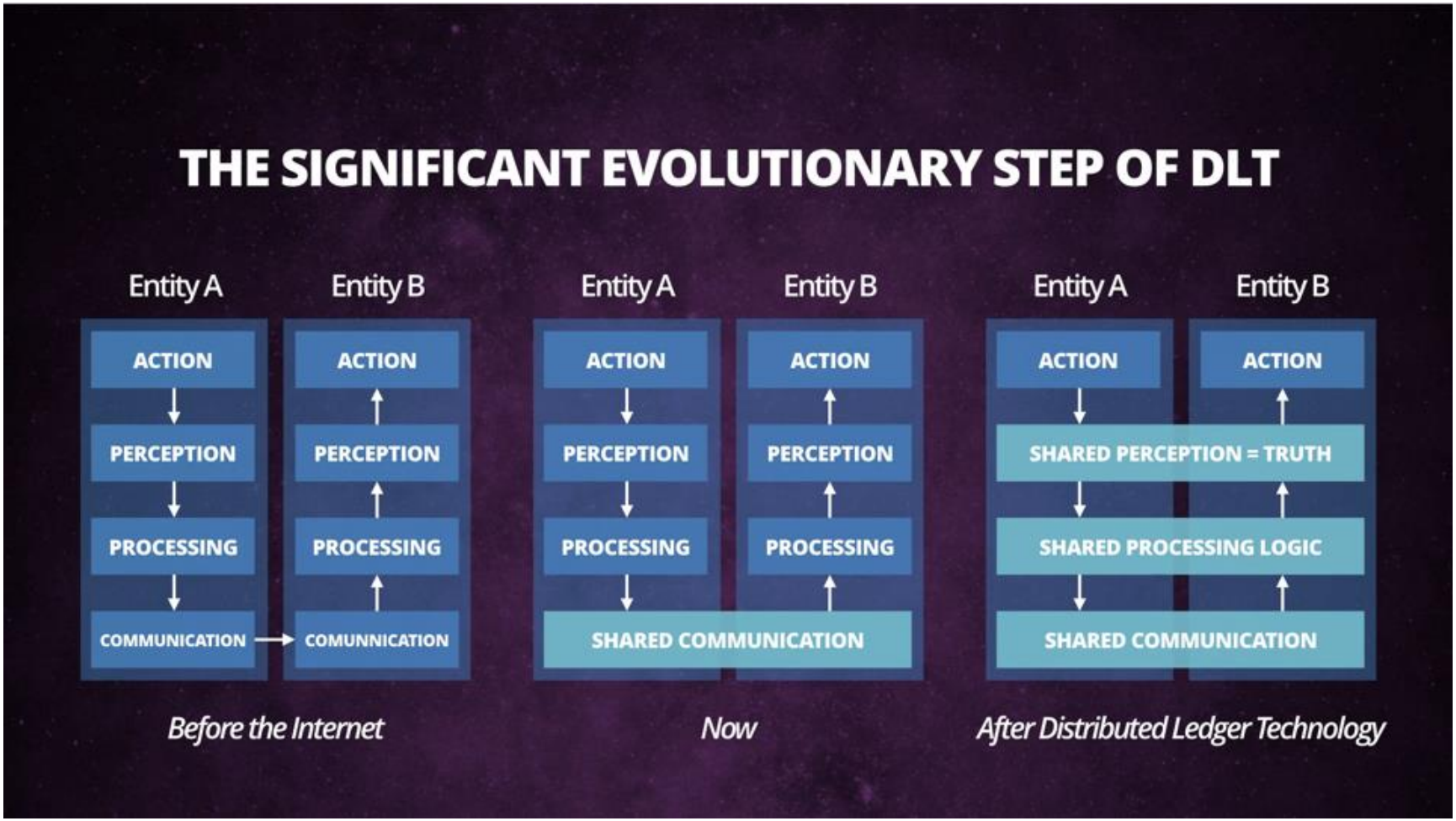
Nodes, Oracles, Notaries, Wallet, Smart Contracts

# Blockchain: estructura conceptual

- Seguridad criptográfica, identificación/autenticidad.
- Registro (base datos) digital compartido, descentralizado, integridad historial de las transacciones. Replicada en tiempo real.
- Arquitectura P2P (sistemas distribuido) vs servidor/cliente.
- Gobernanza: mecanismo de consenso, garantizar consistencia datos entre nodos. Reglas de validación (usuario y transacciones), incentivos,….

La configuración de estos elementos nos lleva a diferentes tipos de tecnologías de registro distribuido y en particular blockchain.

# Evolución DLT/Blockchain



HYPERLEDGER PROJECT

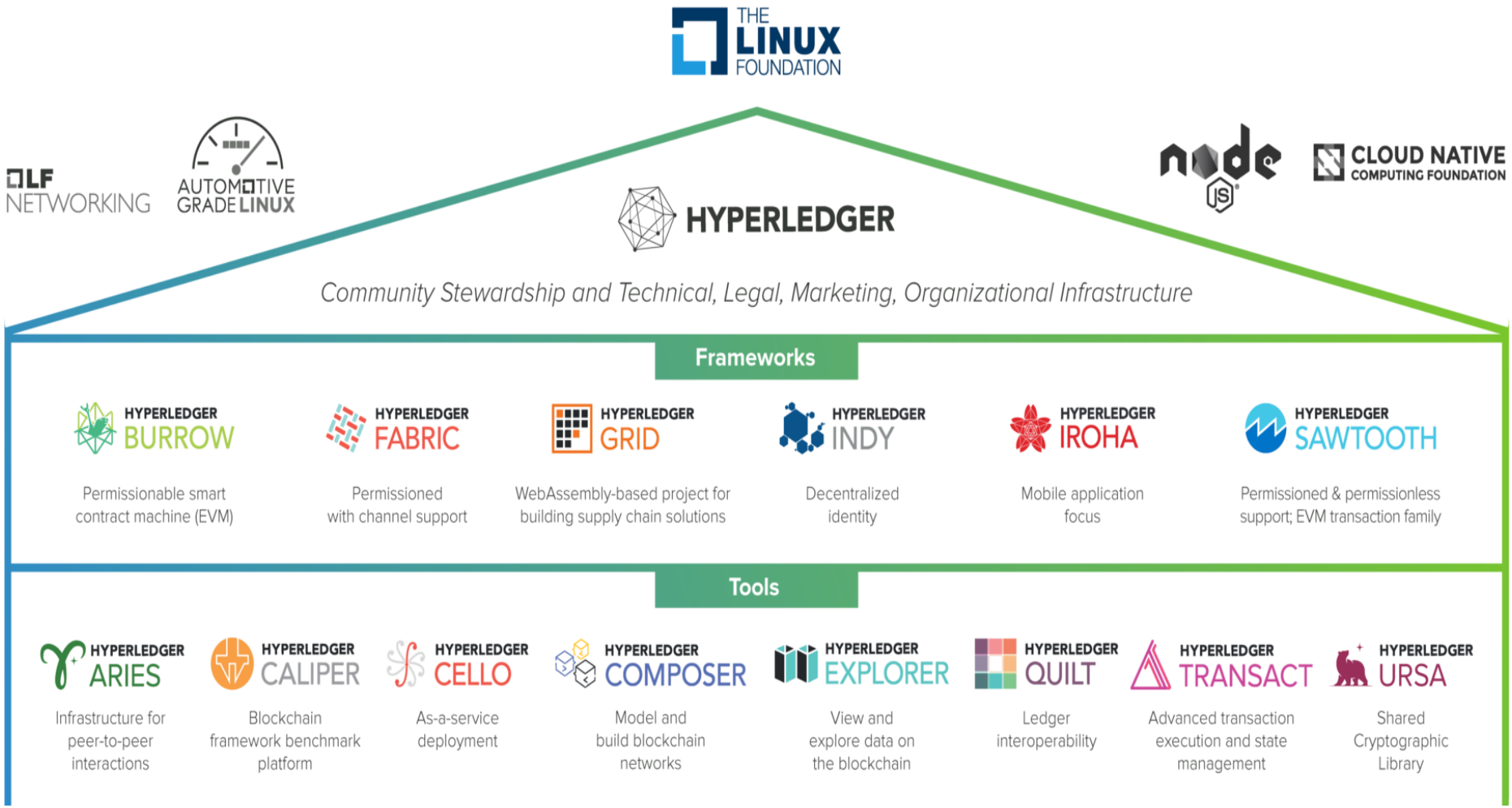
**WWW.HYPERLEDGER.ORG/**

# Proyecto Hyperledger

- Auspiciado por la Fundación Linux
- Open Source
- Contiene:
  - Infraestructura: ecosistema para acelerar desarrollo abierto y adopción comercial.
  - Frameworks: portafolio de soluciones (business blockchains) contribuidos por los miembros.
  - Herramientas: para facilitar el desarrollo.

# Proyecto Hyperledger: Filosofia

# Green house structure



# Frameworks

| | |
|---|---|
| **HYPERLEDGER BURROW** | A modular blockchain client with a permissioned smart contract interpreter developed in part to the specifications of the Ethereum Virtual Machine (EVM). |
| **HYPERLEDGER FABRIC** | A platform for building distributed ledger solutions with a modular architecture that delivers a high degree of confidentiality, flexibility, resiliency, and scalability. This enables solutions developed with Fabric to be adapted for any industry. |
| **HYPERLEDGER INDY** | A distributed ledger that provides tools, libraries, and reusable components purpose-built for decentralized identity. |
| **HYPERLEDGER IROHA** | A blockchain framework designed to be simple and easy to incorporate into enterprise infrastructure projects. |
| **HYPERLEDGER SAWTOOTH** | A modular platform for building, deploying, and running distributed ledgers. Sawtooth features a new type of consensus, proof of elapsed time (PoET) which consumes far fewer resources than proof of work (PoW). |

# Herramientas

| | |
|---|---|
| **HYPERLEDGER CALIPER** | A blockchain benchmark tool that measures the performance of any blockchain by using a set of predefined use cases. |
| **HYPERLEDGER CELLO** | A set of tools to bring the on-demand deployment model to the blockchain ecosystem with automated ways to provision and manage blockchain operations that reduce effort. |
| **HYPERLEDGER COMPOSER** | An open development toolset and framework to make developing blockchain applications easier. |
| **HYPERLEDGER EXPLORER** | A dashboard for viewing information on the network, including blocks, node logs, statistics, smart contracts, and transactions. |
| **HYPERLEDGER QUILT** | A set of tools that offer interoperability by implementing ILP, which is primarily a payments protocol designed to transfer value across distributed and non-distributed ledgers. |

# Nuevos frameworks

- Grid

Supply chain is commonly cited as one of the most promising distributed ledger use-cases. Initiatives focused on building supply chain solutions will benefit from shared, reusable tools. Hyperledger Grid seeks to assemble these shared capabilities in order to accelerate the development of ledger-based solutions for all types of cross-industry supply chain scenarios.

# Nuevas herramientas

- Aries

Aries provides a shared, reusable, interoperable tool kit designed for initiatives and solutions focused on creating, transmitting and storing verifiable digital credentials. It is infrastructure for blockchain-rooted, peer-to-peer interactions. It includes a shared cryptographic +++wallet for blockchain clients as well as a communications protocol for allowing off-ledger interaction between those clients.

- Orsa

Hyperledger Ursa is a shared cryptographic library that would enable people (and projects) to avoid duplicating other cryptographic work and hopefully increase security in the process

# Nuevas herramientas

- Transact

Hyperledger Transact makes writing distributed ledger software easier by providing a shared software library that handles the execution of smart contracts, including all aspects of scheduling, transaction dispatch, and state management.

Transact is fundamentally a transaction processing system for state transitions. State data is generally stored, for example in an SQL database. Given an initial state and a transaction, Transact executes the transaction to produce a new state. These state transitions are considered "pure" because only the initial state and the transaction are used as input. (In contrast, other systems such as Ethereum combine state and block information to produce the new state.)

Transact deliberately omits other features such as consensus, blocks, chaining, and peering. These features are the responsibility of the Hyperledger frameworks (such as Sawtooth and Fabric) and other distributed ledger implementations. The focus on smart contract execution means that Transact can be used for smart contract execution without conflicting with other platform-level architectural design elements.

# ASPECTOS DIFERENCIADORES EN HYPERLEDGER

# Tipos de Blockchain: permisos

- Sin permisos (Permissionless), publica.
- Con permisos (Permissioned), privada.
- Con respecto a escritura y/o lectura en el registro.

# Sin permisos, Blockchain publica

- Transacciones procesadas por todos los nodos.
- Transacciones son completamente visibles (lectura abierta).
- Gran cantidad de nodos
  - Ethereum: 13,978
  - Bitcoin: 9,563.
- Beneficios: escritura y lectura abierta, distribución autentica del registro, resistente a la censura, autenticidad del registro garantizado por la regla de minado (>51%)

# Con permisos, Blockchain privada

- Transacciones procesadas por algunos nodos (especialización de nodos).
- Transacciones pueden ser visibles o privadas.
- Distribución local de la red: dentro de una(s) organización(s).
- Beneficios: Empresas u organizaciones quieren guardar control sobre su información y transacciones, transacciones mas rápidas, mejor escalabilidad, soporte, consenso eficiente.

# Blockchain Privadas vs Publicas

| | Public (Permissionless) | Private (Permissioned) |
|---|---|---|
| Access to Ledger | Open Read/Write | Permissioned Read/Write |
| Identity | Anonymous | Known Identities |
| Security and Trust | Open Network (Trust Free) | Controlled Network(Trusted) |
| Transaction Speed | Slower | Faster |
| Consensus | POW/POS | Proprietary or Modular |
| Open Source | Yes | Depends on Blockchain |
| Code Upkeep | Public | Consortium or Managed |
| Examples | Ethereum, Multichain | R3 Corda, Quantum, Hyperledger |

# Gobernanza / Consenso

- El consenso es un mecanismo de alcanzar algún acuerdo entre un grupo.
- En blockchain el consenso es sobre la información consignada en el registro, “World State”.
- Los mecanismos de consenso implican unas reglas de juego y unos incentivos.
- Hay varios diseños de este tipo de mecanismos.

# Consenso Hyperledger modular

- Hyperledger ofrece un mecanismo de concenso: Plenum Byzantine Fault Tolerance (PBFT):
  - En PBFT cada noda mantiene una copia del registro.
  - Cuando un nodo recibe un mensaje (Kafka), lo firma para verificar su autenticidad.
  - En el momento en que una cantidad suficiente de mensajes se reciben con las misma características entonces se alcanza el consenso y la transacción es valida.

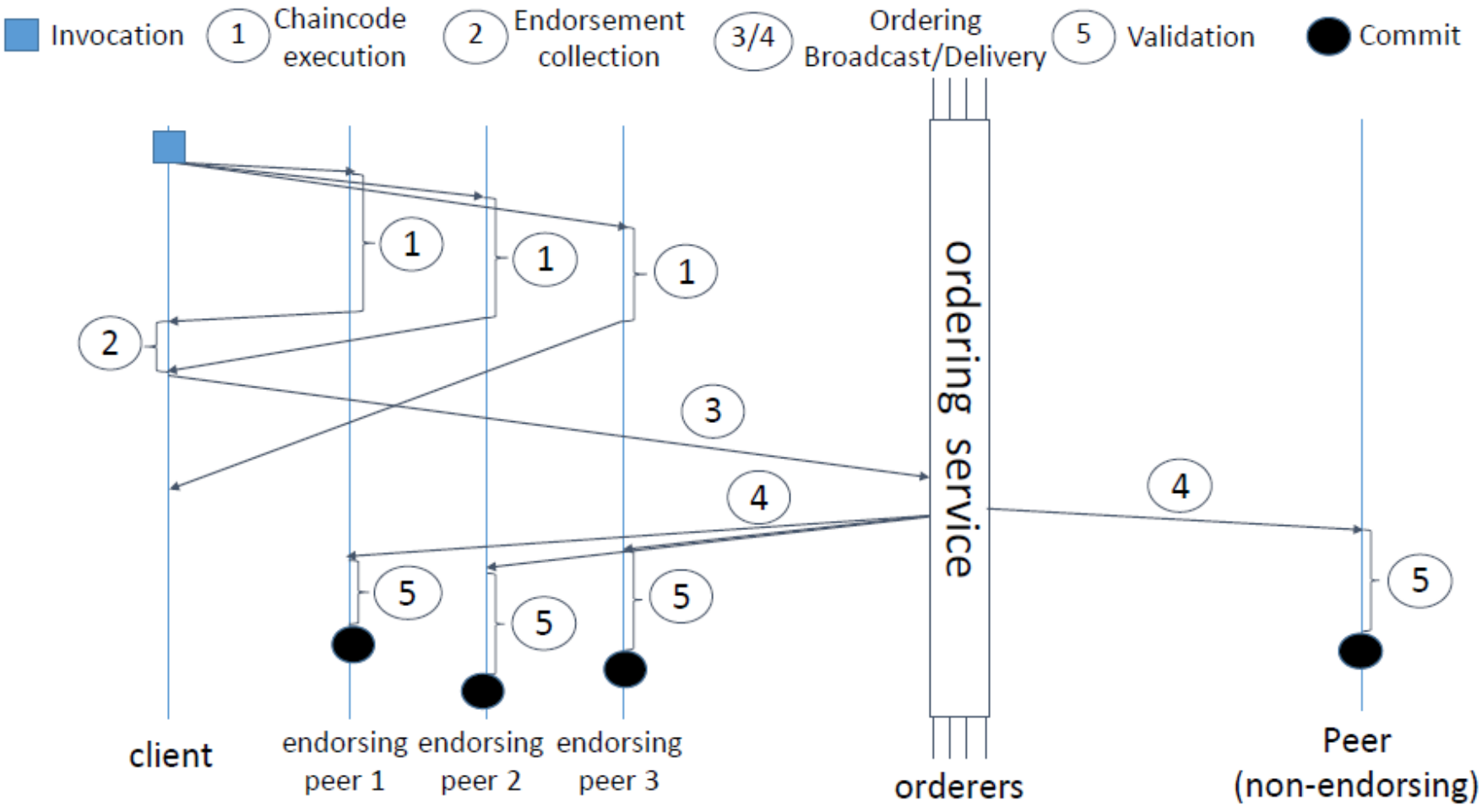
# Consenso Hyperledger

- Hyperledger utiliza un mecanismo de consenso basado en los votos o mensajes que recibe de aquellos nodos que se les permite participar en el mecanismo.
- Los algoritmos de consensos basados en sistemas de votación proporcionan un procesamiento alto de mensajes con mínimo rezago (low-latency).
- Sin embargo hay un *trade-off* entre escalabilidad y desempeño, ya que mas nodos (votantes) implica mayor tiempo para alcanzar un concenso.

# Consenso Hyperledger

- Tiene tres fases:
- Endorsment: regla de juego: m de n firmas a partir de las cuales los participantes apoyan una transaccion.
- Ordering: recolecta las transacciones apoyadas y determina el orden en que deben ser registradas.
- Validation: analiza y valida los bloques.

# Consenso Hyperledger



# HYPERLEDGER FABRIC

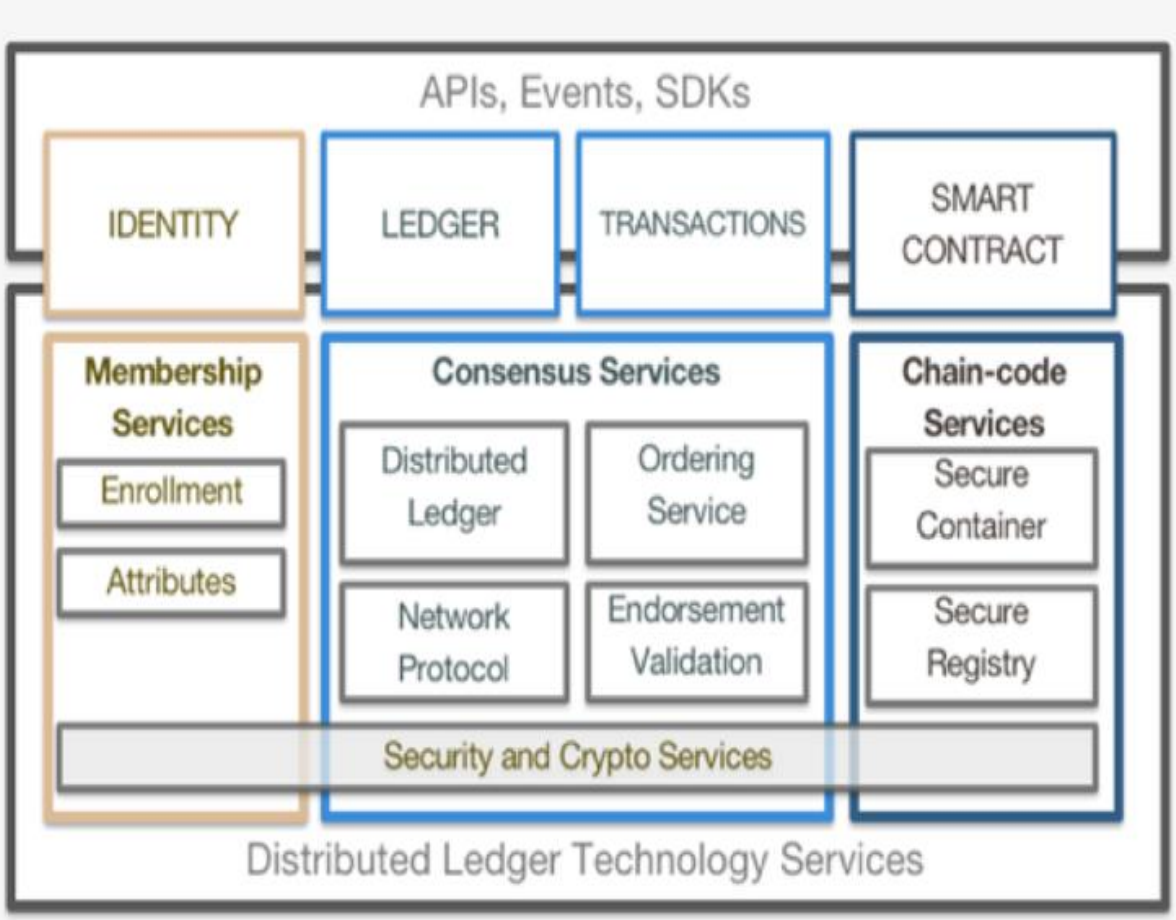
# Hyperledger Fabric

- Implementacion de blockchain diseñada para soportar despliegues modulares y una arquitectura escalable a las necesidades de la empresa.
- Su estructura modular permite que diferentes implementaciones se puedan conectar e implementar a lo largo del tiempo.

# Modularidad en Fabric:

- Solución con Permisos
- Varios protocolos de Consenso
- Smart Contracts (chaincode)
- Comunicación (canales)
- API’s
- No esta diseñada para operar con criptomoneda o token
- Servicios de identidad (membership service) y seguridad. Varios sistemas de registro de la información (diferentes niveles de complejidad en la busqueda).
- Interoperabilidad.

# Reference Arquitecture




**IDENTITY**
Pluggable, Membership, Privacy and Auditability of transactions.

**LEDGER | TRANSACTIONS**
Distributed transactional ledger whose state is updated by consensus of stakeholders

**SMART CONTRACT**
"Programmable Ledger", provide ability to run business logic against the blockchain (aka smart contract)

**APIs, Events, SDKs**
Multi-language native SDKs allow developers to write DLT apps

# Arquitectura de la red: los nodos

- Los nodos de la red utilizan un mecanismo de comunicación peer-to-peer para mantener el registro actualizado.
- La red esta compuesta por nodos (especializados).
- Estos nodos deben contar con un certificado (permiso) valido para interactuar en la red.
- El certificado de cada uno de los nodos se utiliza para firman las transacciones que procesa.

# Tipos de nodos: cliente

- Client: inician una transacción. Se deben conectar con un nodo peer para interactuar con el blockchain.
- Se puede conectar con cualquier peer.
- Inicia e invoca las transacciones (chain code).

# Tipos de nodos: Peer

**Committing Peer**

- Maintains ledger and state
- Commits transactions
- May hold smart contract (chaincode)

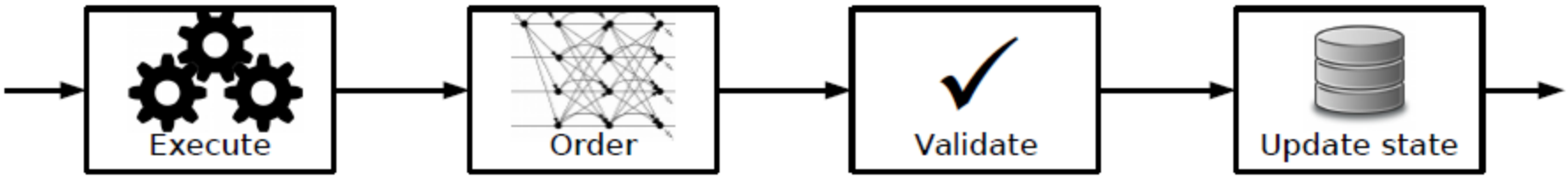
**Endorsing Peer**

- Receives a transaction proposal for endorsement, responds granting or denying endorsement
- Must hold smart contract
- Verifies that its content obeys a given smart contract
- Endorser “signs” the contract

**Ordering Node**

- Approves the inclusion of transaction blocks into the ledger and communicates with committing and endorsing peer nodes
- Controls what goes in the ledger making sure that the ledger is consistent
- Does not hold smart contract
- Does not hold ledger

# Tipos de nodos: Peer



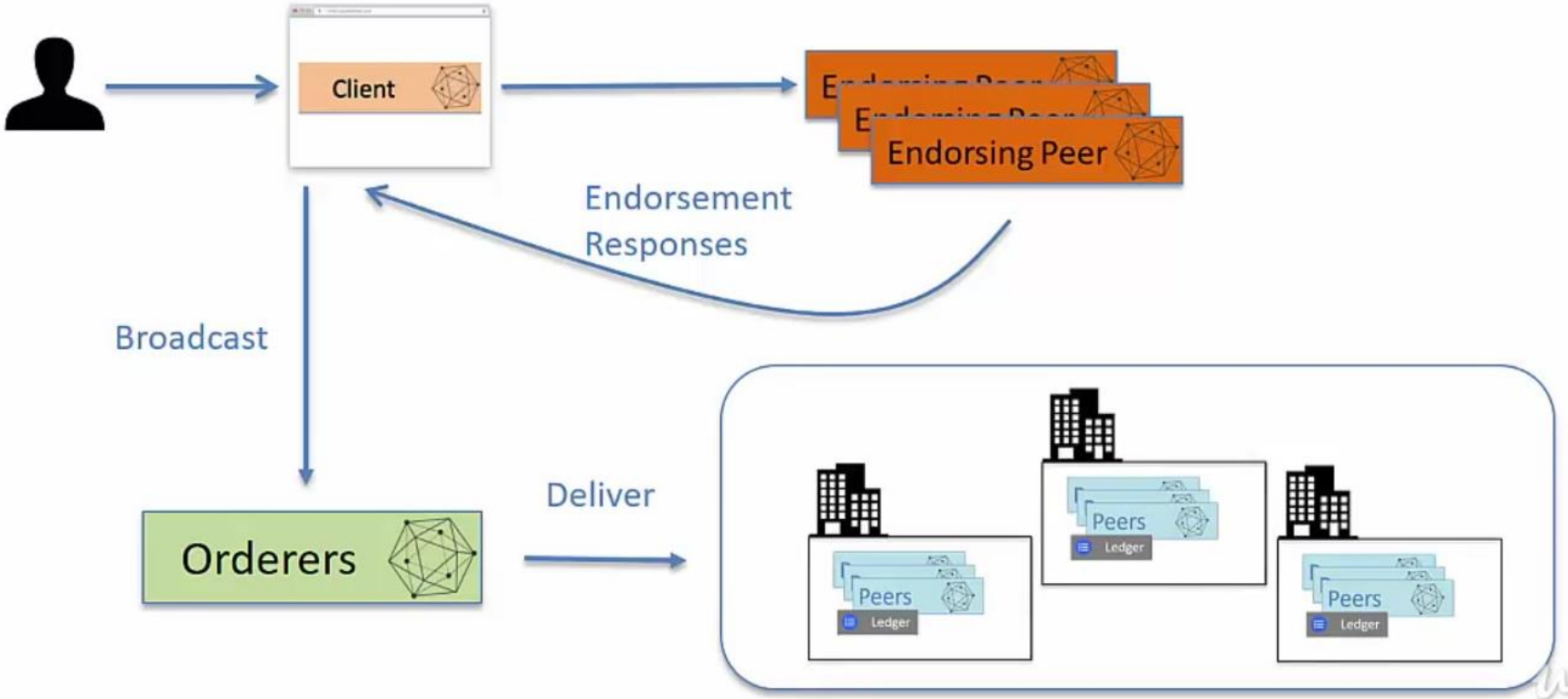

- Simulate trans. and endorse
- Create rw-set
- Collect endorse-ments

- Order rw-sets
- Atomic broadcast (consensus)
- Stateless ordering service

- Validate endorse-ments & rw-sets
- Eliminate invalid and conflicting trans.

- Persist state on all peers

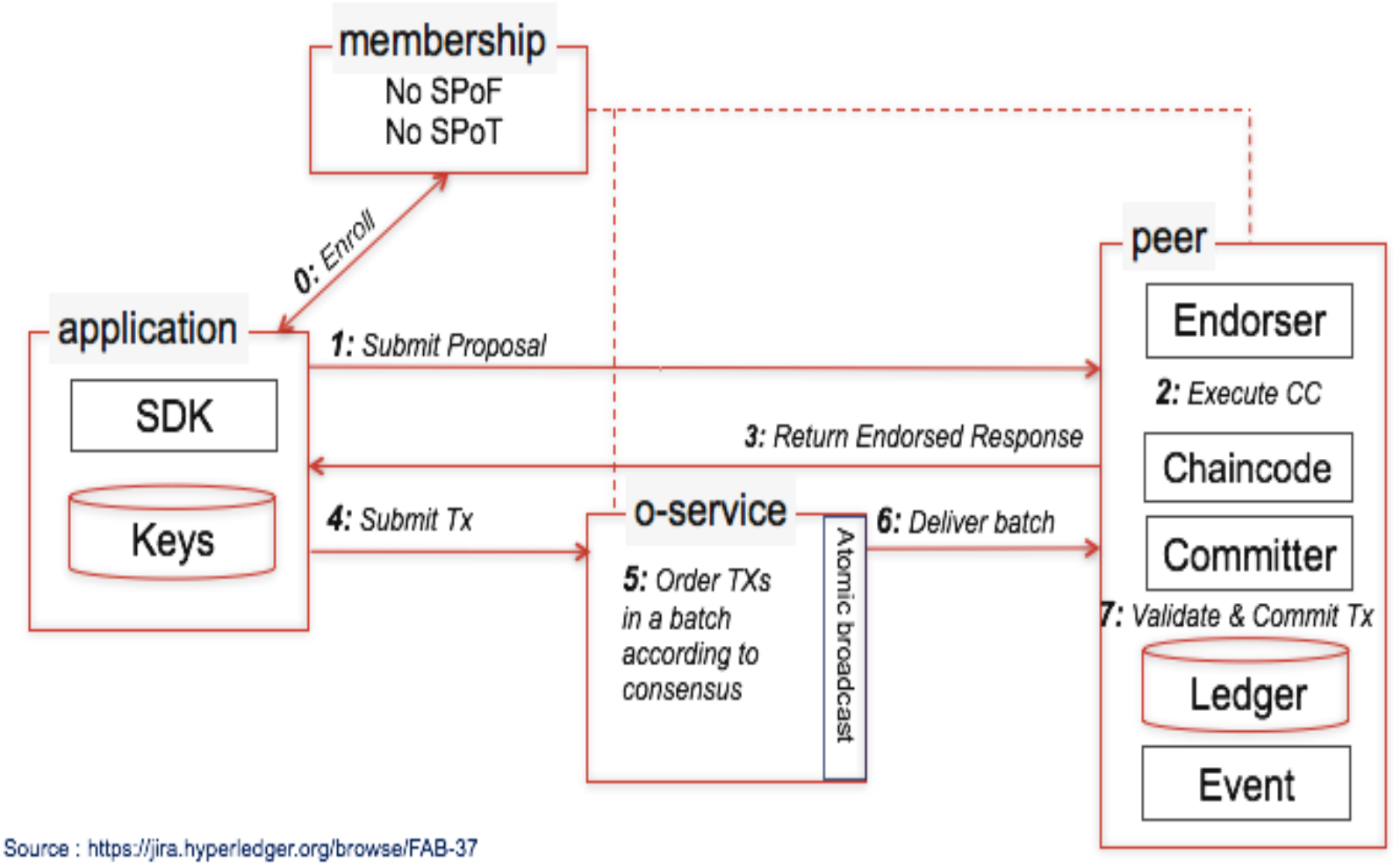
# Interacción entre nodos



# Transacciones

- Fabric permite definir políticas con respecto a la ejecución de las transacciones (chain code)
- Endorcement policies definen cuales peer deben estar de acuerdo con el resultado de la transacción antes de que se adjunte al registro. Por ejemplo los peers A,B y C deben aprobar la transacción de tipo X.
- Endorcement policies utiliza un lenguaje especifico utilizado en la configuración de las transacciones.
- 3500 tps, 100 peers, Adroulaki et al. (2018)

# Flujo de transacciones




Source : https://jira.hyperledger.org/browse/FAB-37

# Transacciones por segundo, VISA: 24,000

| | Block Generation Time | Transactions Per Second (tps)[23] |
|---|---|---|
| **Bitcoin** | 10 minutes | Average 3 tps (Max: 7 tps) |
| **Corda** | n/a | > 500 tps |
| **Ethereum** | 10-19 seconds | Average 15-20 tps, but no theoretical limit |
| **Fabric** | variable | > 10 tps |
| **Multichain** | Configurable (≥ 2 seconds) | Configurable |
| **Neo** | 15 seconds | 10,000 tps |
| **NXT** | 1 minute | 12 tps |
| **Quorum** | 50 mSec | >500 tps |
| **Sawtooth** | Configurable | >500 tps |

Mercy Corps (2018). BLOCK BY BLOCK A Comparative Analysis of the Leading Distributed Ledgers

**University of Waterloo (Mayo 3, 2019) Hyperledger Fabric blockchain from 3,000 to 20,000 transactions per second (TPS).**

# Fabric: sistema de registro

El ledger es una secuencia de registros (resistentes a la manipulación) de los cambios en el estado de los elementos de interés (**activos**).

Los cambios en el estado se producen a través de las transacciones que invocan los **participantes**.

# Fabric: The Ledger

1. State data (operaciones CRUD): representa el estado actual de los activos. El estado cambia en función de las transacciones que operan sobre los activos.
2. Transaction log (inmutable): Registro de todas las transacciones que modifican el State data.

Create, Read, Update and Delete

# Propiedades Base de Datos

| | Transaction Logs | State Date (World) |
|---|---|---|
| Type | Immutable | Mutable |
| Operations | Create, Retrieve | ALL-CRUD |
| DC | levelDB | levelDB/CouchDB |
| Attitude | Embedded in peers | Key-Value Paired(JSON, Binary) |
| Query | Simple | Couch DB for Complex |

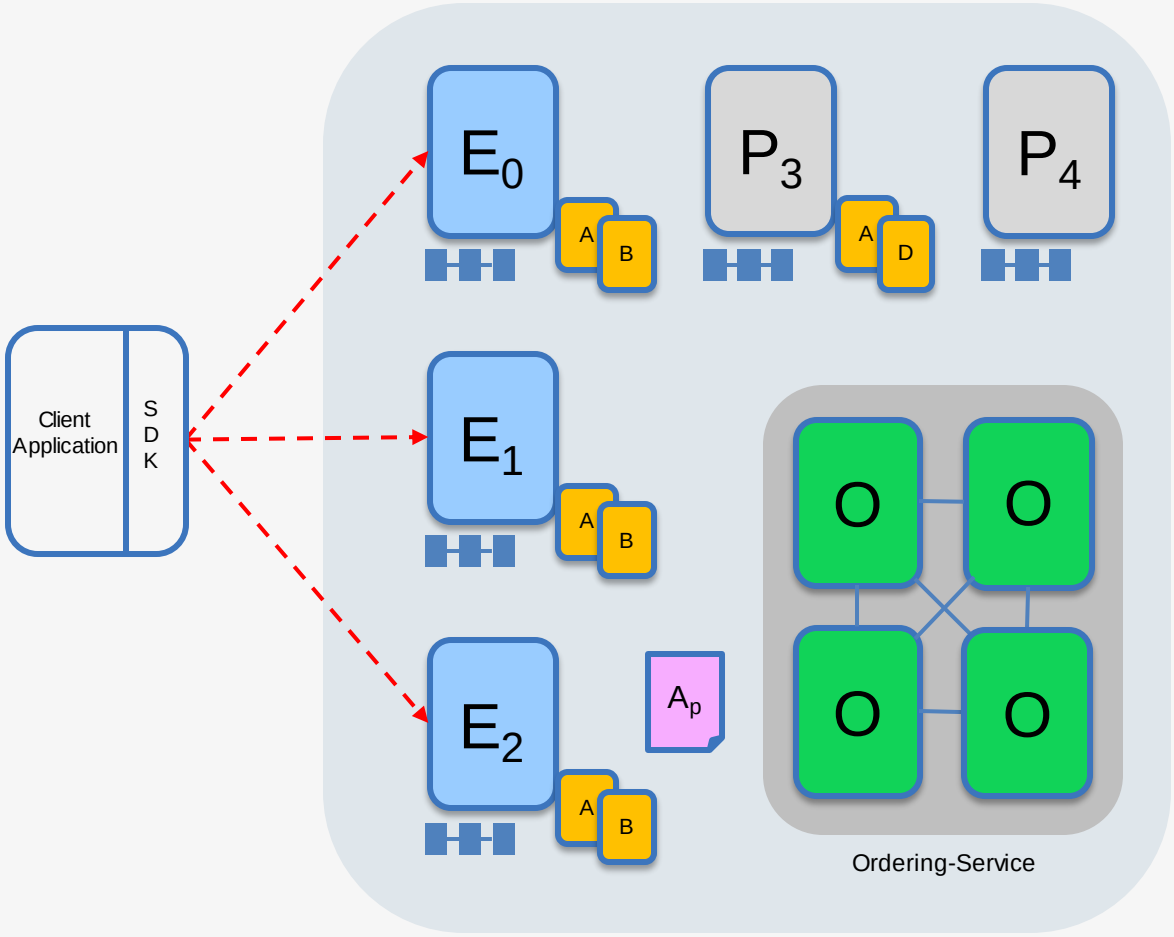
# Utilización de Fabric-UHLevel

- Instalación de pre-requisitos y descargar las imágenes Docker de Fabric
- El diseño de la arquitectura se hace a través de archivo de configuración (*.yml).
  - Configuración de artefactos (nodos y su especialización).
  - Componente criptográfico.
- Iniciar la red, crear canal de comunicación, creación de tarjetas de administrador (red y nodo), desplegar e iniciar BNA.

# Anexo

(Hyperledger, 2017)

# Sample transaction: Step 1/7 – Propose transaction



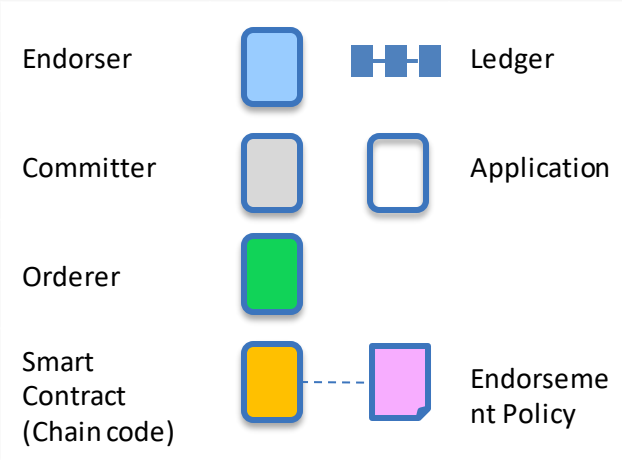

Hyperledger Fabric

**Application proposes transaction**

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

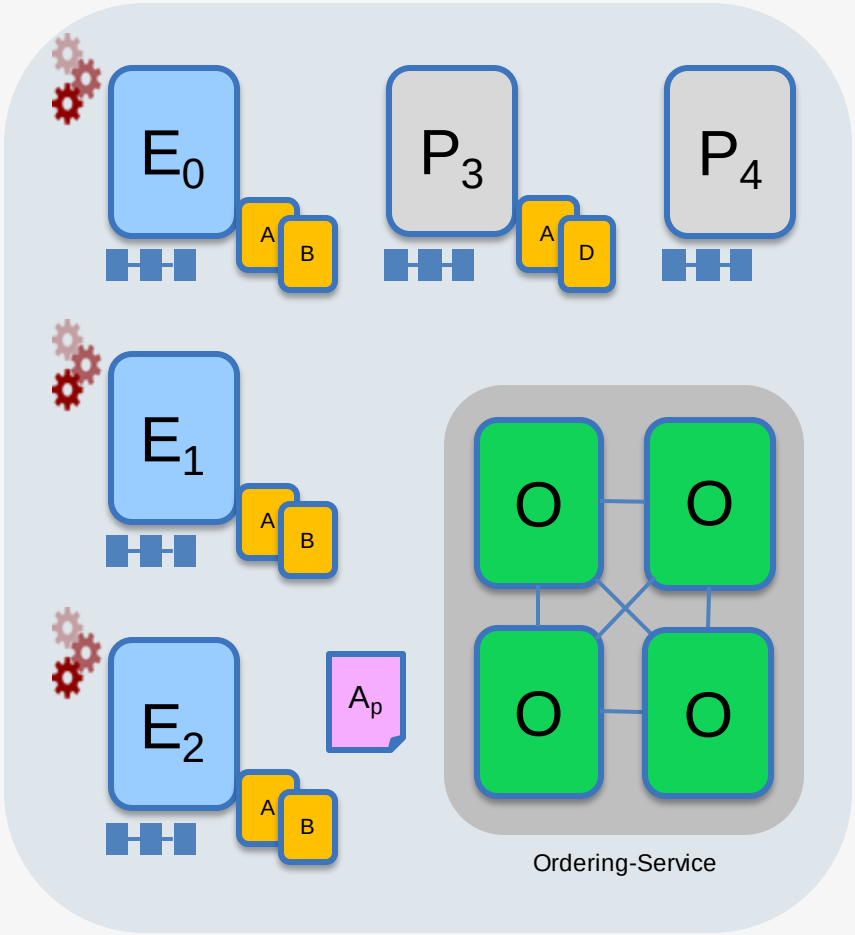
Client application submits a transaction proposal for **chaincode A.** It must target the required peers {**$E_0$, $E_1$, $E_2$**}

**Key:**




HYPERLEDGER
BLOCKCHAIN TECHNOLOGIES FOR BUSINESS

# Sample transaction: Step 2/7 – Execute proposal




Client Application | SDK

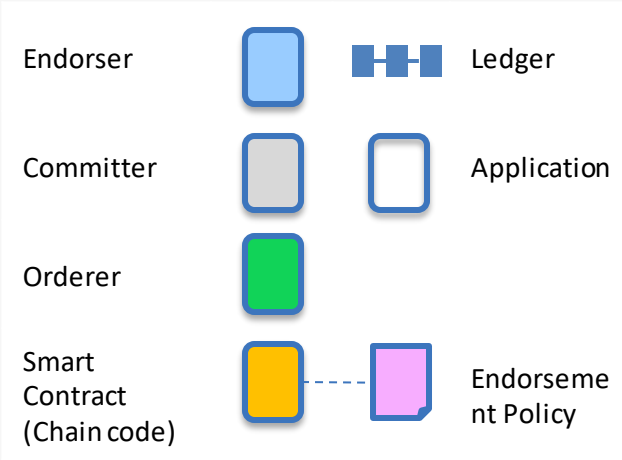
Hyperledger Fabric

## Endorsers Execute Proposals

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

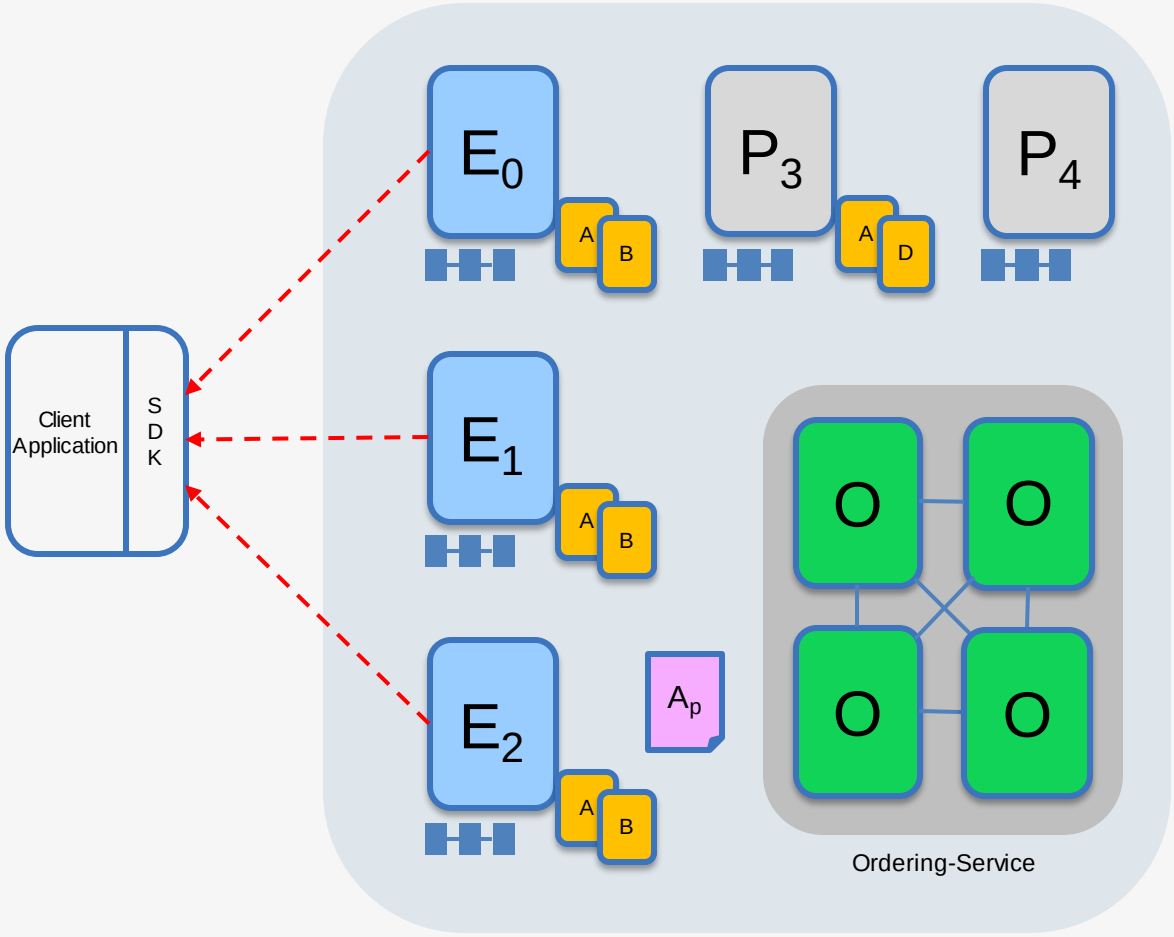
Each execution will capture the set of **R**ead and **W**ritten data, called **RW sets,** which will now flow in the fabric.

**Key:**




HYPERLEDGER
BLOCKCHAIN TECHNOLOGIES FOR BUSINESS

# Sample transaction: Step 3/7 – Proposal Response

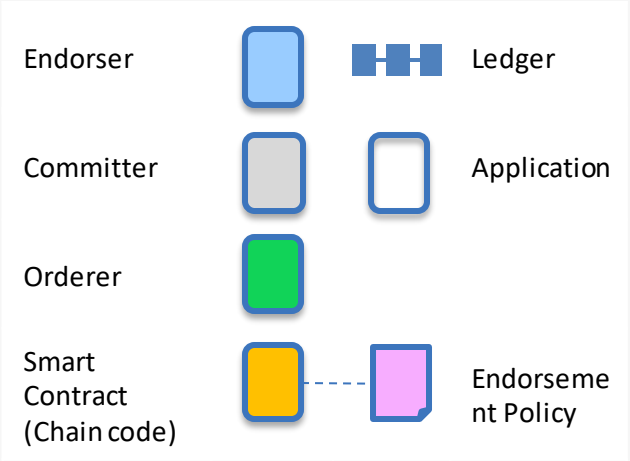



Hyperledger Fabric

**Application receives responses**

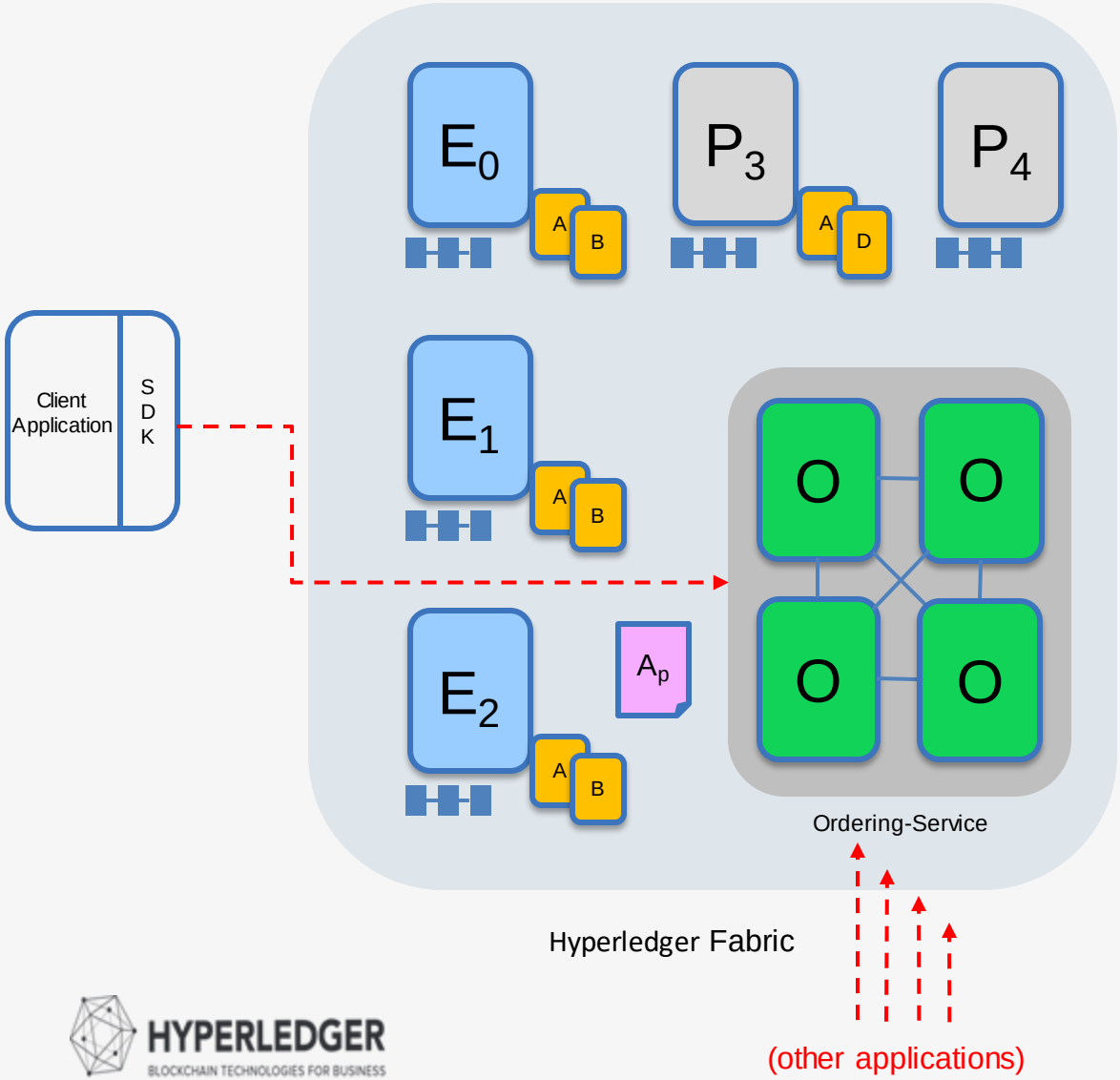
The RW sets are signed by each endorser and returned to the application

**Key:**




HYPERLEDGER
BLOCKCHAIN TECHNOLOGIES FOR BUSINESS

# Sample transaction: Step 4/7 – Order Transaction

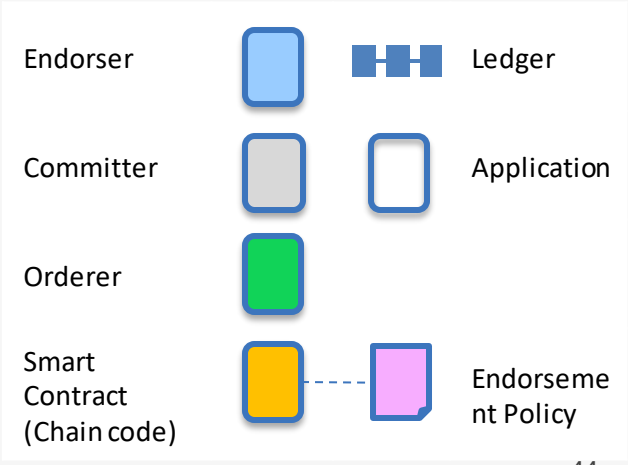



**Application submits responses for ordering**

Application submits responses as a **transaction** to be ordered.

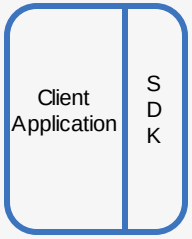
Ordering happens across the fabric in parallel with transactions submitted by other applications

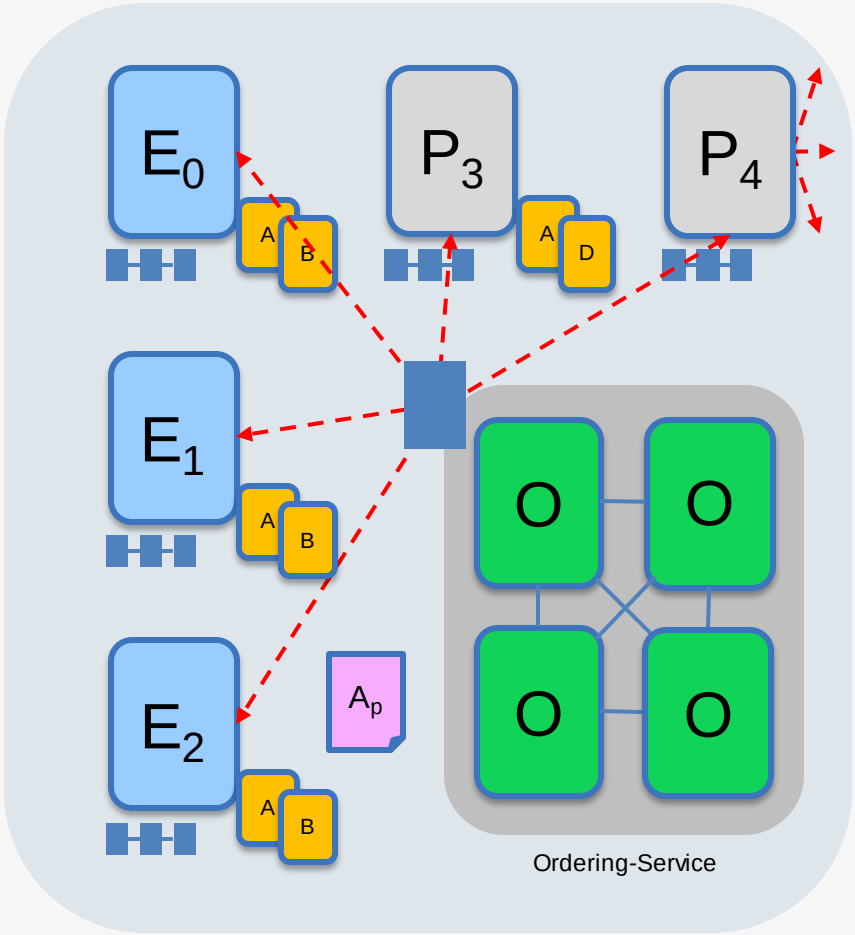
**Key:**




HYPERLEDGER
BLOCKCHAIN TECHNOLOGIES FOR BUSINESS

# Sample transaction: Step 5/7 – Deliver Transaction




Hyperledger Fabric

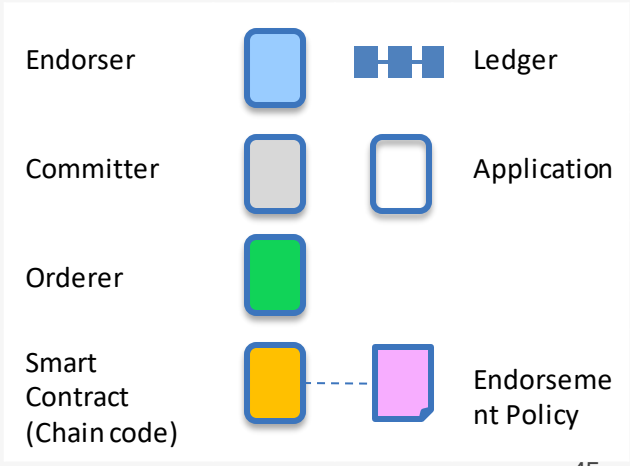
**Orderer delivers to all committing peers**

Ordering service collects transactions into blocks for distribution to committing peers. Peers can deliver to other peers using gossip (not shown)

Different ordering algorithms available:

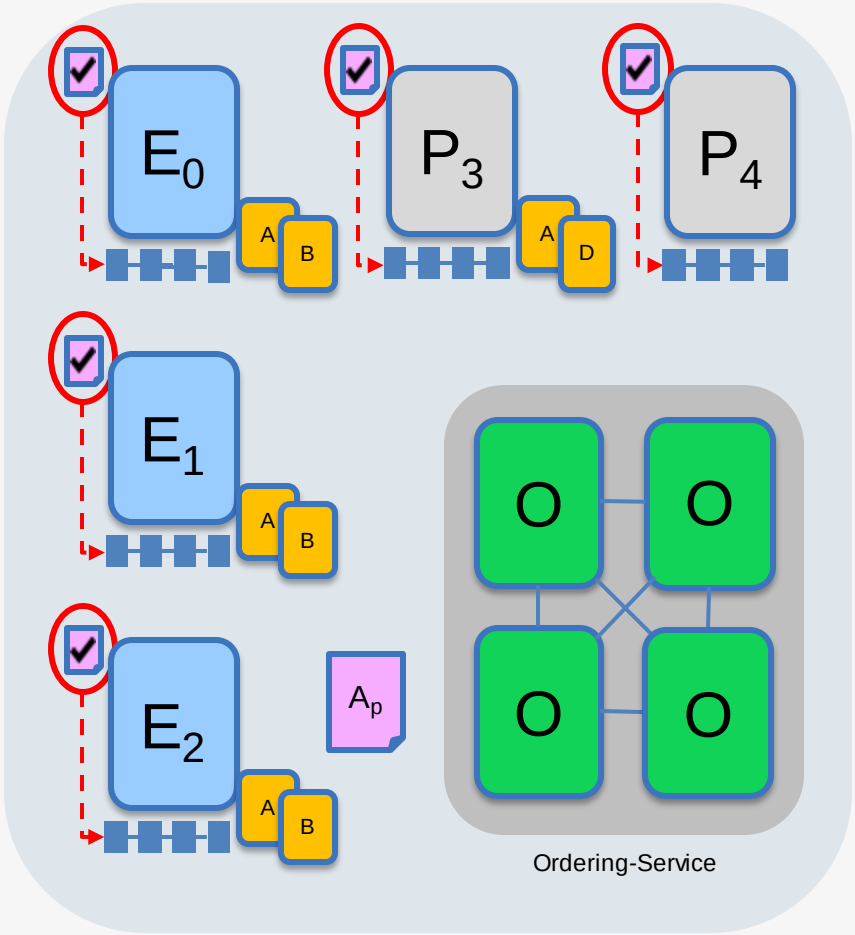
- SOLO (single node, development)
- Kafka (blocks map to topics)
- SBFT (tolerates faulty peers, future)

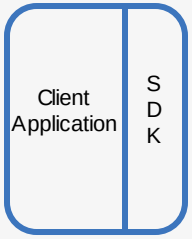
**Key:**




HYPERLEDGER
BLOCKCHAIN TECHNOLOGIES FOR BUSINESS

# Sample transaction: Step 6/7 – Validate Transaction



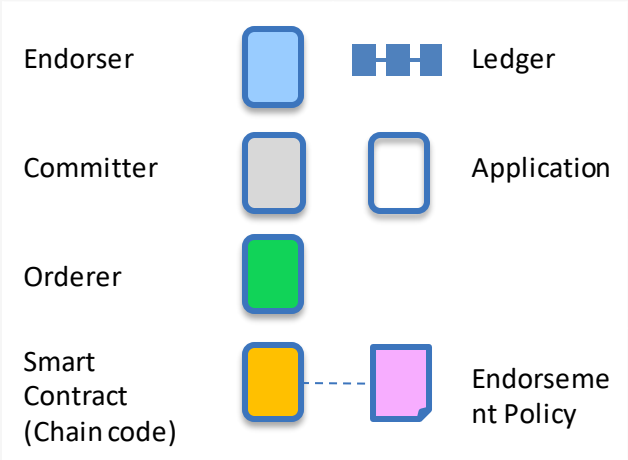

Hyperledger Fabric

**Committing peers validate transactions**

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

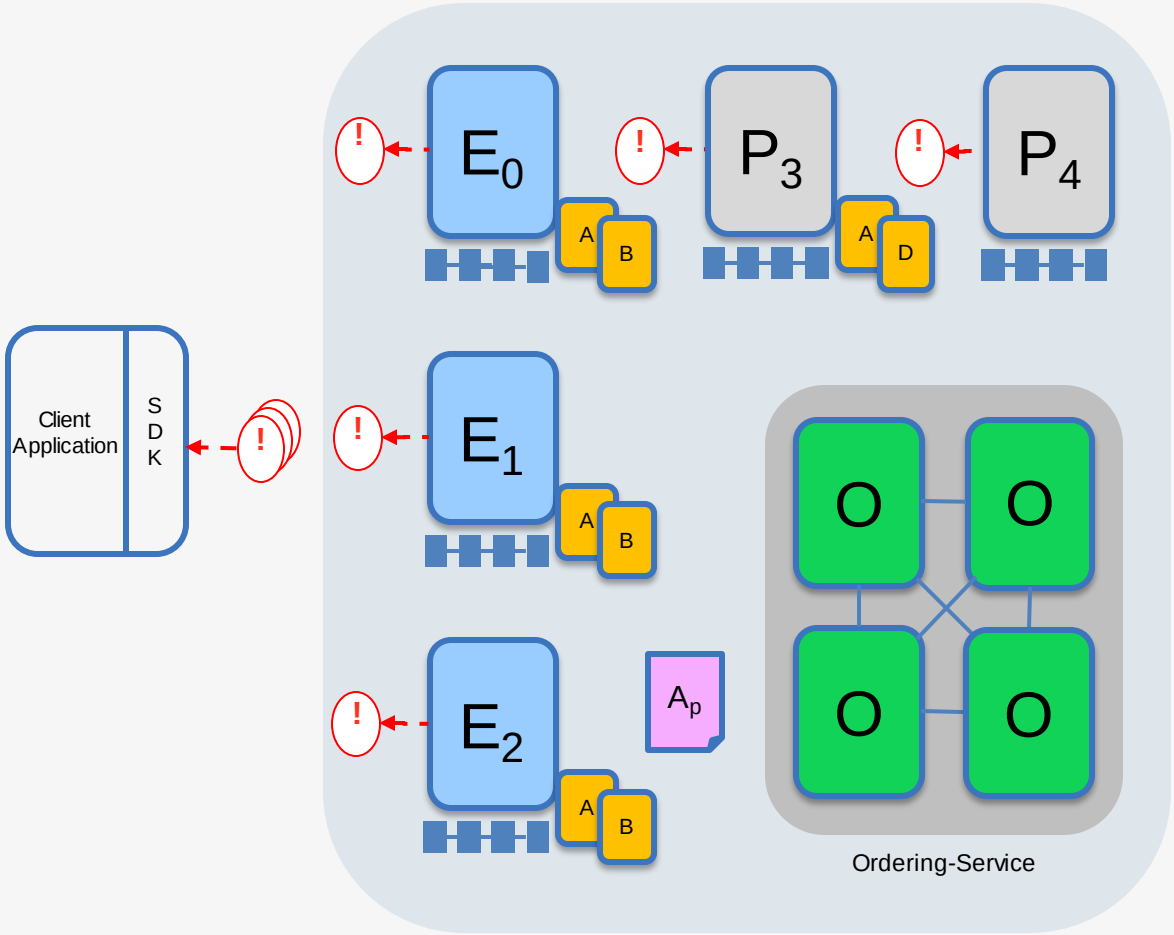
Transactions are written to the ledger and update caching DBs with validated transactions

**Key:**




HYPERLEDGER
BLOCKCHAIN TECHNOLOGIES FOR BUSINESS

# Sample transaction: Step 7/7 – Notify Transaction



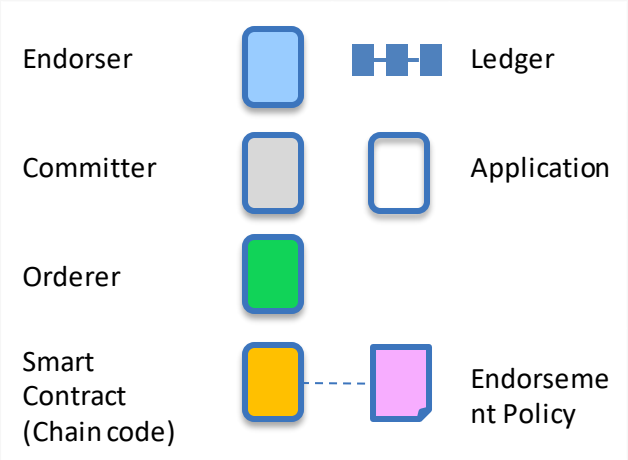

Hyperledger Fabric

**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:**




HYPERLEDGER
BLOCKCHAIN TECHNOLOGIES FOR BUSINESS