

TENEMOS
MUCHO
QUE HACER
JUNTOS

1- Introducción a Hyperledger Fabric: Conceptos Básicos



HYPERLEDGER

Tecnología Blockchain diseñada para el ámbito empresarial

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.

Miembros Hyperledger

Premier Members



General Members



General Members



Associate Members



Objetivos Hyperledger

Diferentes aproximaciones open-source de sistemas basados en tecnologías Blockchain para el ámbito empresarial



Create enterprise grade, open source, distributed ledger frameworks & code bases

to support business transactions



Provide neutral, open, & community-driven infrastructures

supported by technical and business governance



Build technical communities

to develop blockchain and shared ledger POCs, use cases, field trials and deployments



Educate the public

about the market opportunity for blockchain technology



Promote our community of communities

taking a toolkit approach with many platforms and frameworks

Proyectos Hyperledger

Infrastructure

Technical, Legal, Marketing, Organizational

Ecosystems that accelerate open development and commercial adoption



HYPERLEDGER



OPEN CONTAINER INITIATIVE

Frameworks

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



HYPERLEDGER SAWTOOTH



HYPERLEDGER IROHA



HYPERLEDGER FABRIC



HYPERLEDGER BURROW



HYPERLEDGER INDY

Tools

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



HYPERLEDGER CALIPER



HYPERLEDGER CELLO



HYPERLEDGER COMPOSER



HYPERLEDGER EXPLORER



HYPERLEDGER QUILT

Frameworks Blockchain Hyperledger

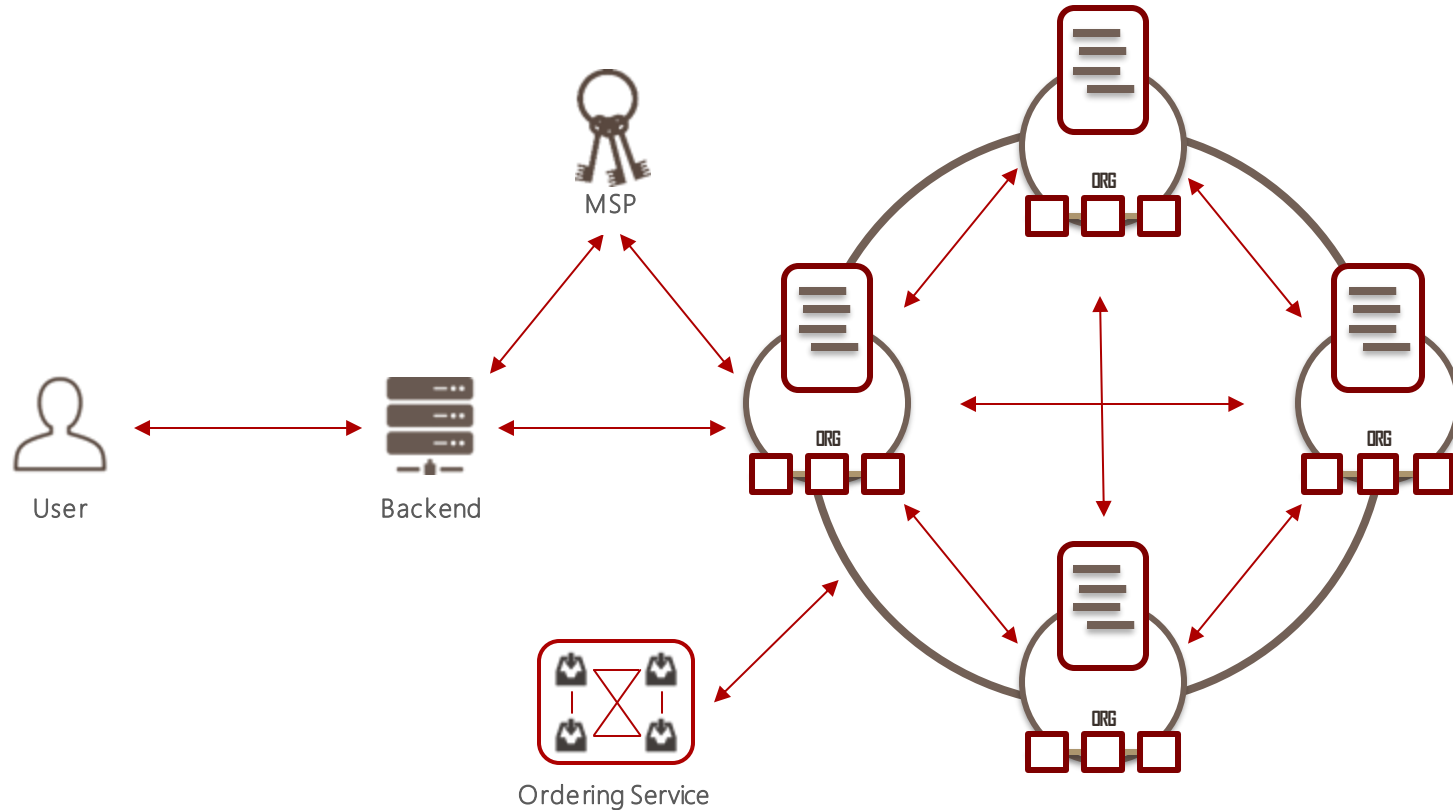
- **Hyperledger Fabric**
 - Base para el desarrollo de aplicaciones o **soluciones Blockchain permissionadas con arquitectura modular**.
 - Permite agregar como **componentes plug-and-play** funcionalidades tales como el consenso o la membresía.
 - Es el proyecto Hyperledger más activo.
 - Impulsada por IBM.
- **Hyperledger Sawtooth**
 - Plataforma modular para la construcción, desarrollo y ejecución de **ledgers distribuidos tanto permissionadas como no permissionadas**.
 - Utiliza un algoritmo de consenso PoET (Proof of Elapsed Time), que persigue el **soporte de grandes redes con el mínimo consumo de recursos**.
 - Impulsada por Intel.
- **Hyperledger Indy**
 - Herramientas, librerías y componentes reusables para la **provisión de identidades digitales originadas en redes Blockchain** (u otro tipo de redes de ledgers distribuidos), de modo que puedan ser interoperables a través de dominios administrativos, aplicaciones, etc.
 - Impulsada por la Sovrin Foundation.
- **Hyperledger Burrow**
 - **Intérprete permissionado de Smart Contracts construido sobre la especificación de la Ethereum Virtual Machine (EVM)**.
 - Está desarrollado en Solidity.
- **Hyperledger Iroha**
 - Framework Blockchain diseñado para ser sencillo y fácil de incorporar en proyectos de infraestructura que requieren tecnología de ledgers distribuidos.
 - **Optimizado para dispositivos móviles**.
 - Desarrollado en C++.



HYPERLEDGER FABRIC

- **Blockchain privada y permissionada de ámbito empresarial**
- **Arquitectura modular**
- Soporta **canales** para compartir información confidencial entre organizaciones.
- El **Ordering Service** entrega adecuadamente las transacciones a los nodos de la red.
- Las **endorsement policies** definen el proceso de aprobación de las transacciones.
- El **world state** soporta un abanico amplio de consultas (cuando se utiliza CouchDB).
- Abstracción de los procesos de autenticación y firma a través de un **Membership Service Provider (MSP)** plug-and-play.

Hyperledger Fabric: Componentes





Membership Service Provider (MSP)

- Provee credenciales y certificados de usuarios y nodos
- Abstrae de los procesos de autenticación y firma



Registro (ledger)

- Inmutabilidad
- Eficiencia en el coste (con respecto a blockchains públicas clásicas)
- Un ledger por canal
- Distribuida para las organizaciones del canal



Ordering Service

- Consenso intercambiable (plug-and-play)
- Ordenación de las transacciones en bloques
- Ordering System Channel:
 - Definición de consorcios
 - Constitución de nuevos canales



Red de nodos (Organizaciones)

- 2 tipos de nodos:
 - Endorsers
 - Ejecutan y aprueban las transacciones
 - Committers (todos)
 - Verifican los endorsements
 - Validan los resultados de las transacciones
 - Almacenan el ledger
- Características
 - Garantía de origen
 - Acceso permissionado
 - Nodos distribuidos
 - Descentralización



Smart Contracts (Chaincode)

- Reglas de negocio
- Instalado en los peers asociados al canal
- Instanciados sobre el canal
- Lenguaje Go (entre otros)

Tipología de nodos



Orderers

- Encargados de la creación de canales y de la suscripción a los mismos → **Ordering System Channel**
- **Ordenan las transacciones en bloques**
- Orquestan la comunicación
- **Consenso**



Endorsers

- **Ejecutan el chaincode**
- **Aprueban las transacciones**
- Todos los endorsers son también committers (no necesariamente al revés)
- **Endorsement policies:** Indican las organizaciones cuyos endorsers han de aprobar las transacciones (AND, OR...)



Committers

- Verifican los endorsements
- Validan los resultados de las transacciones
- **Almacenan el ledger** (Blockchain + World State)

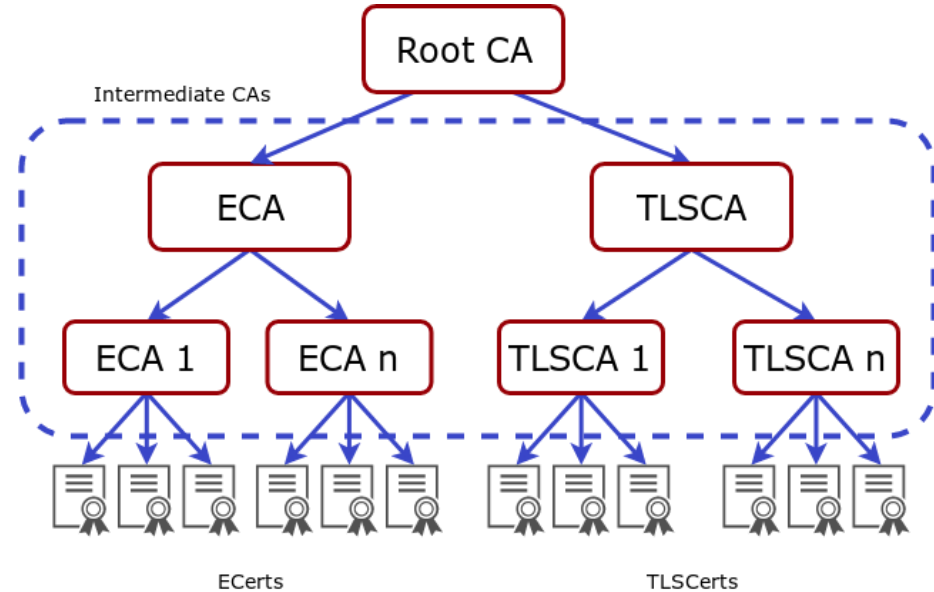
Membership Service Provider I

- *Pluggable interface* que soporta distintas arquitecturas de credenciales de autenticación y control de acceso.
- El MSP abstrae:
 - Formato de identidad concreto
 - Validación de las credenciales de usuario
 - Revocación de las credenciales de usuario
 - Generación y verificación de firmas
- La implementación por defecto del MSP en Fabric está basada en el modelo jerárquico PKI (Public Key Infrastructure):
 - Identidad → Certificado X.509

Membership Service Provider II

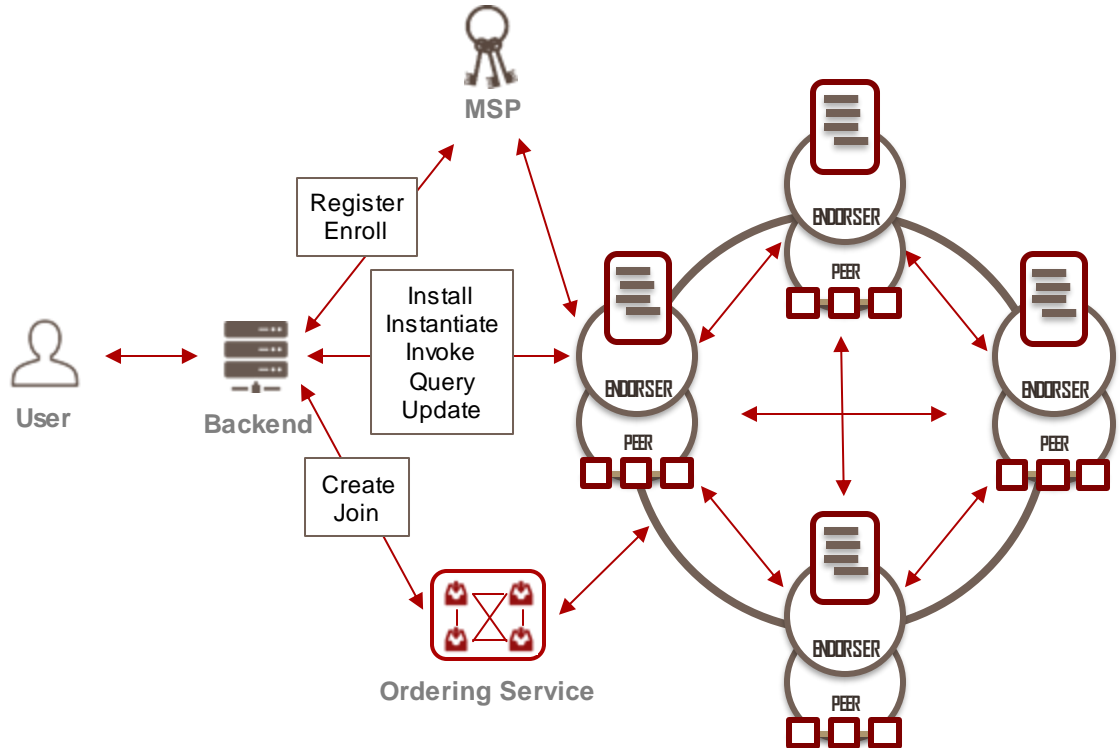
- **ECA** (Enrollment CA) → Certificados de usuario
- **TLSCA** (TLS CA) → Securitización de los canales de comunicación

- **Register**
 - Registro de una identidad
 - hf.Registrar.Roles: peer, app, user
 - Afiliación
 - Atributos
 - Enrollment password
- **Enroll**
 - Generación de ECert, clave privada y cadena de certificación de una identidad previamente registrada



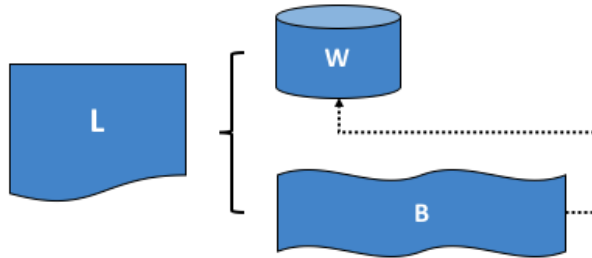
Smart Contracts






- Se les conoce como **Chaincode**
- Lenguajes:
 - **Golang**
 - Java
 - Python (incubation)
 - NodeJS (incubation)
- **SDK:**
 - Operaciones permitidas:
 - Register
 - Enroll
 - Create channel
 - Join channel
 - Install
 - Instantiate
 - Upgrade
 - Invoke
 - Query



Ledger

- Registro de transacciones.
- Un registro por canal:
 - Compartido por las organizaciones suscritas al canal.
 - Almacenado en committer nodes.
- 2 componentes:
 - **World State**: Guarda los valores actuales del estado del ledger.
 - **Blockchain**: Log de transacciones que recoge cada cambio que determina el world state actual.


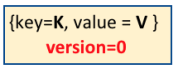
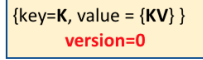


	Ledger
	World State
	Blockchain
	L comprises B and W
	B determines W

Ledger: World State

- Guarda los valores actuales del estado del ledger.
- BBDD:
 - LevelDB → Clave-valor
 - CouchDB → Documentos JSON → Soporta consultas más complejas.

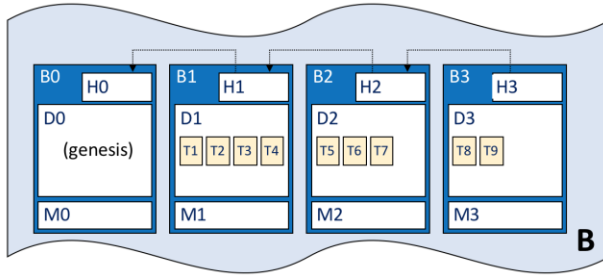


	Ledger world state
	A ledger state with key=K . It contains a set of facts expressed as a simple value, V . The state is at version 0.
	A ledger state with key=K . It contains a set of facts expressed as a set of key-value pairs {KV} . The state is at version 0.

Ledger: Blockchain

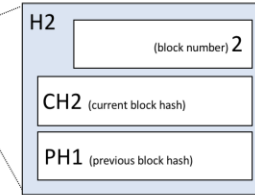
- Log con todas las transacciones efectuadas sobre el canal, estructuradas en bloques enlazados criptográficamente mediante hashes.
- Inmutable.
- Implementado como un fichero (no como una BBDD).
- Se puede determinar el World State a partir de las transacciones de la Blockchain.

Blockchain



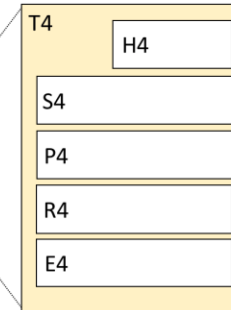
B	Blockchain
B1	Block
H3	Block header
D1	Block data
T5	Transaction
M3	Block metadata
H1 H2	H2 is chained to H1

Bloque



H2	Block header
2	Block number
CH2	Hash of current block transactions
PH1	Copy of hash from previous block
H2 V2	V2 is detailed view of H2

Transacción

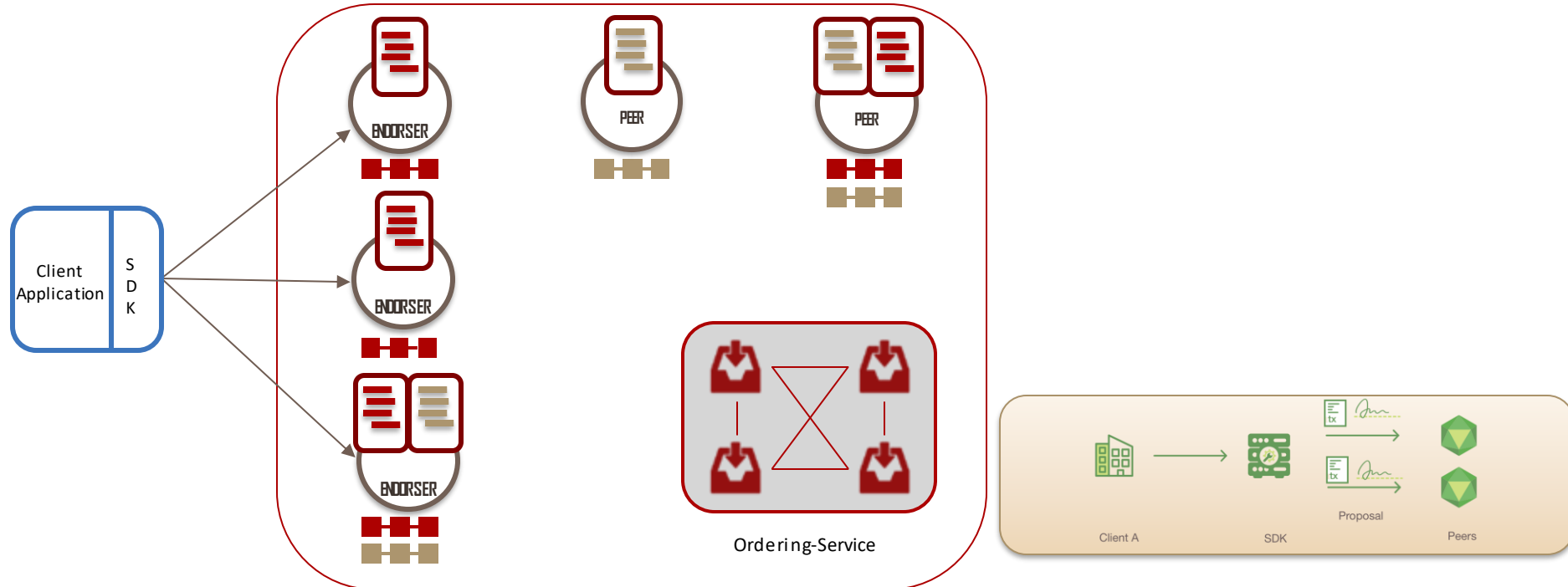


T4	Transaction
H4	Header
S4	Signature
P4	Proposal
R4	Response
E4	Endorsements
T4 V4	V4 is detailed view of T4

Flujo de transacción

1. Transaction Proposal

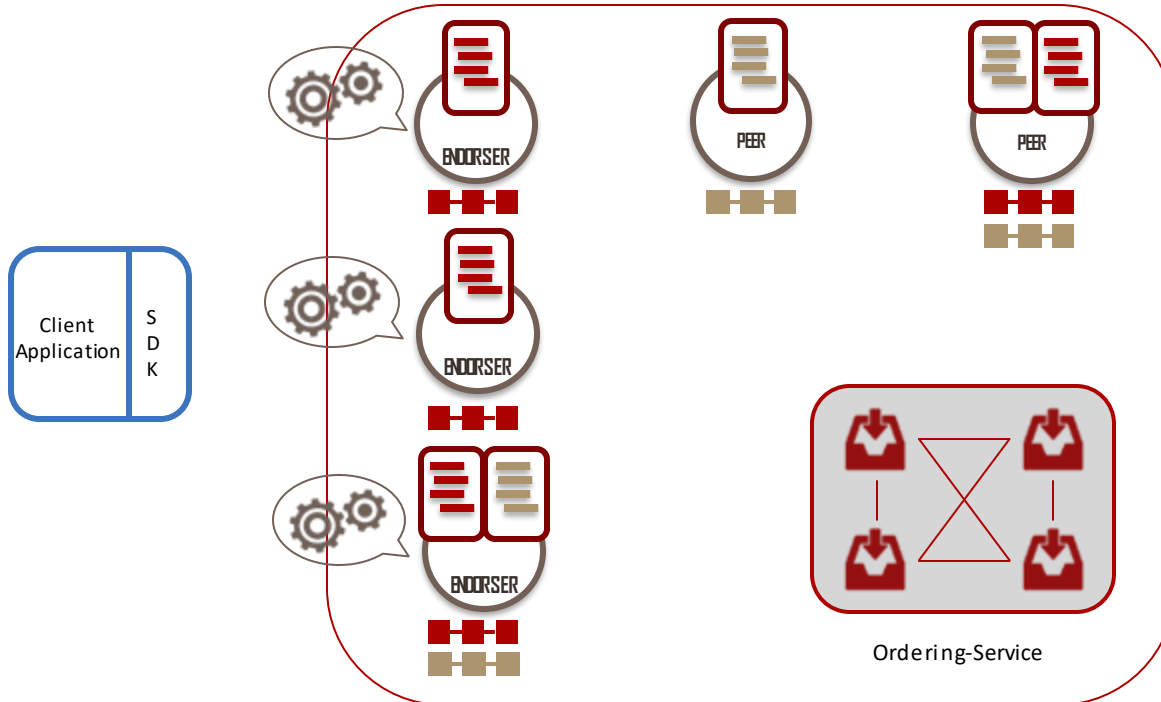
El cliente firma una **Transaction Proposal** y la envía a los Endorser Peers.



Flujo de transacción

2. Ejecución del chaincode

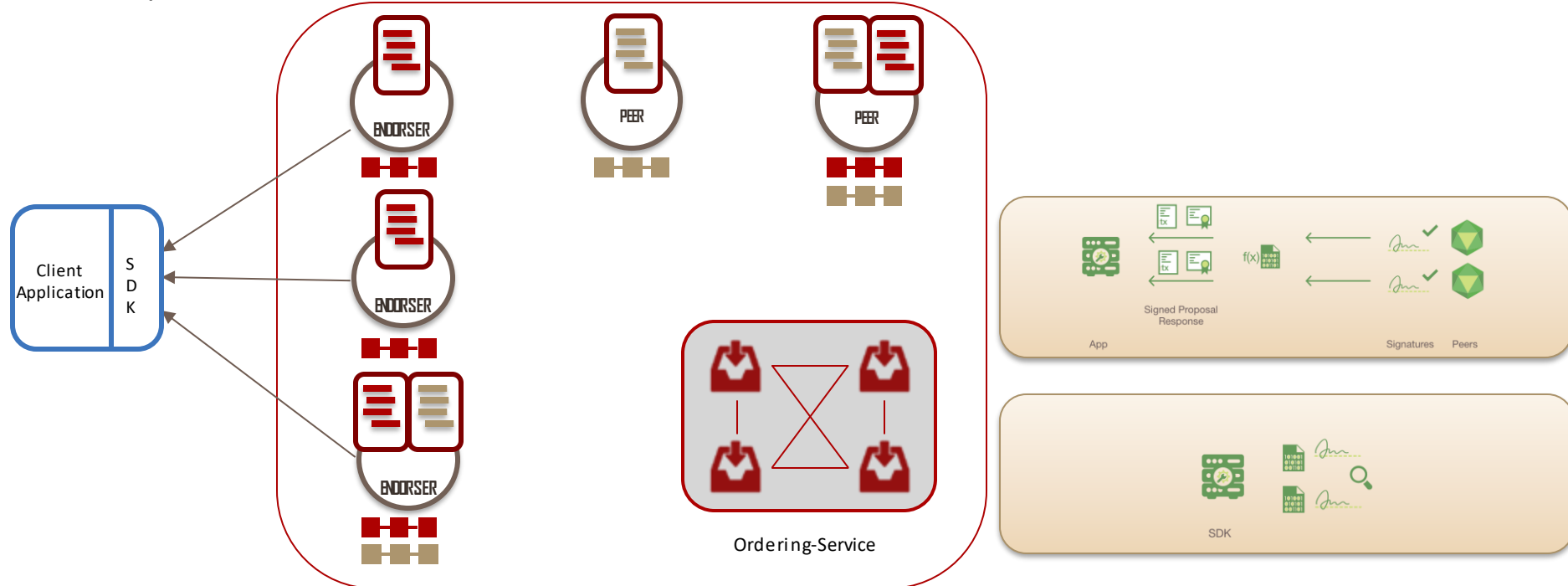
Los Endorser Peers verifican la firma y ejecutan la transacción.



Flujo de transacción

3. Proposal Response

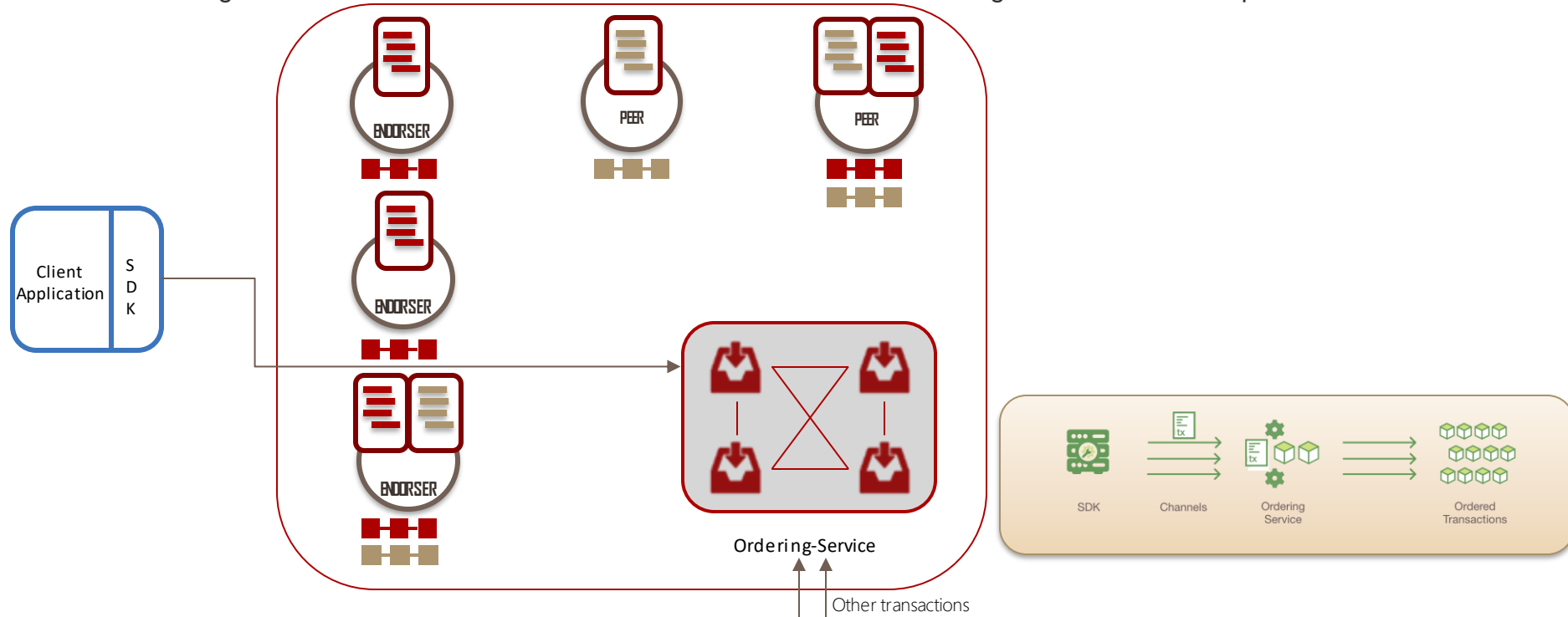
Los Endorser Peers firman las **Proposal Response** y las devuelven al cliente, que verifica las firmas requeridas para cumplir las Endorsement Policies.



Flujo de transacción

4. Ordering Service

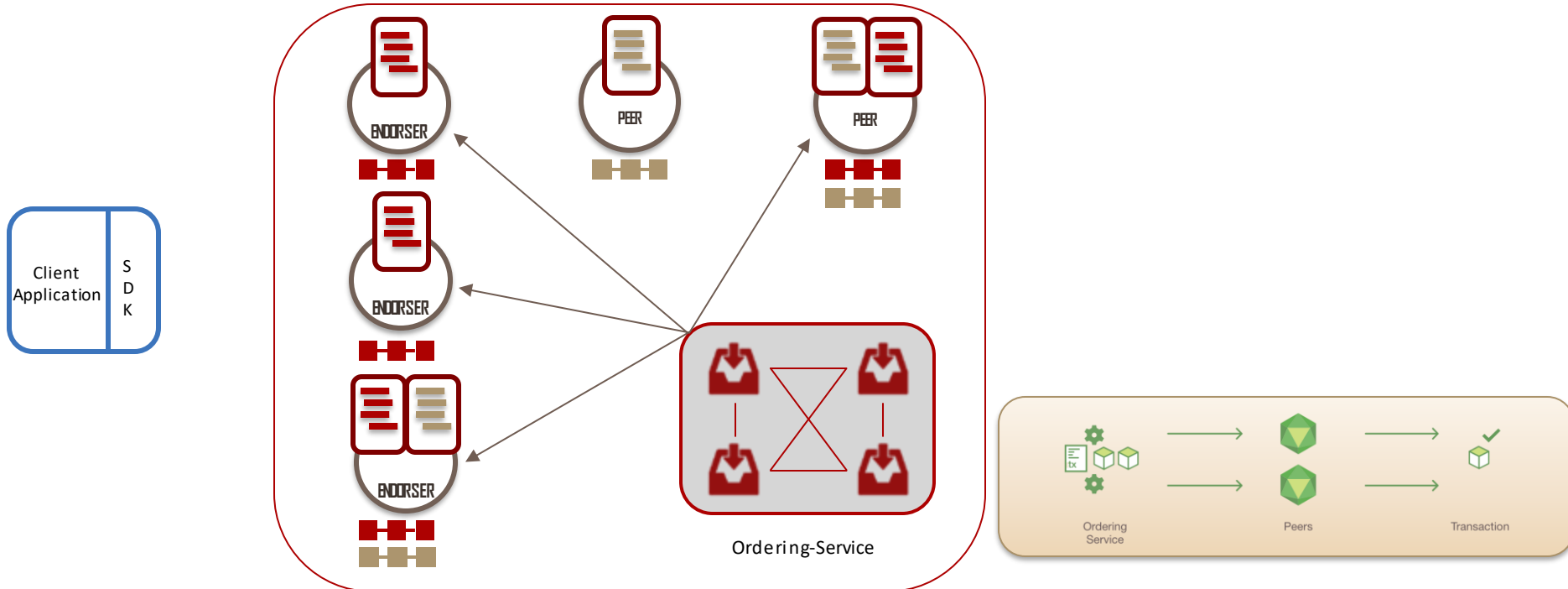
El cliente construye un único **Transaction Message** con todos los endorsements y hace un broadcast al Ordering Service. El Ordering Service ordena distintas transacciones del mismo canal cronológicamente en un bloque.



Flujo de transacción

5. Devolución del bloque

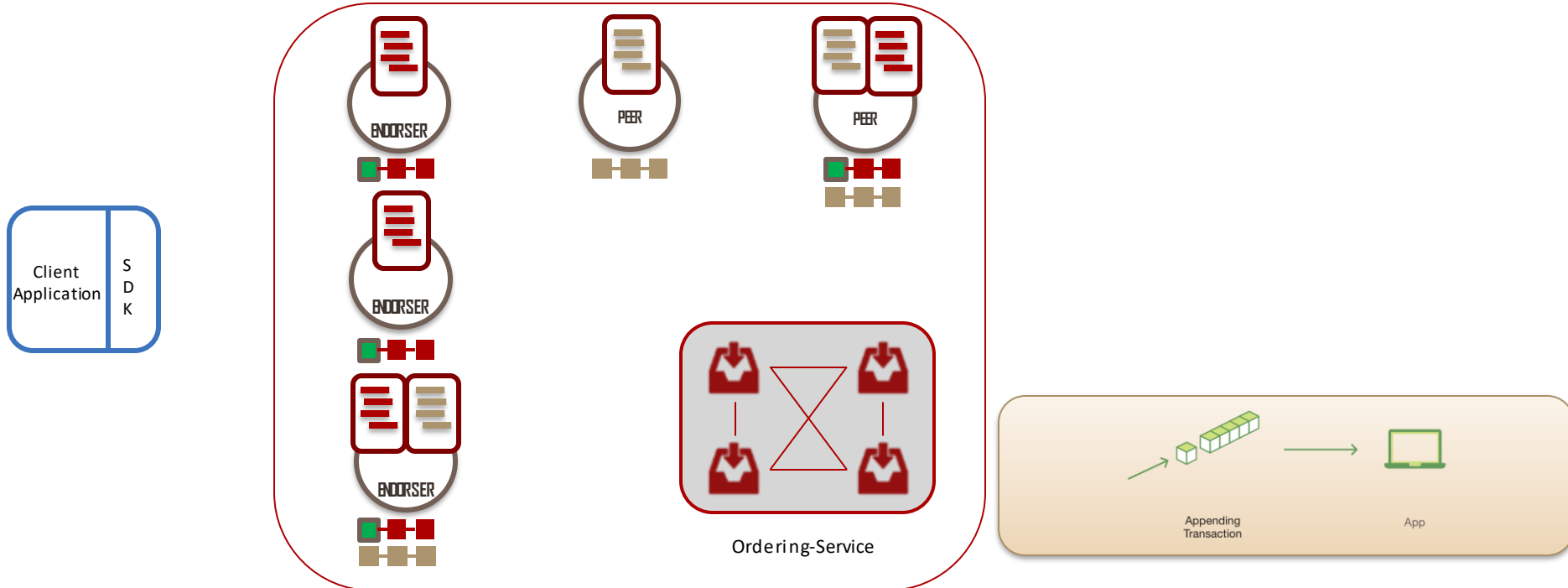
El Ordering Service envía el bloque a todos los committers de organizaciones suscritas al canal.



Flujo de transacción

6. Inserción en el ledger

Los committers verifican las endorsement policies, agregan el bloque a la Blockchain y actualizan el World State en consecuencia.

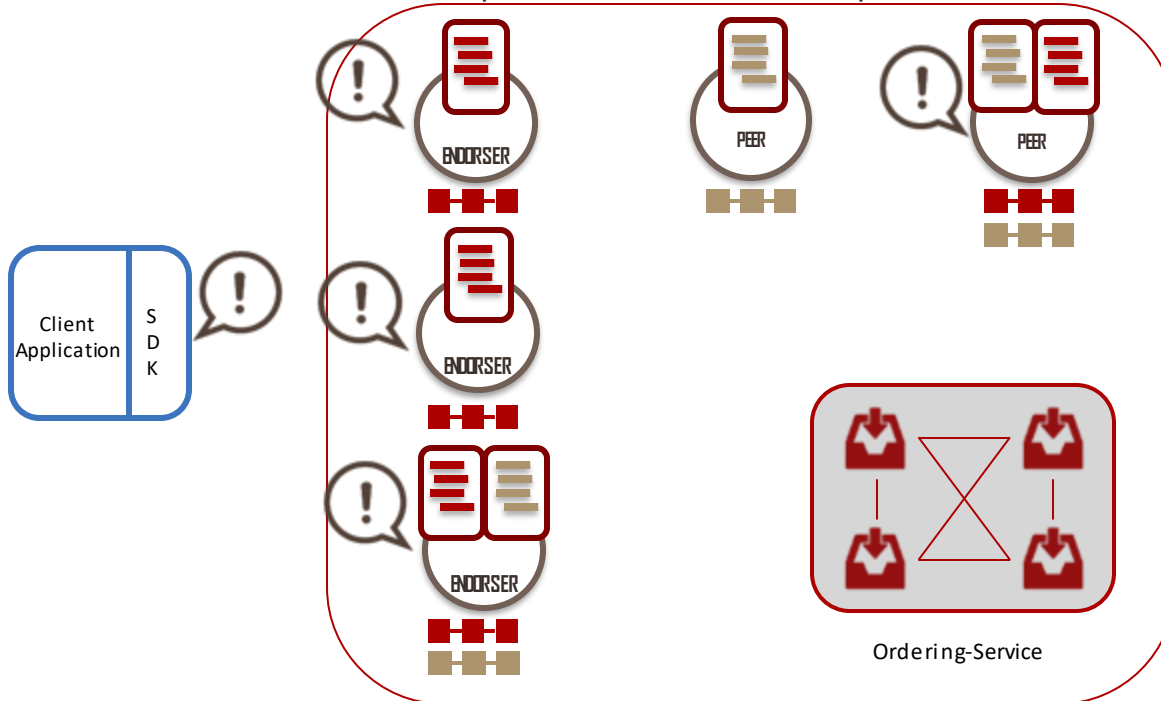


Flujo de transacción

7. Fin.

El ledger del canal queda actualizado.

El SDK notifica al cliente que su transacción ha sido procesada.



Consenso

- Entre orderers del Ordering Service
- Pluggable
- Actualmente 3 posibilidades:
 - Solo
 - Testing
 - Un único orderer
 - Sin consenso
 - Kafka-based
 - Kafka: Sistema de mensajería publish-suscribe distribuido
 - Zookeeper: Servicio de coordinación de aplicaciones distribuidas de alto rendimiento
 - ZAB (Zookeeper Atomic Broadcast): PoC (Proof of Correctness)
 - PBFT
 - Practical Byzantine Fault Tolerance
 - En desarrollo

Hyperledger Fabric: Ventajas

- **Gobernanza:** Linux Foundation
- Amplia **documentación** y **comunidad** de desarrollo
- Plataforma **modular** (identidad, consenso...)
- **Lenguajes de programación** estándar: Go, Java, NodeJS y Python
 - Variables nativas
 - Clases
- Concepto de **organización**
- **Canales privados**
- **Permissionado** detallado
- **BBDD modular:**
 - LevelDB: Key-value
 - CouchDB: JSON documents
- **Interledger** (a través de Hyperledger Quilt)

Hyperledger Fabric: Desventajas

- Infraestructura compleja
 - Despliegue
 - Escalabilidad
- No incorpora una divisa nativa
- Privacidad de alto nivel (no llega a nivel de operación):
 - Canales privados
 - Lógica del propio chaincode
 - TLS → Comunicaciones punto a punto
 - Fabric 1.2:
 - Private Data Collections
 - Access Control Lists