HYPERLEDGER

BLOCKCHAIN TECHNOLOGIES FOR BUSINESS

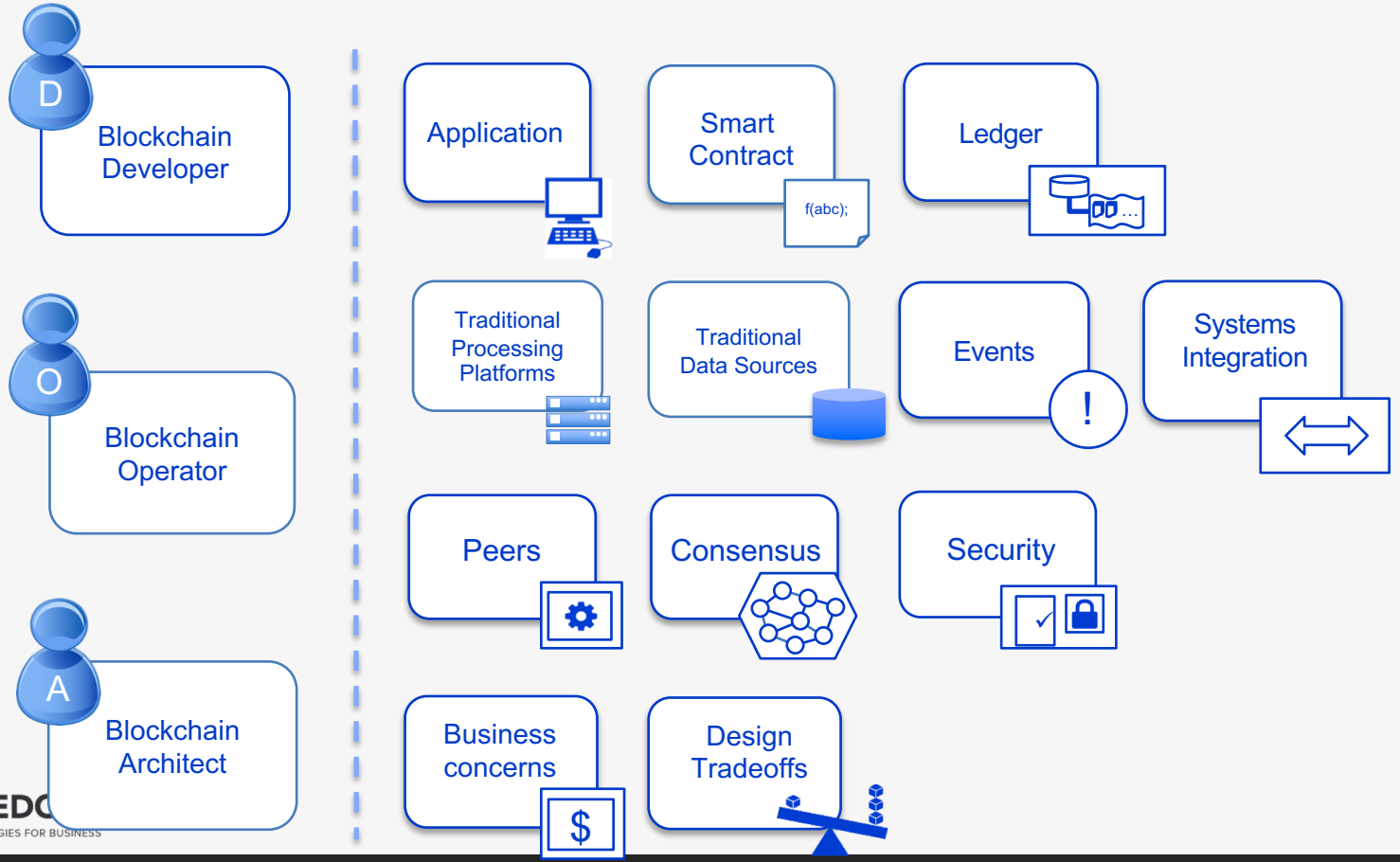
Hyperledger Fabric

Architecture

Technical Deep Dive

Architectural Overview

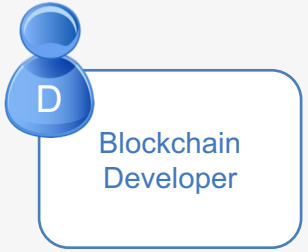
블록체인에서의 역할



HYPERLEDG
BLOCKCHAIN TECHNOLOGIES FOR BUSINESS

블록체인 개발자의 역할

블록체인 개발자의 주된 역할은 어플리케이션, 스마트 컨트랙트의 개발



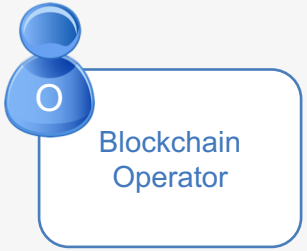
그리고 블록체인과 원장의 상호관계나 타 시스템과의 연동과 관련한 개발

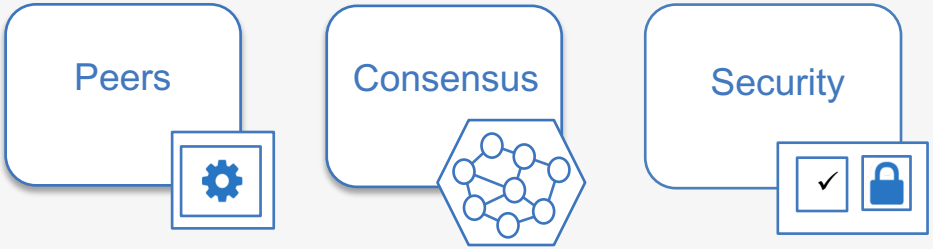
블록체인의 동작 구조에 대해서 반드시 알 필요는 없음.

X Peers Consensus Security

블록체인 운영자의 역할

블록체인 운영자는 블록체인의 동작 구조에 대해서 상세히 파악하고 있어야 함:



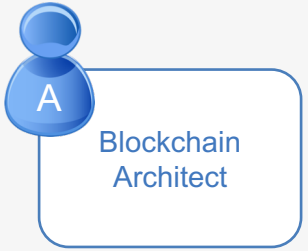


반드시 블록체인 개발을 할 수 있어야 하는 건 아님:

X Application code Smart contract code

HYPERLEDGER
BLOCKCHAIN TECHNOLOGIES FOR BUSINESS

블록체인 아키텍트의 역할

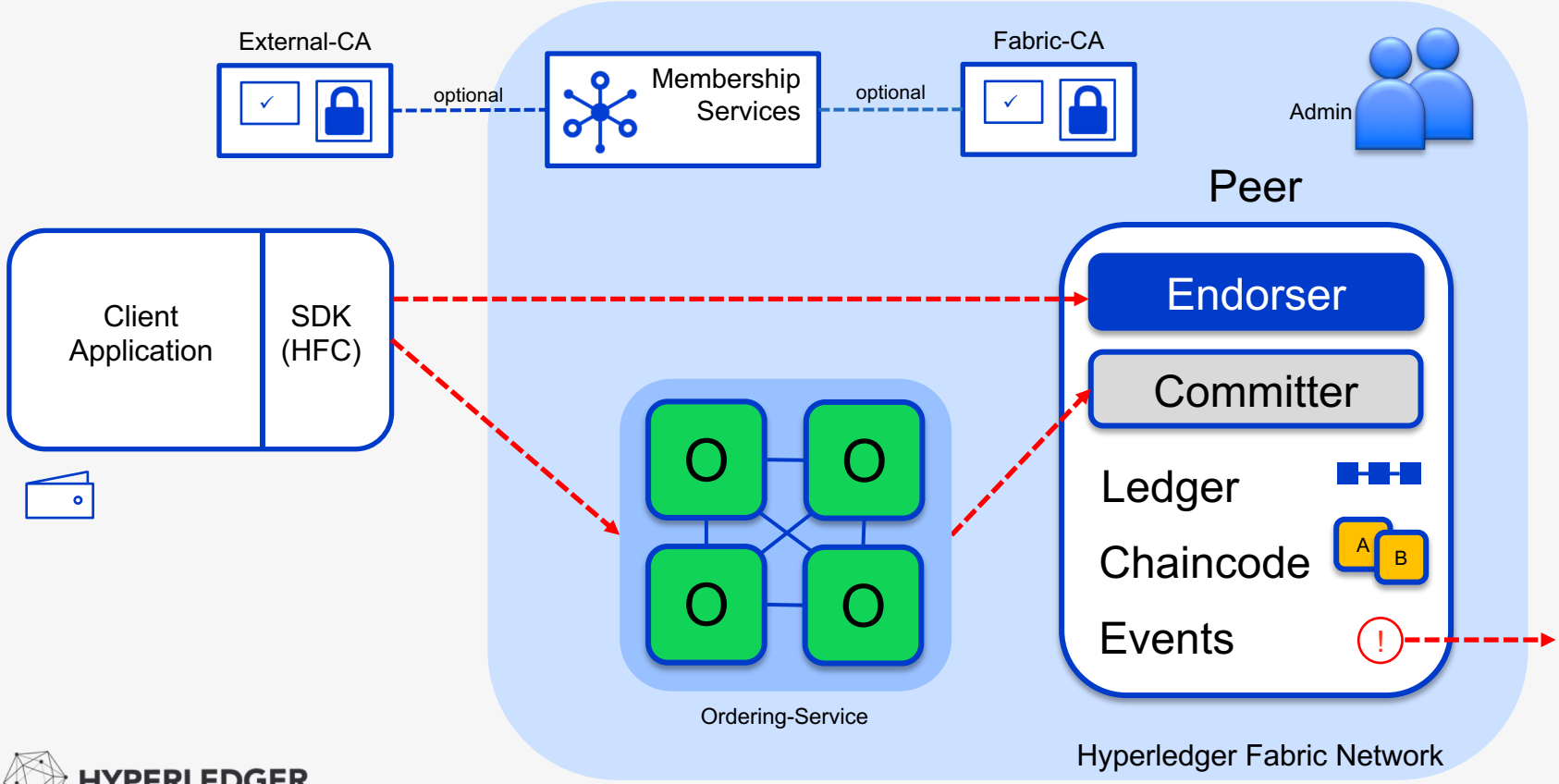


성공적인 블록체인 도입을 위해서 블록체인 아키텍트는 개발과 운영 관점에서 많은 지식을 보유하고 있어야 함:

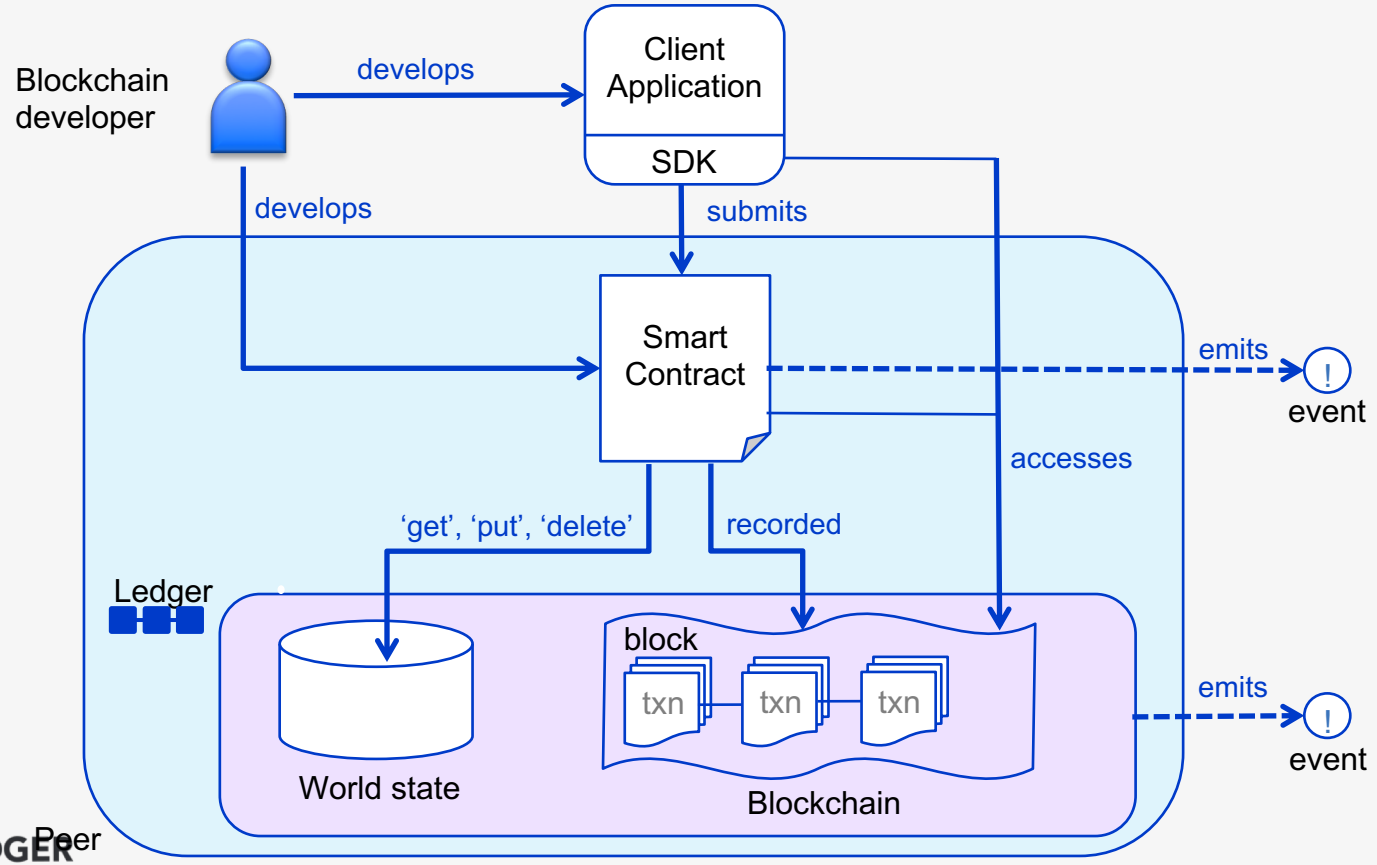
- Applications
- Smart contracts
- Events and Integration
- Peers
- Consensus
- Security

그리고 추가적으로 다음과 같은 영역을 항상 고려해야 함.

Hyperledger Fabric V1 Architecture

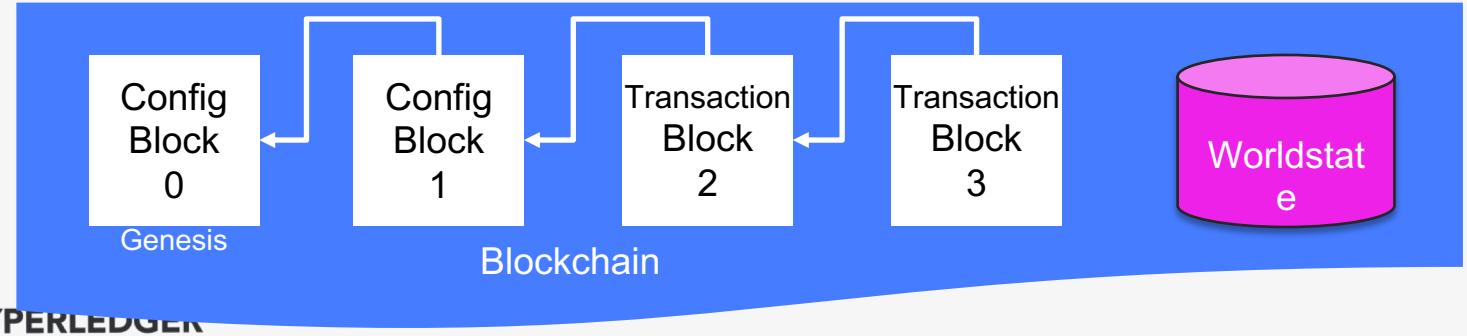


애플리케이션과 원장의 호출 관계

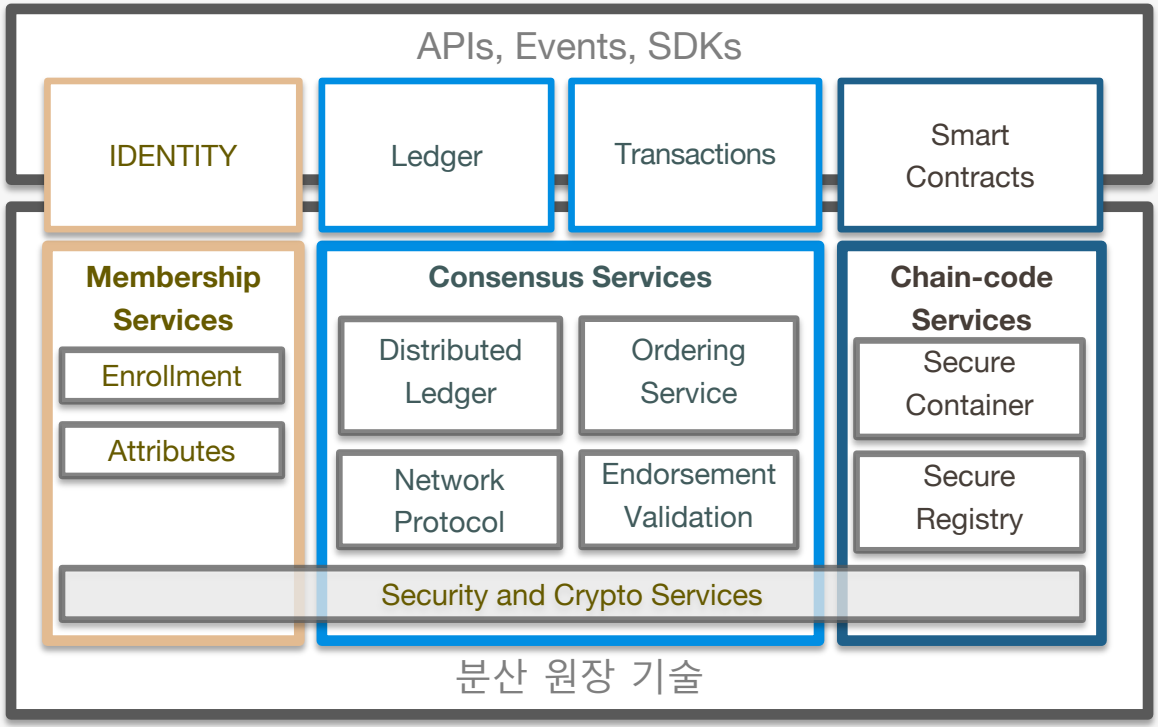


Fabric Ledger

- 블록체인과 worldstate를 포함하고 있는 패브릭 원장은 각 피어에 의해 유지 됨
- 피어가 조인하는 각 채널에 대해 별도의 원장이 생성/관리 됨
- 트랜잭션의 Read/Write sets 는 블록 체인에 기록 됨
- 채널의 구성 정보도 블록 체인에 기록 됨
- worldstate는 LevelDB (기본값) 또는 CouchDB 선택 가능함
  - LevelDB는 key/value 스토어
  - CouchDB는 도큐먼트 스토어
- 스마트 컨트랙트에서 worldstate에 데이터를 입력 함



Fabric 아키텍처



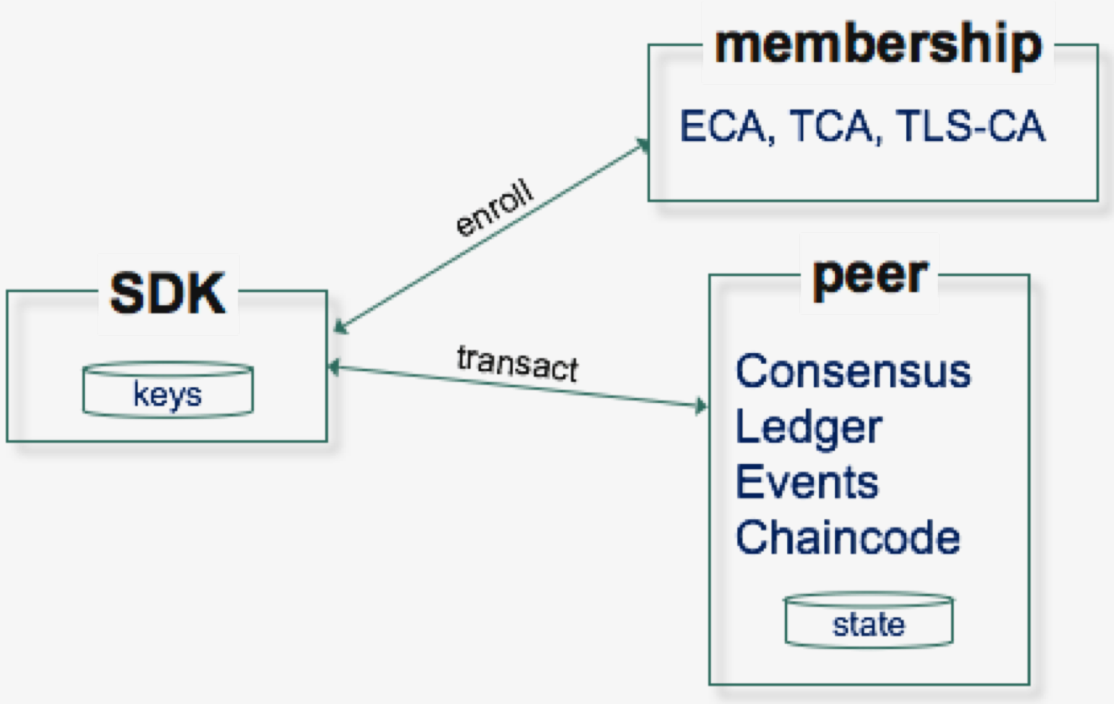
IDENTITY
플러그인 방식, 멤버쉽, 프라이버시 제공 및 트랜잭션 감사

LEDGER | TRANSACTIONS
이해 당사자간의 합의에 의한 분산 트랜잭션 원장 저장

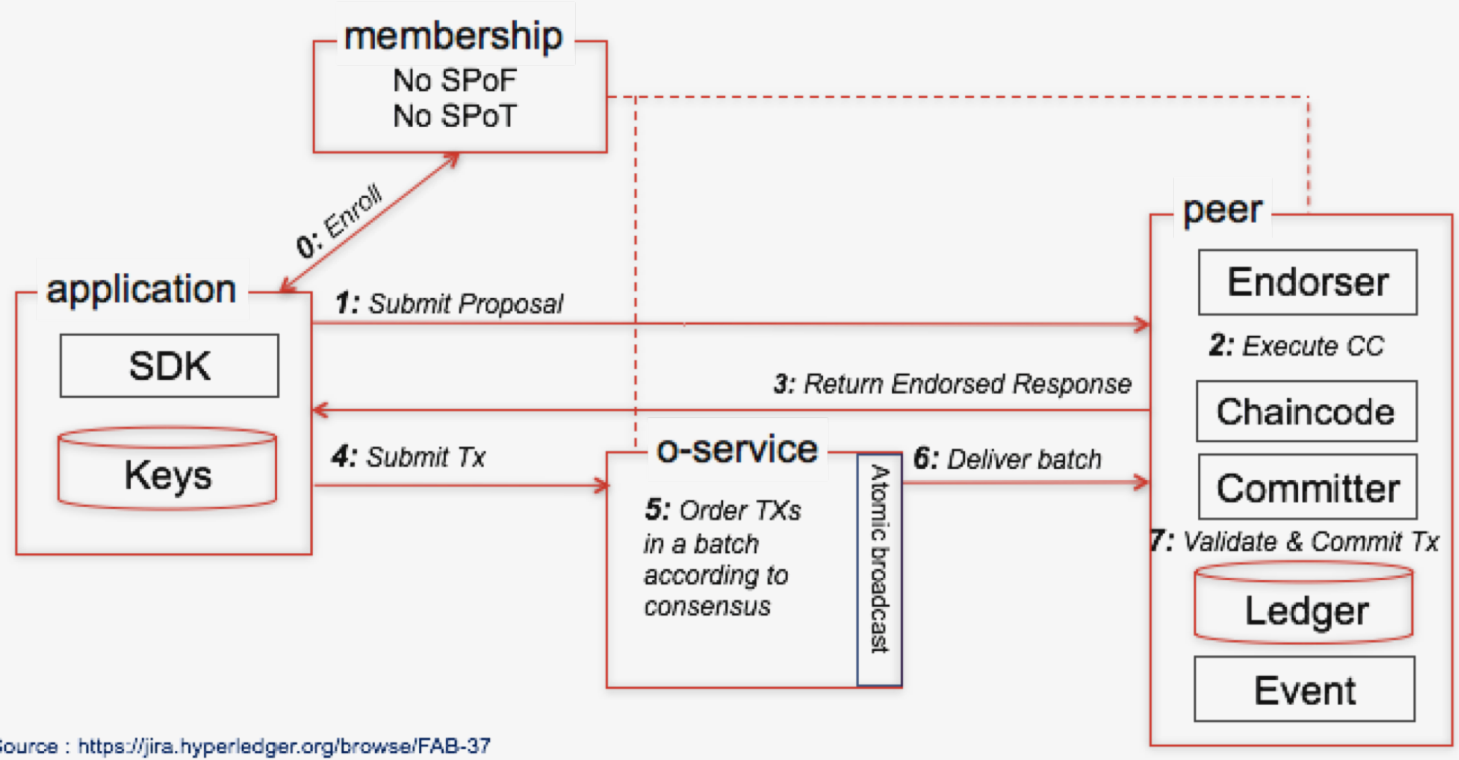
SMART-CONTRACT
"프로그래밍 가능한 원장", 블록체인에서 비즈니스 로직을 실행할 수 있는 환경 제공

APIs, Events, SDKs
다양한 개발 언어가 지원되는 SDK를 이용항 DLT 앱 개발

v0.6 Architecture



v1.0 Architecture



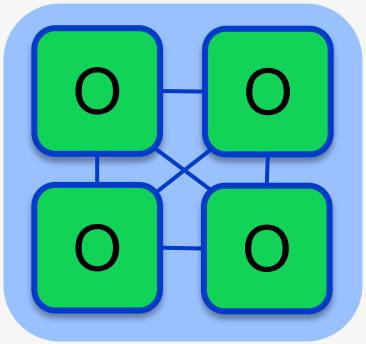
Source : https://jira.hyperledger.org/browse/FAB-37

Technical Deep Dive

Component

Ordering Service

Odering service는 트랜잭션들을 블록으로 패키징하여 피어에게 전달하는 역할을 담당함. 채널을 통해 Ordering service와 통신 함.



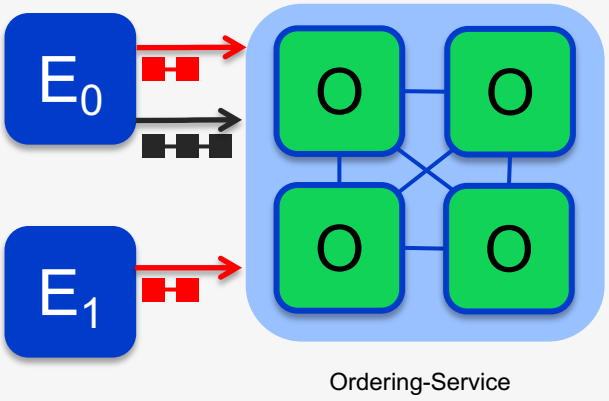
Ordering-Service

Ordering service 설정 옵션:

- SOLO
  - 개발을 위한 싱글 노드
- Kafka : Crash fault tolerant consensus
  - 최소 3 노드
  - 홀수 노드 권장 함

채널

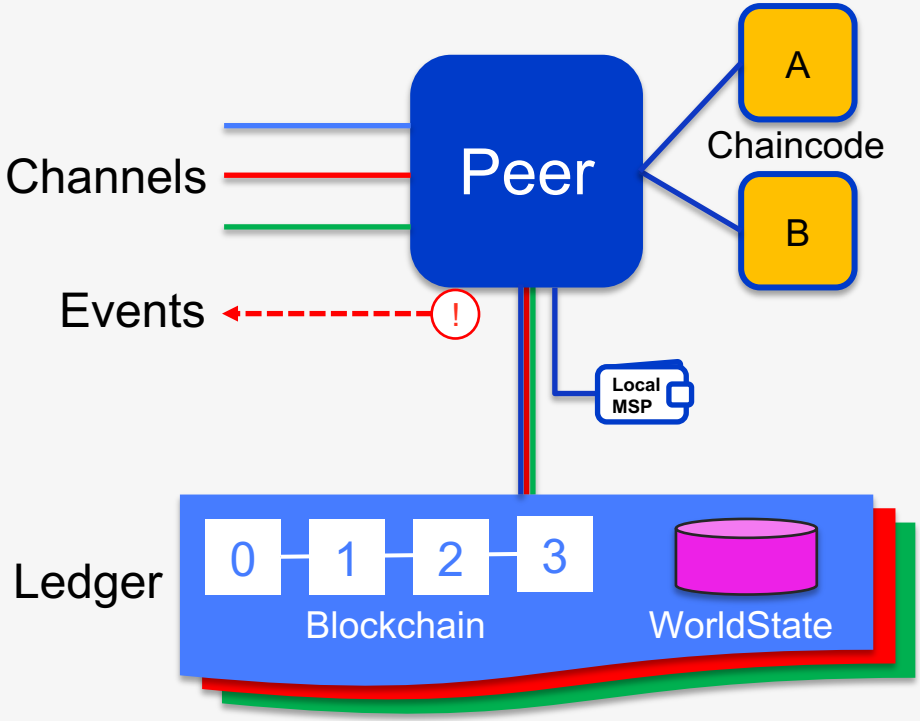
채널은 서로 다른 원장간 프라이버시를 제공 함



- 원장은 채널의 범위로 한정 됨
  - 피어의 전체 네트워크에서 채널을 공유할 수 있음
  - 특정한 참여자 그룹 별로 채널 권한을 부여할 수 있음
- 체인코드는 피어에 설치되며 worldstate에 접속할 수 있음
- 체인코드 특정 피어에서 인스턴스화 됨
- 피어는 멀티 채널에 참여 할 수 있음
- 퍼포먼스와 확장성을 고려하여 동시에 실행 가능함

Fabric Peer

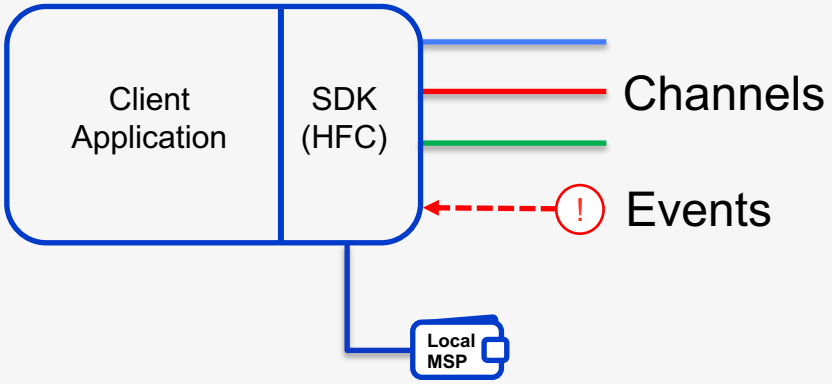
- 피어:
  - 한 개 이상의 채널에 연결됨
  - 각 채널에 대해서 한 개 이상의 원장을 관리
  - 체인코드는 도커 컨테이너로 분리되어 인스턴스화 됨
  - 체인코드는 채널을 통해 공유 됨
  - Local MSP (Membership Services Provider) 는 암호화 방식을 제공
  - 클라이언트 어플리케이션에 이벤트 발생



클라이언트 어플리케이션

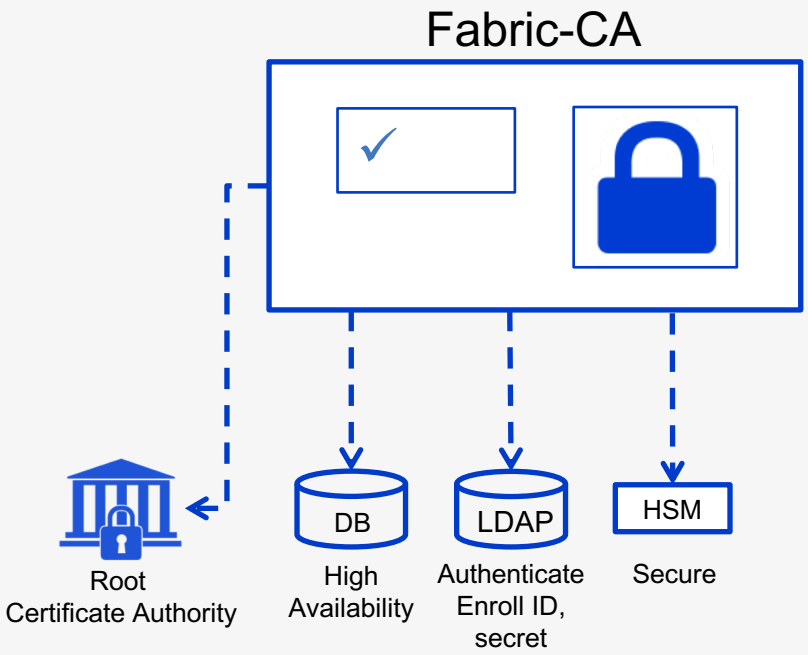
- 모든 클라이언트는 패브릭 SDK 를 사용하여:
  - 채널을 통해 하나 이상의 피어에 접속
  - 채널을 통해 하나 이상의 Orderer에 접속
  - 피어로 부터 이벤트 수신
  - Local MSP 기능 제공

  - 다양한 개발 언어 지원
    (Node.js, Go, Java, Python?)



Fabric-CA

- 패브릭 네트워크에서 Ecerts를 발행하는 Certificate Authority
- HA를 위한 클러스터링 지원
- 사용자 인증을 위한 LDAP 지원
- 보안을 위한 HSM 지원
- Intermediate CA를 위한 설정 가능



Technical Deep Dive

Network Consensus

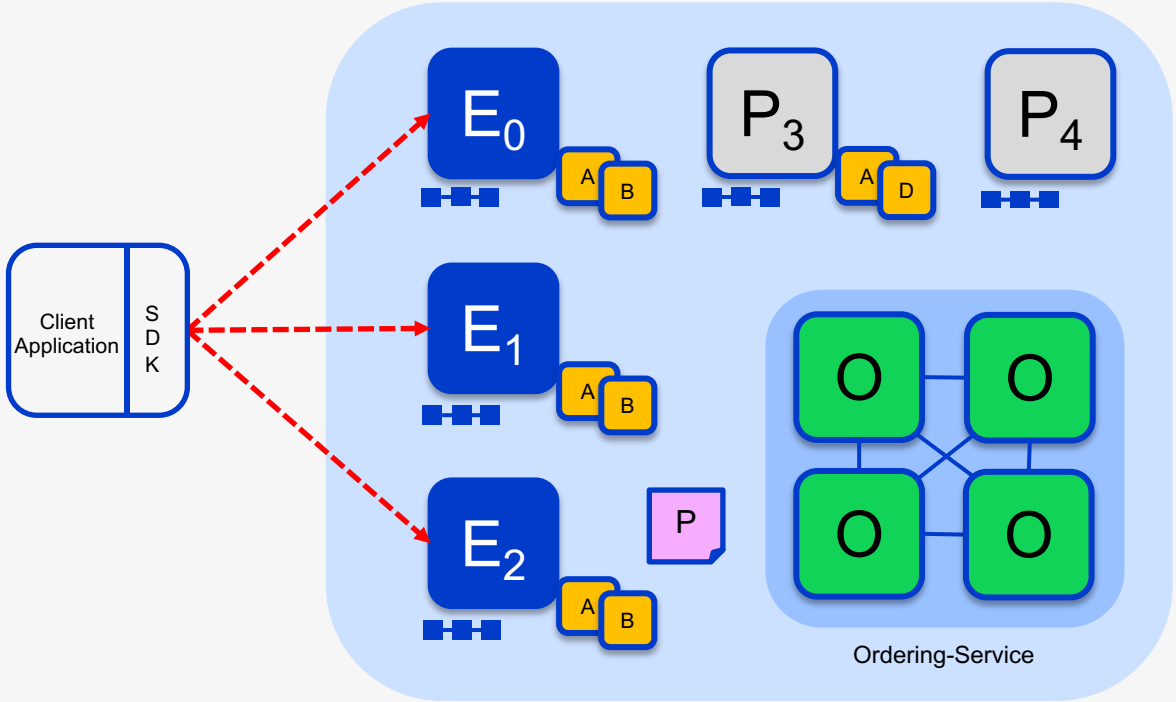
노드 와 역할

	**Peer:** 원장과 state 관리, 트랜잭션 커밋, 스마트 컨트랙트(체인코드) 실행
	**Endorsing Peer:** Endorse를 위한 특화된 피어로 트랜잭션 제안을 검증하여 수용 및 거부를 결정 함
	**Ordering Node:** 원장에 트랜잭션 블록을 포함시키기 위해 Comitting 피어와 Endorsing 피어들과 통신 함, 스마트 컨트랙트나 원장을 가지고 있지 않음

Hyperledger Fabric 컨센서스

Hyperledger Fabric에서는 다음의 단계를 거쳐 컨센서스를 이룸:

Sample transaction: Step 1/7 – Propose transaction



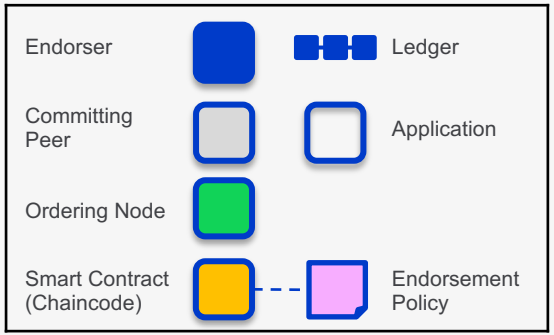
Hyperledger Fabric Network

Application proposes transaction

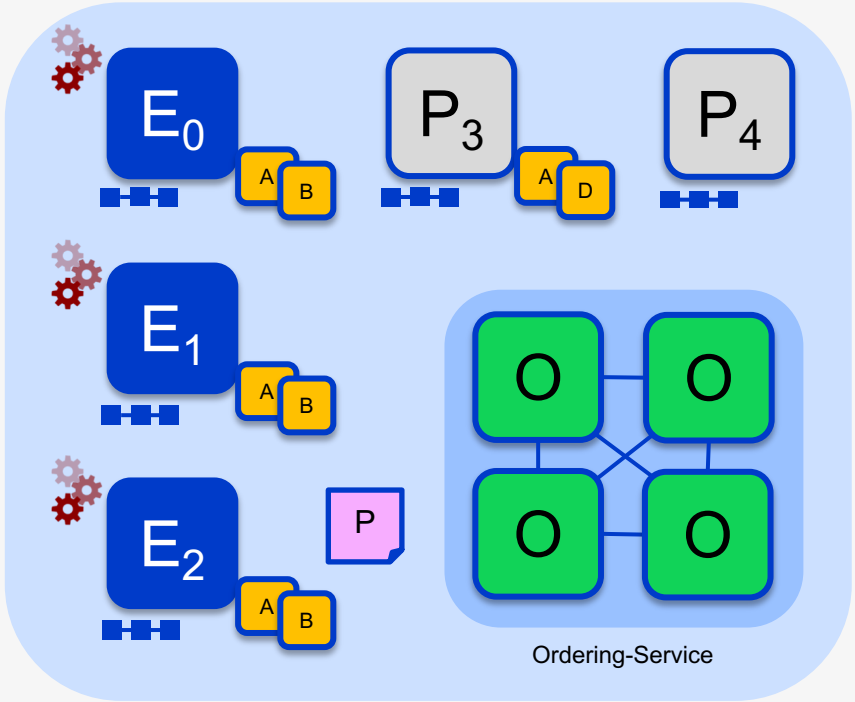
Endorsement policy:
- "E_0, E_1 and E_2 must sign"
- (P_3, P_4 are not part of the policy)

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

Key:



Sample transaction: Step 2/7 – Execute proposal



Hyperledger Fabric Network

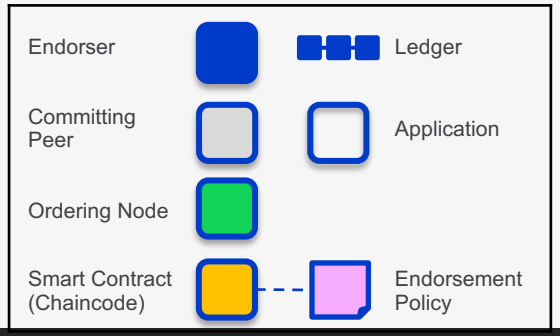
Endorsers Execute Proposals

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

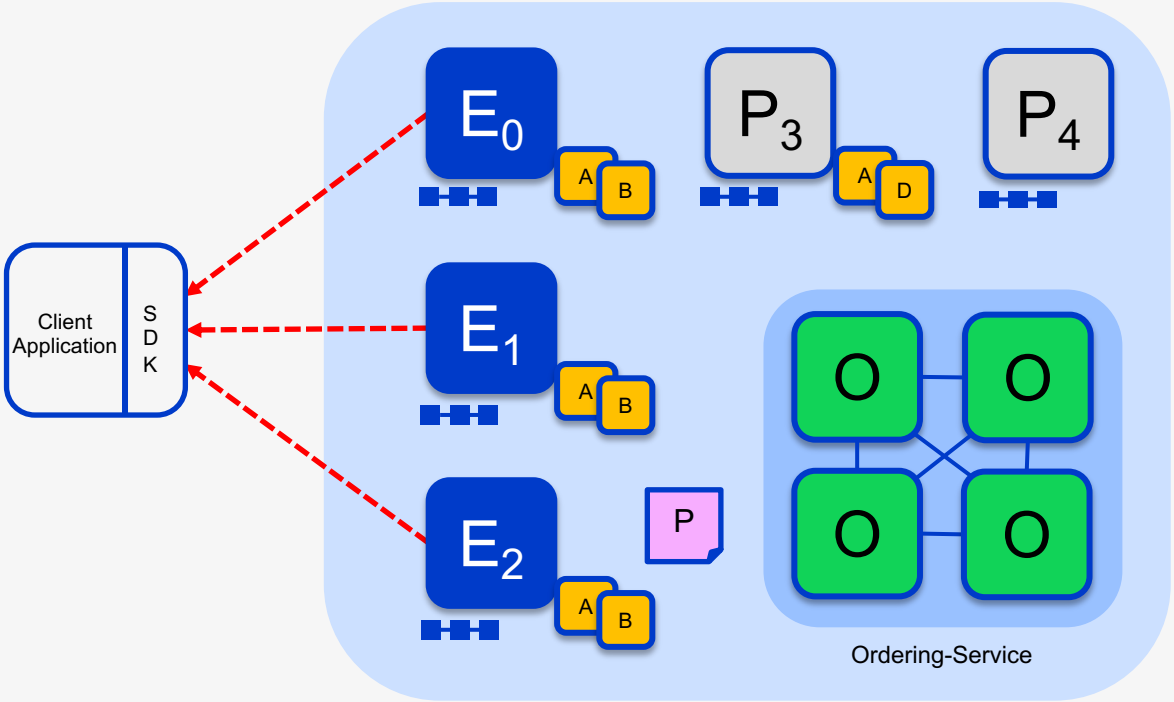
Each execution will capture the set of Read and Written data, called RW sets, which will now flow in the fabric.

Transactions can be signed & encrypted

Key:



Sample transaction: Step 3/7 – Proposal Response



Hyperledger Fabric Network

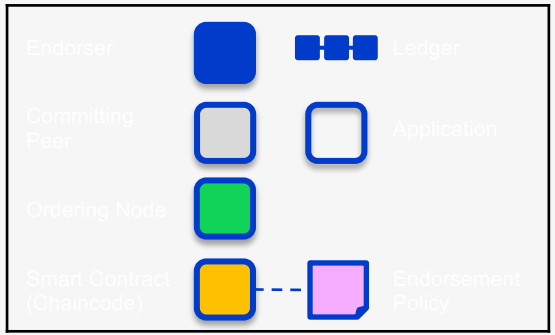
Application receives responses

RW sets are asynchronously returned to application

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

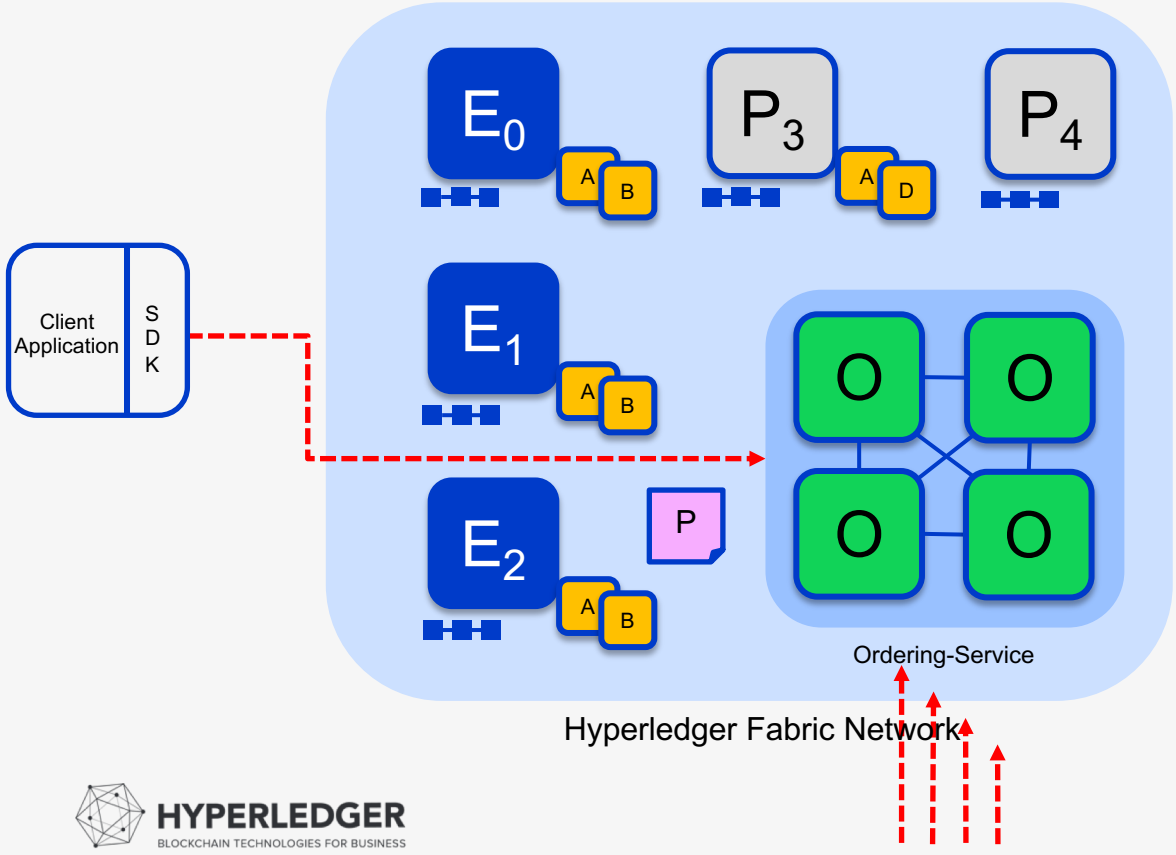
(This information will be checked much later in the consensus process)

Key:



HYPERLEDGER
BLOCKCHAIN TECHNOLOGIES FOR BUSINESS

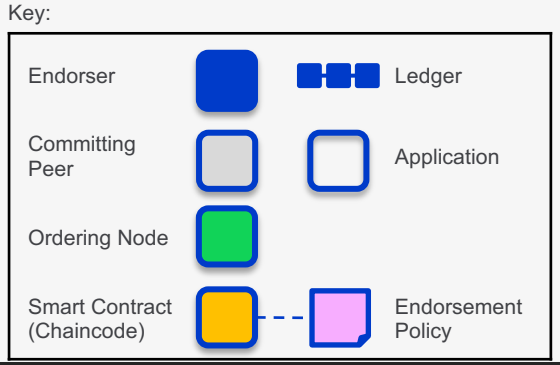
Sample transaction: Step 4/7 – Order Transaction



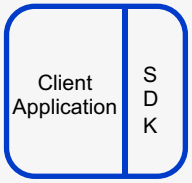
Responses submitted for ordering

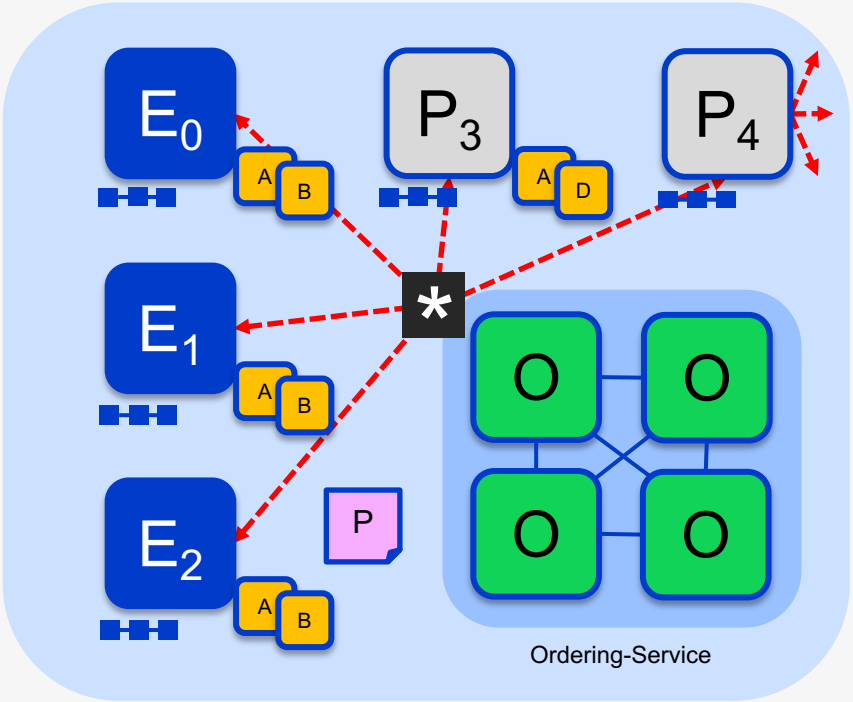
Application submits responses as a transaction to be ordered.

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



Sample transaction: Step 5/7 – Deliver Transaction





Hyperledger Fabric Network

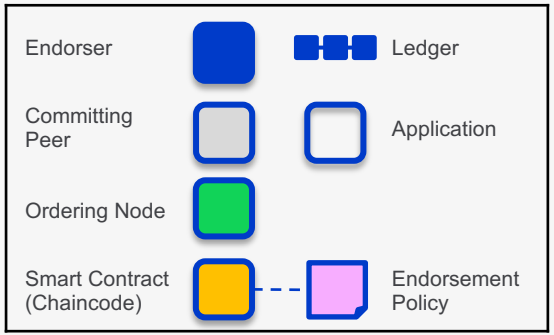
Orderer delivers to committing peers

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

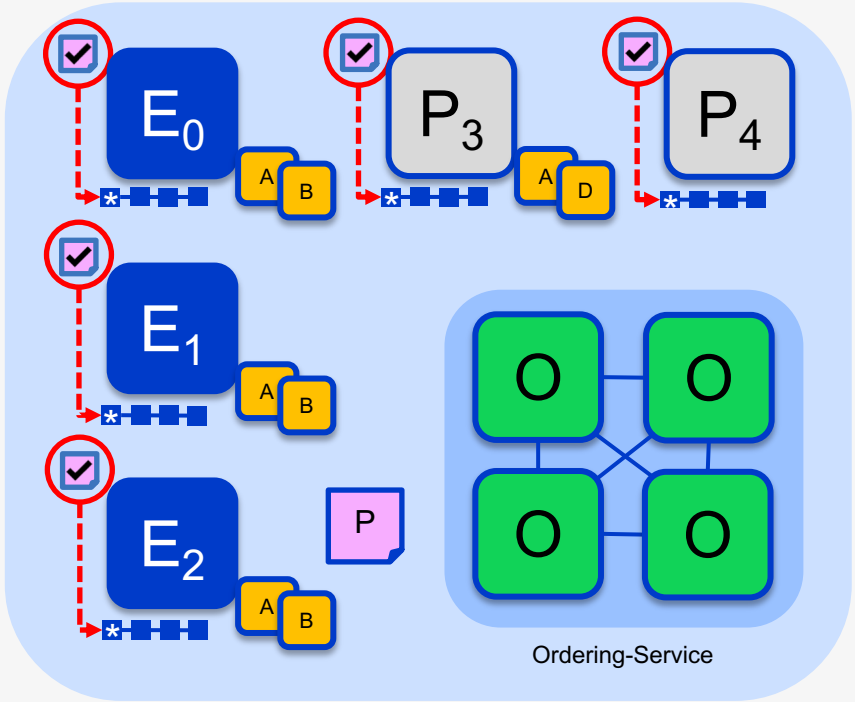
Different ordering algorithms available:

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

Key:



Sample transaction: Step 6/7 – Validate Transaction



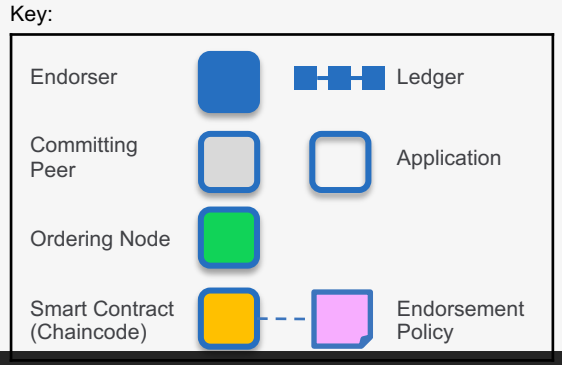
Hyperledger Fabric Network

Committing peers validate transactions

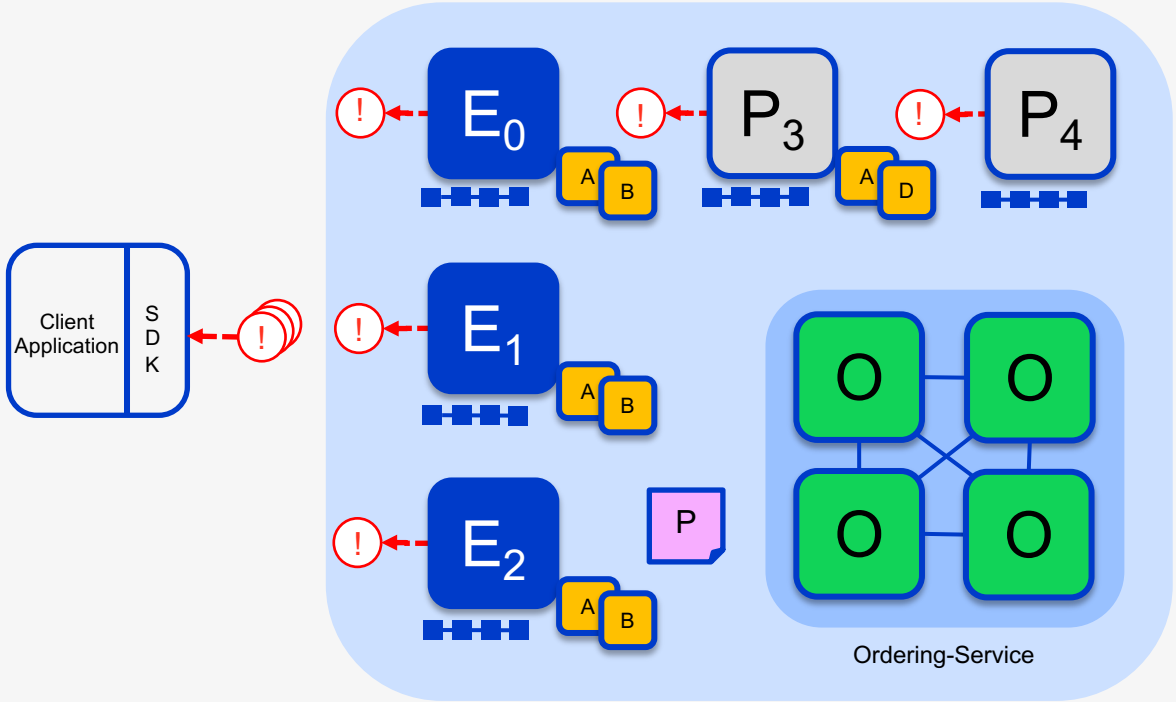
Every committing peer validates against the endorsement policy. Also check RW sets are still valid for current world state

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



Sample transaction: Step 7/7 – Notify Transaction



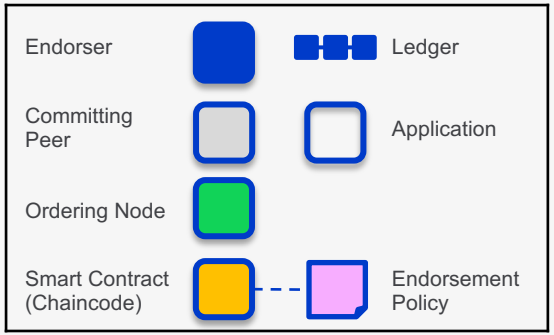
Hyperledger Fabric Network

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:



HYPERLEDGER
BLOCKCHAIN TECHNOLOGIES FOR BUSINESS

HYPERLEDGER

BLOCKCHAIN TECHNOLOGIES FOR BUSINESS

Thank you