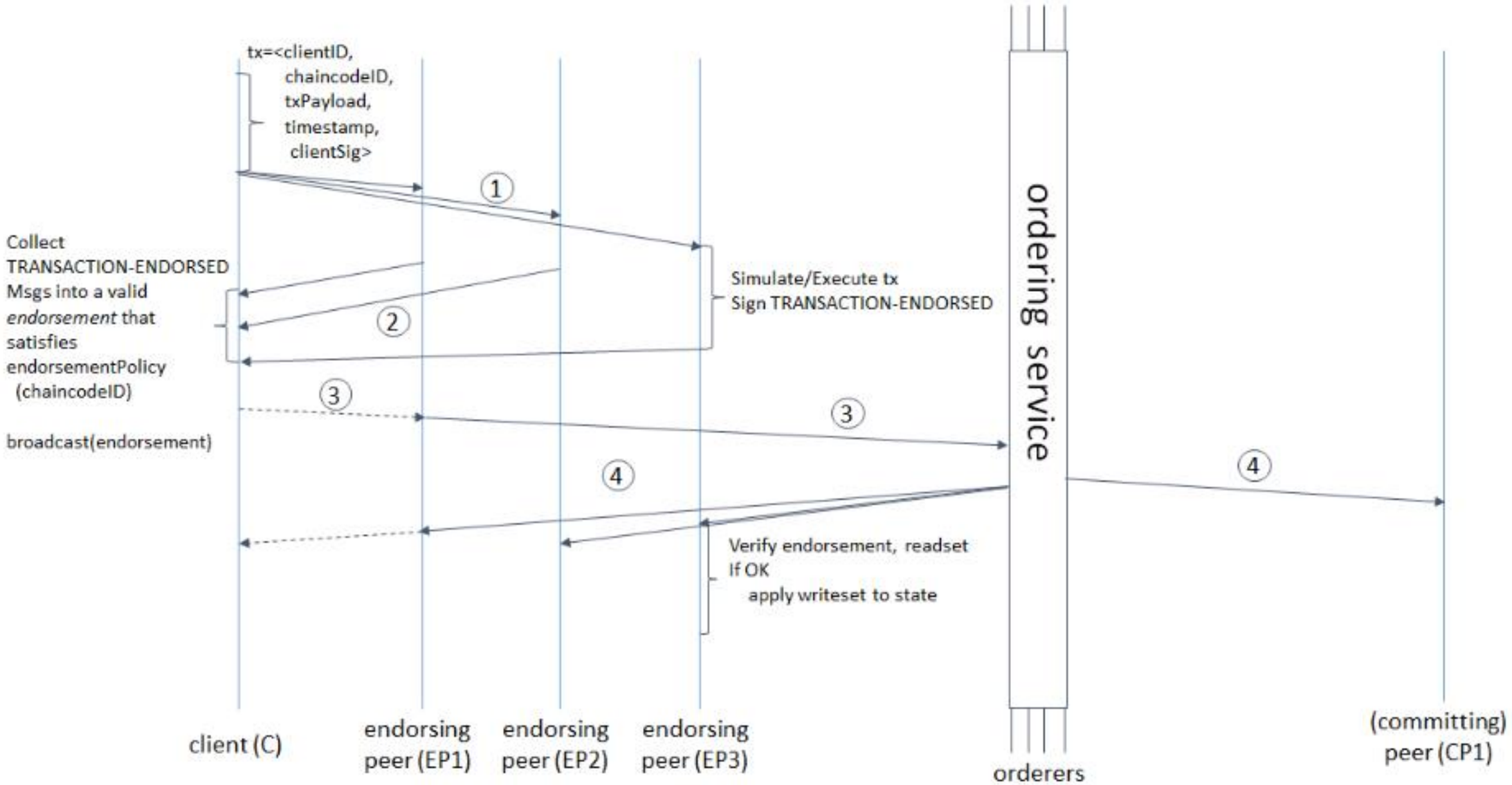
Hyperledger Fabric 구성과 흐름

1등 학습 조직 - 1주차

Fabric Transaction Flow



내용

1. CA
2. Peer
3. Orderer
4. HFC(Hyperledger Fabric Client)

1.Fabric-CA

네트워크 멤버(Peer, Orderer, HFC) 및 사용자에게 PKI 기반 인증서를 발급하는 인증 컴퍼넌트

- 각 구성원에게 루트인증서(rootCert) 하나를 발급하고 각 인증된 사용자에게 하나의 인증서(Ecert)를 발급
- Rest API 연동

1.Fabric-CA

실제 구성 시

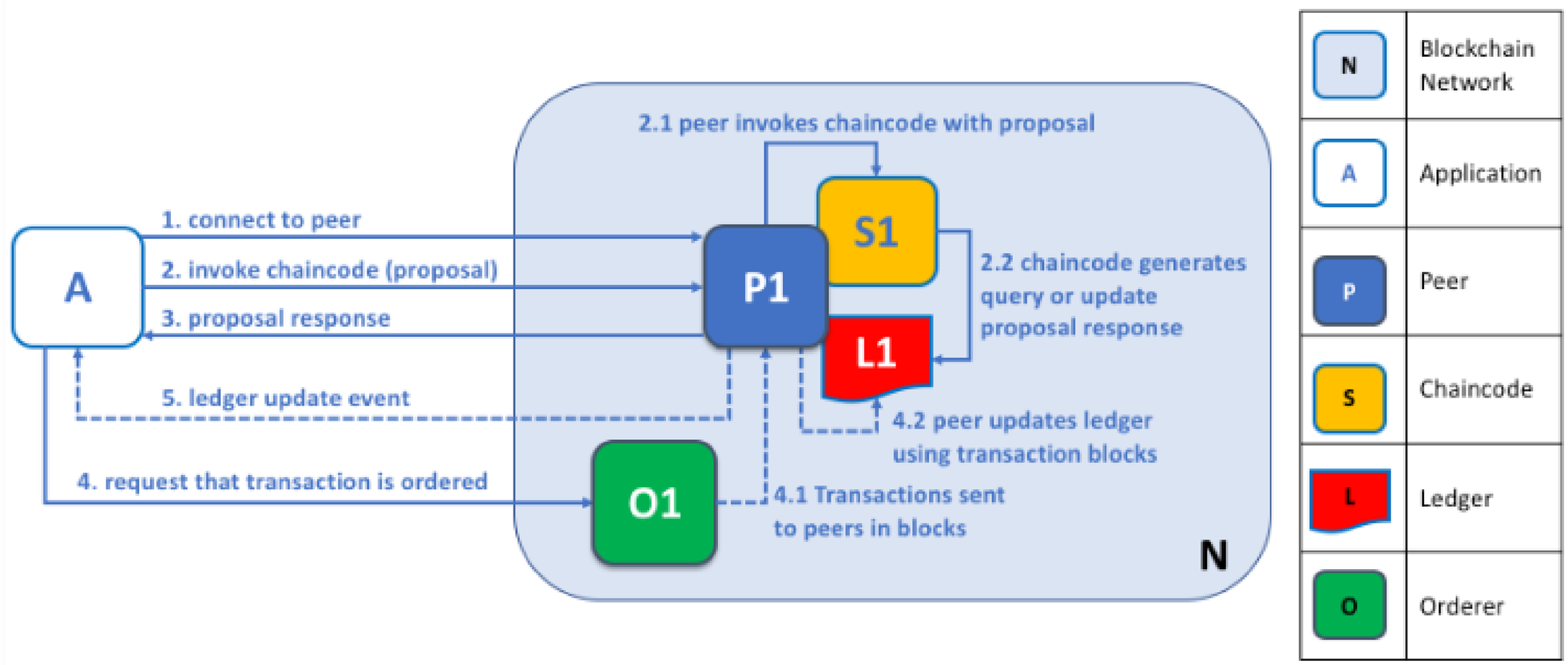
- Fabric-CA server / client 이용하여 인증서 발급
- Cryptogen 유틸리티와 crypto-config.yaml과 같은 설정 파일을 이용하여 MSP 전체 발급 가능
- 설정 우선 순위
  : Cli 사용 시 플래그 〉 도커 실행 환경 변수 〉 설정파일

2.피어(Peer)

분산 원장 데이터의 보증(Endorsement)과 검증(Validation)이 이뤄지는 기본적인 노드

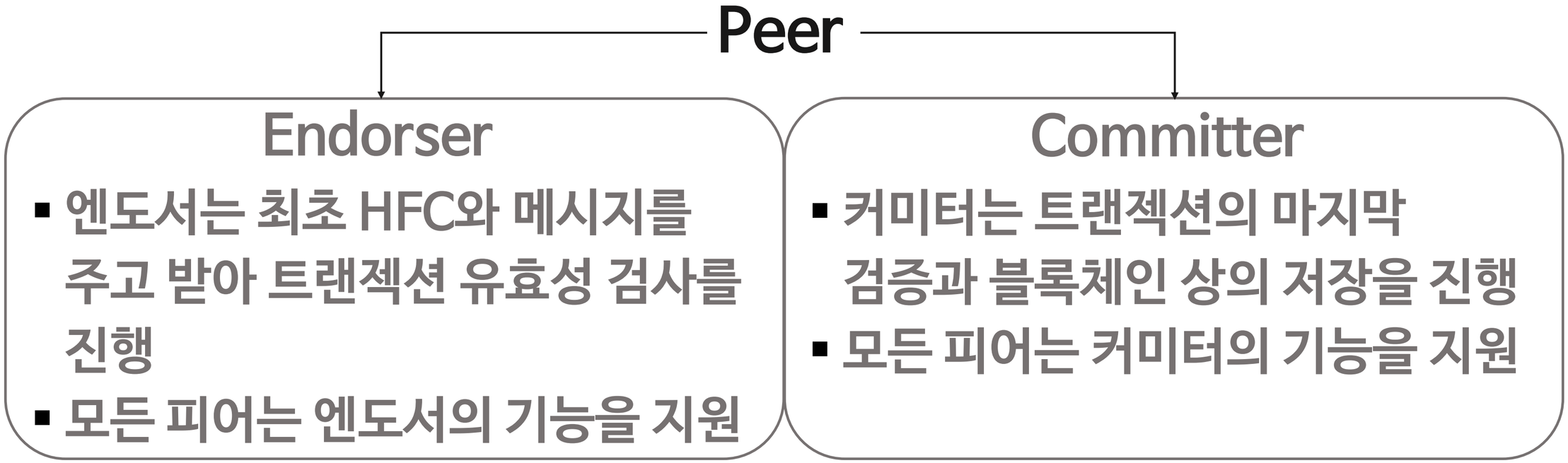
- 원장 데이터 및 스마트계약(SmartContract)이 이뤄지는 블록체인 컴포넌트
- 다뤄지는 원장은 스마트계약에 의해 생성된 모든 트랜젝션을 변치 않는 형태로 만들어진 기록
- 스마트계약과 원장은 각각 네트워크에서 블록체인의 분산 처리 및 분산 저장하도록 하는데(블록체인화) 사용
- 피어의 P2P 구성으로 인해 다중화 분산 원장과 스마트계약이 존재 하게 되므로 단일 장애 지점(Single Point of Failure)을 방지 하는 수단(분산화)

2.피어(Peer)



- HFC는 피어와 연결되어 체인코드의 배포 및 인스턴스화, 버전업 등의 관리
- 트랜젝션 처리를 위해 HFC와 피어는 상호 연동 하여 보증/검증 과정을 거침

2.피어(Peer)


Peer
Endorser
▪ 엔도서는 최초 HFC와 메시지를 주고 받아 트랜젝션 유효성 검사를 진행
▪ 모든 피어는 엔도서의 기능을 지원
Committer
▪ 커미터는 트랜젝션의 마지막 검증과 블록체인 상의 저장을 진행
▪ 모든 피어는 커미터의 기능을 지원

2.피어(Peer)

Fabric Network에서 엔도서(Endorser)와 커미터(Committer)의 구분방법

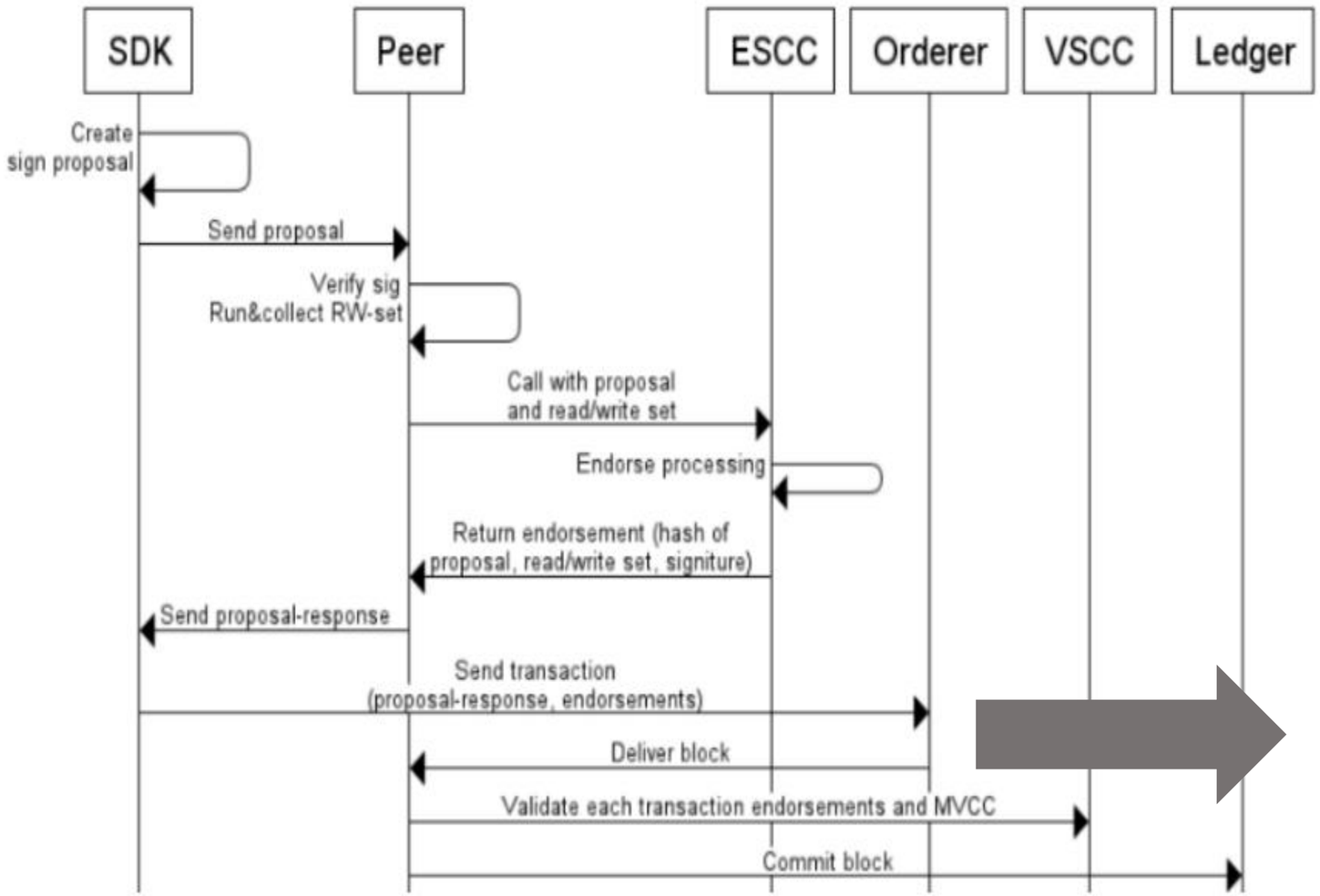
- CORE_PEER_ENDORSER_ENABLED (V1.0.x 사용 방법)
- Chaincode의 Install/Instantiate 여부 (잘못된 정보)
- HFC에서 구조에 맞게 설정 (V1.1 이후 방법)

3.오더러(Orderer)

Fabric Network의 합의알고리즘이 동작 하는 노드

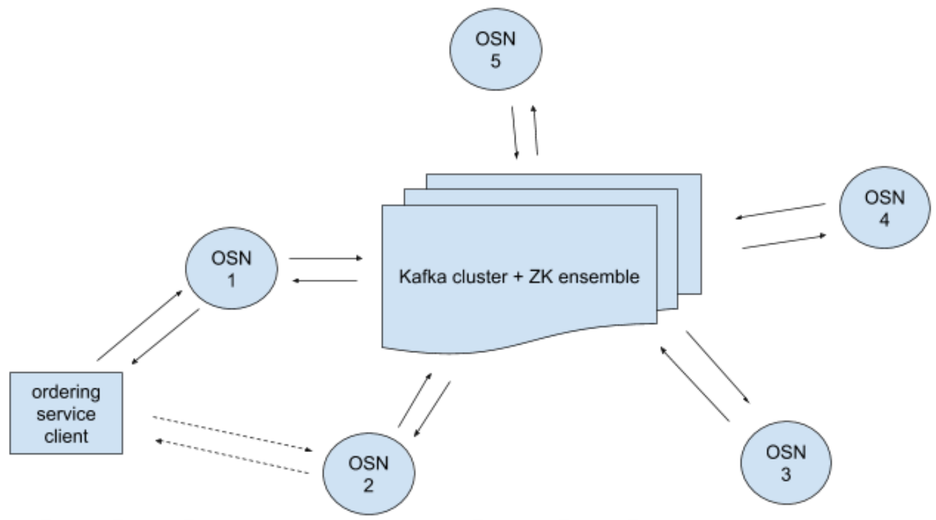
- 피어와 HFC의 보증 단계 이후 유효성이 판별된 트랜젝션에 대한 처리 순서를 정렬해 주는 노드
- 현재까지(v1.2) SOLO, KafKa 두 가지 방법의 합의 알고리즘 지원(향후 RAFT, SBFT 지원 예정)
- Fabric Network의 구성 채널에 대해 선착순에 따라 정렬
- Fabric Network 전체에 공통으로 연결되는 구조로 암호화된 ID와 자료를 포함

3.오더러(Orderer)



- HFC와 엔도서의 보증 단계 이후 유효성이 판명된 트랜젝션에 대한 처리 순서(Ordering)을 결정 하기 위해 Orderer로 트랜젝션은 전송된다.
- 블록생성 시간(BatchTimeout) 설정과 블록 당 트랜젝션수 (BatchSize)설정에 의해 블록이 생성된다.
- 블록당 트랜젝션 수 (BatchSize)설정은 최대 트랜젝션 수(MaxMessageCount) 와 한계 트랜젝션 사이즈(AbsoluteMax Bytes), 최대 트랜젝션 사이즈 (Preferre MaxBytes)로 세분화 되어 있다.

3.OSN(Ordering Service Node)



Kafka 합의 알고리즘

- 각 채널의 원장은 카프카 별도의 단일 파티션 Topic에 매핑
- 오더러(Orderer)의 broadcast RPC를 통해 트랜젝션 수신 후 클라이언트의 Write 권한 확인
- 카프카의 적절한 파티션으로 중계
- 오더러(Orderer)는 설정값을 이용하여 블록생성 후 Deliver RPC 클라이언트 수신자에게 배포

3.OSN(Ordering Service Node)

솔로(Solo)와의 차이점

- 솔로 합의 방식은 오더러(Orderer)프로세스에서 고루틴을 이용하여 채널 별 트랜젝션 오더링
- 카프카 합의 방식은 카프카(Kafka)-주키퍼(Zookeeper) 클러스트를 이용하여 채널 별 트랜젝션 오더링
- 서비스가 매우 적은 경우 솔로를 사용가능 하지만 다양한 서비스가 존재 하는 경우 카프카 합의 방식을 권고
  : 단일 장애 지점(Single Point of Failure) 예방 차원

3.OSN(Ordering Service Node)

합의 방식 선택 기준

- 구성 채널의 개수에 따라
- 트랜젝션 부하량에 따라
- **상용 서비스 환경에서는 안정성을 고려 하여 카프카 합의 방식을 권고**

4.HFC(Hyperledger Fabric Client)

Hyperledger Fabric Network와 통신 할 수 있는 웹/앱 어플리케이션

- 클라이언트 SDK(Software Development Kit)를 지원하여 체인코드(Chaincode) 배포 및 설치, 업데이트 등의 운용 기능 제공
- CA 클라이언트 SDK(Software Development Kit)를 지원하여 사용자 계정의 등록 및 인증서 발급 기능 제공
- 클라이언트 SDK(Software Development Kit)를 지원하여 Fabric Network의 트랜젝션 처리 진행
- 지원 언어로는 Go, Node.js, Java, Rest API 버전
- 비 결정적 트랜젝션 처리를 위해 서비스 로직 중 일부 기능 처리 필요