하이퍼레저 패브릭 기반의 프라이빗 블록체인 개발 과정

2022-02-15

빅픽처랩㈜

안 휘

전체 목차

- **하이퍼레저 패브릭 소개 (1 day)**
  - 블록체인이란?
  - 블록체인 기술들과 하이퍼레저 패브릭
  - 하이퍼레저 패브릭 구조 개요
  - 하이퍼레저 패브릭 구동 실습
- **하이퍼레저 패브릭 개발 (2 day)**
  - 하이퍼레저 패브릭 아키텍처
  - 블록체인 기반 소프트웨어 시스템 개발
  - 스마트 컨트랙트 개발
  - 하이퍼레저 패브릭 스마트 컨트랙트 개발 실습

하이퍼레저 패브릭 아키텍처

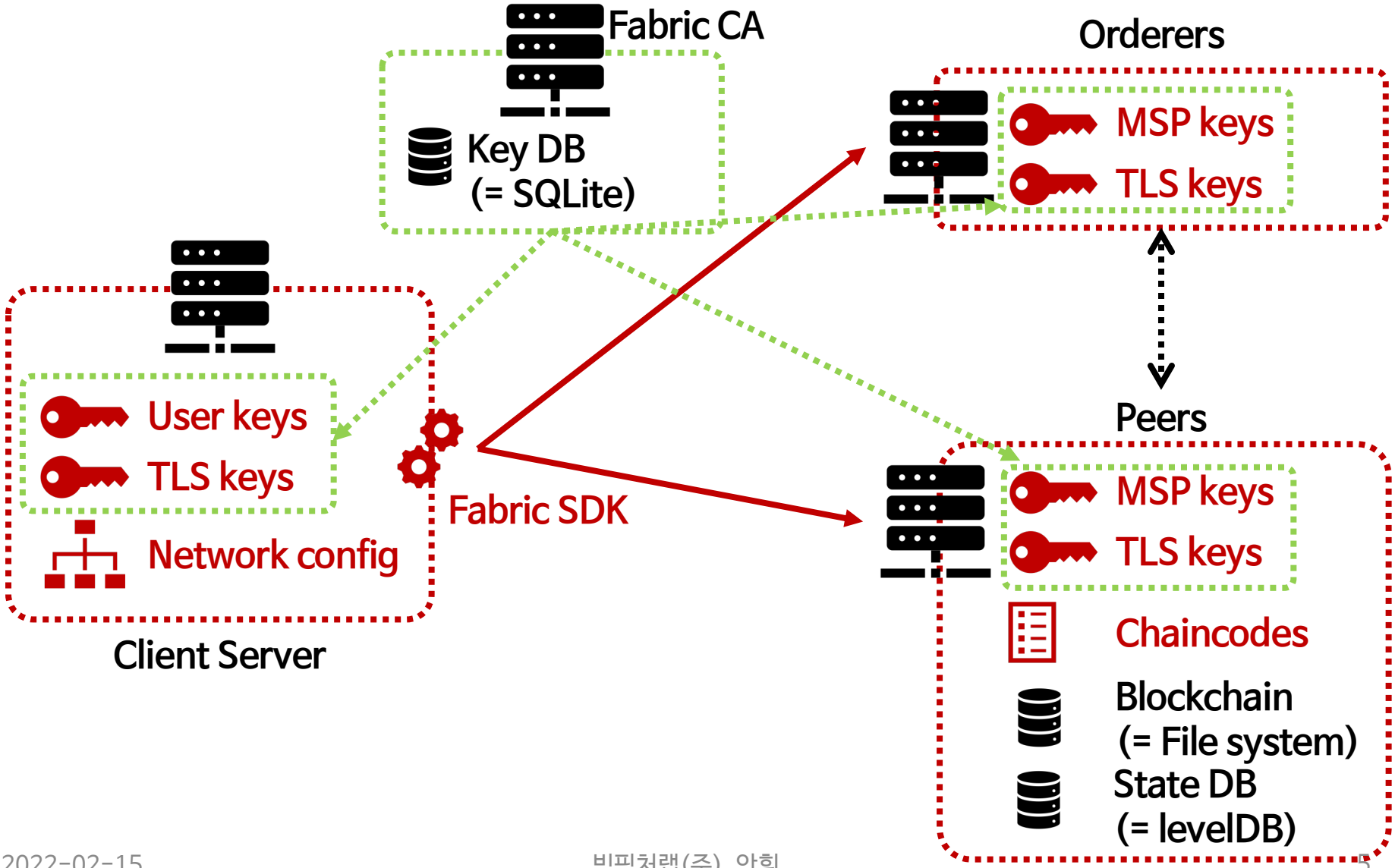
2022-02-15

빅픽처랩(주)

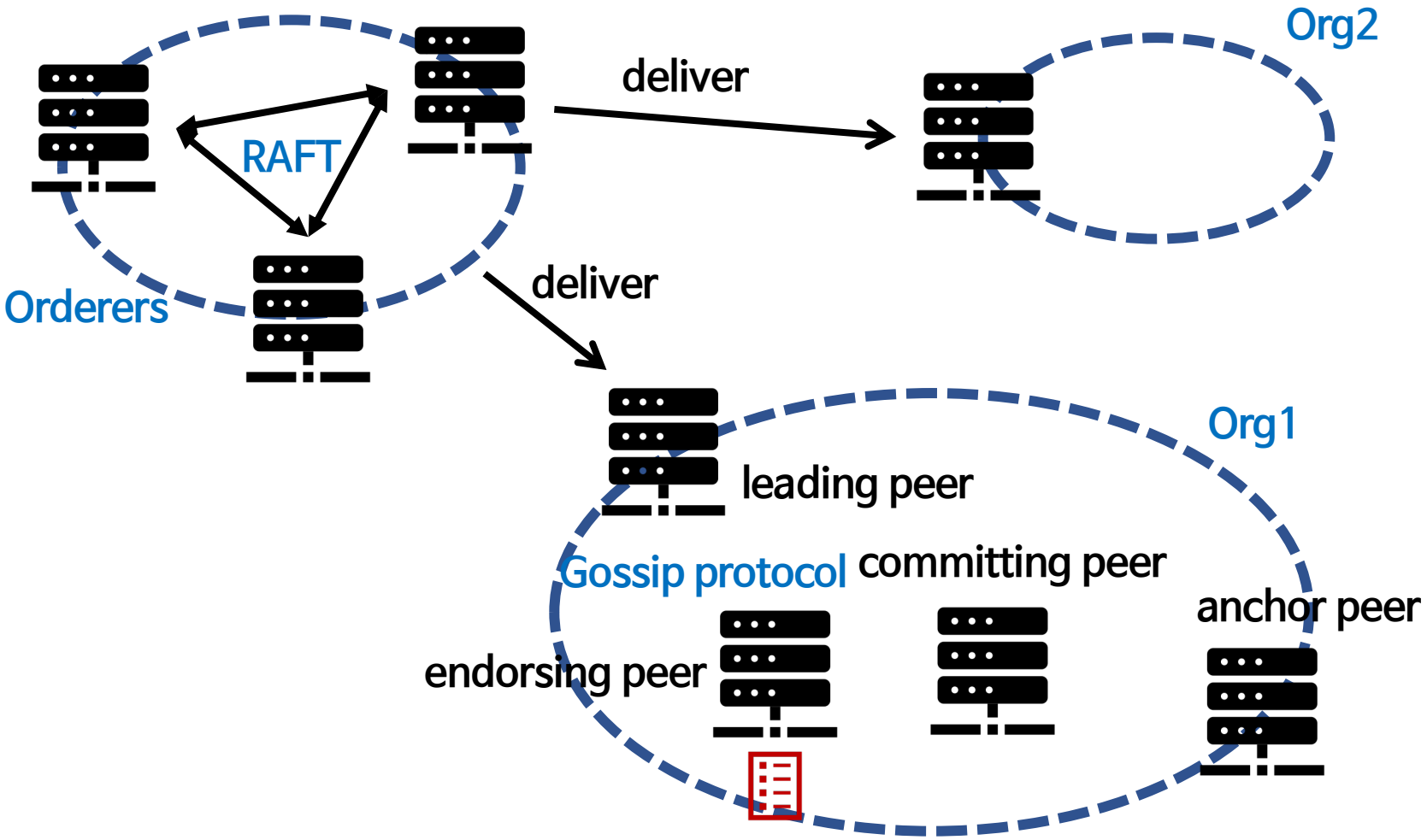
안휘

목차

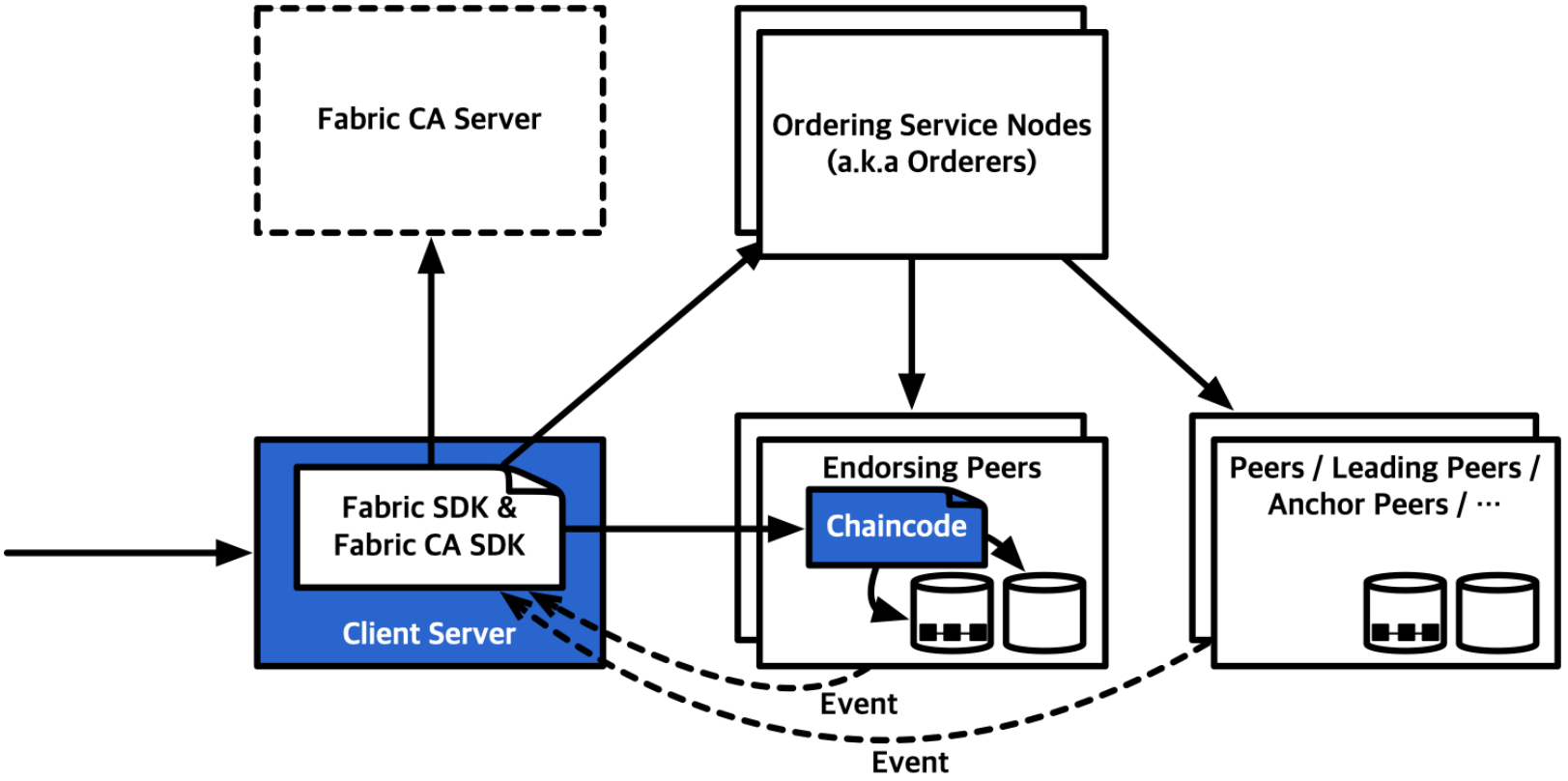
1. 하이퍼레저 패브릭 구조



1. 하이퍼레저 패브릭 구조

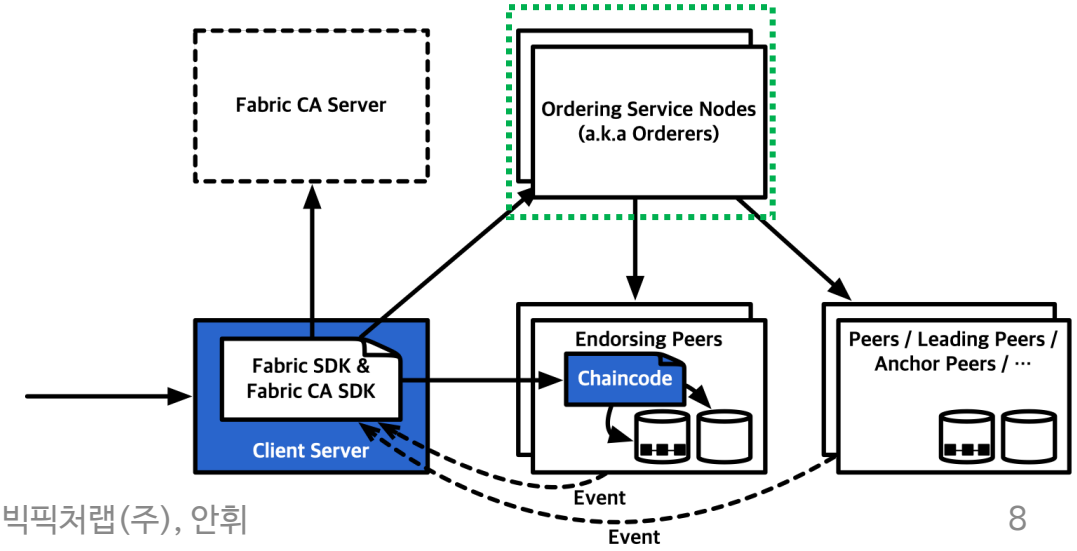


1. 하이퍼레저 패브릭 구조



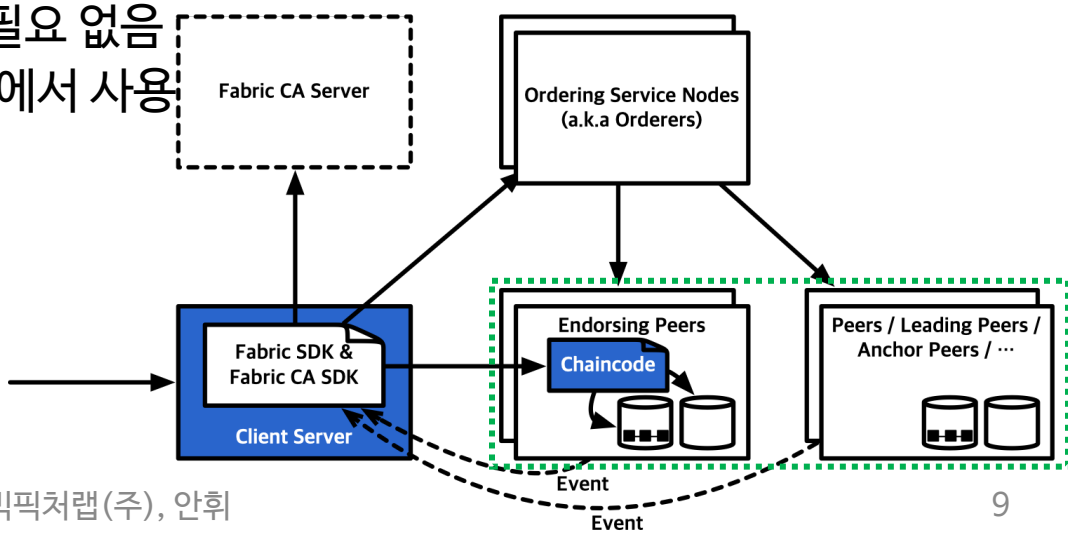
1. 하이퍼레저 패브릭 구조

- Orderers
  - 트랜잭션을 순서대로 나열함
    - Apache Kafka
  - 블록을 생성함
  - **검증은 하지 않음 → Committing Peer가 담당**
  - RAFT: Orderer들 사이의 Crash Tolerance를 보장, Byzantine Fault Tolerance는 보장하지 못함



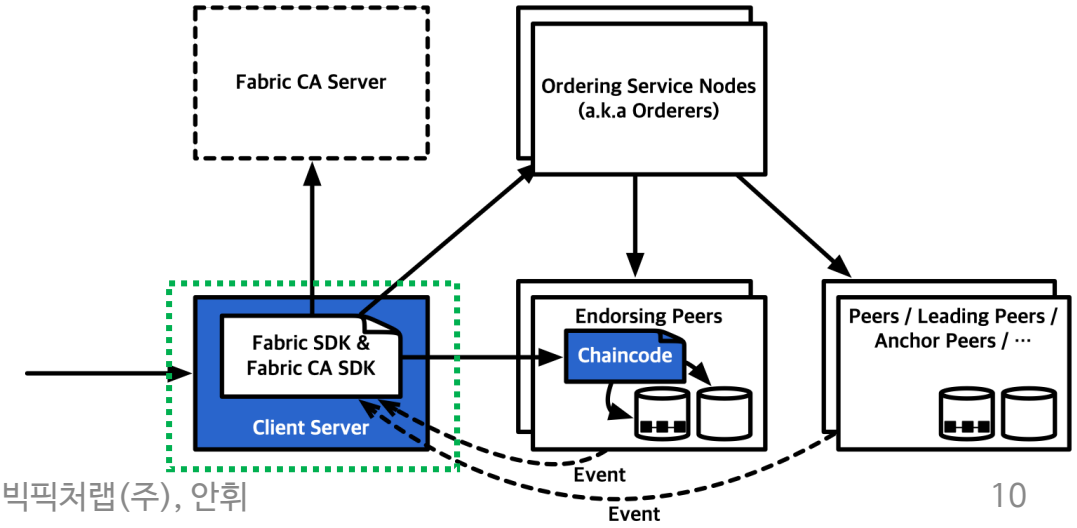
1. 하이퍼레저 패브릭 구조

- **Committing Peer**
  - Orderer로부터 블록을 받아 저장하는 Peer (즉, 채널에 가입된 모든 Peer)
- **Endorsing Peer**
  - Chaincode를 실행하는 Peer
  - 트랜잭션을 미리 실행해보고, 에러가 있는지 없는지 확인
- **Anchor Peer**
  - 조직 외부에서 조직 내 Peer들을 찾을 때 사용
  - 외부에서 모든 Peer를 알고 있으면 필요 없음
  - Service Discovery, Private Data에서 사용
- **Leading Peer**
  - 조직을 대표하여, Orderer로부터 블록을 deliver 받음
  - 자동 선출



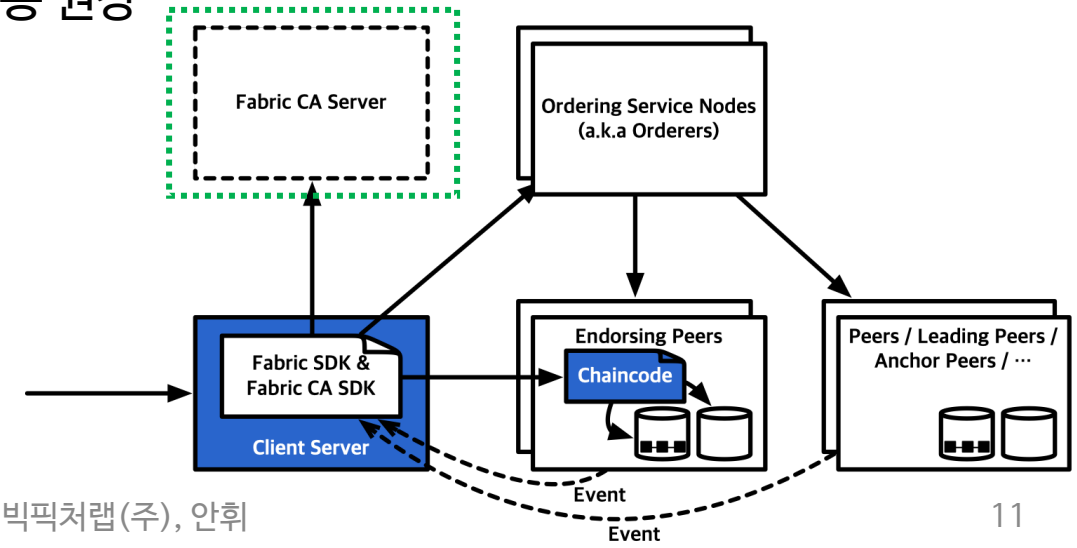
1. 하이퍼레저 패브릭 구조

- Client Server
  - 트랜잭션 생성
  - 모든 네트워크 엔터티들과 접속 필요
    - Fabric CA: End user 키 Register 및 Enroll
    - Endorsing Peer 또는 Anchor Peer: Endorsement 생성
    - Orderer: 트랜잭션을 전송
      (Client Server가 수집한 Endorsement와 함께 직접 보내야 함)



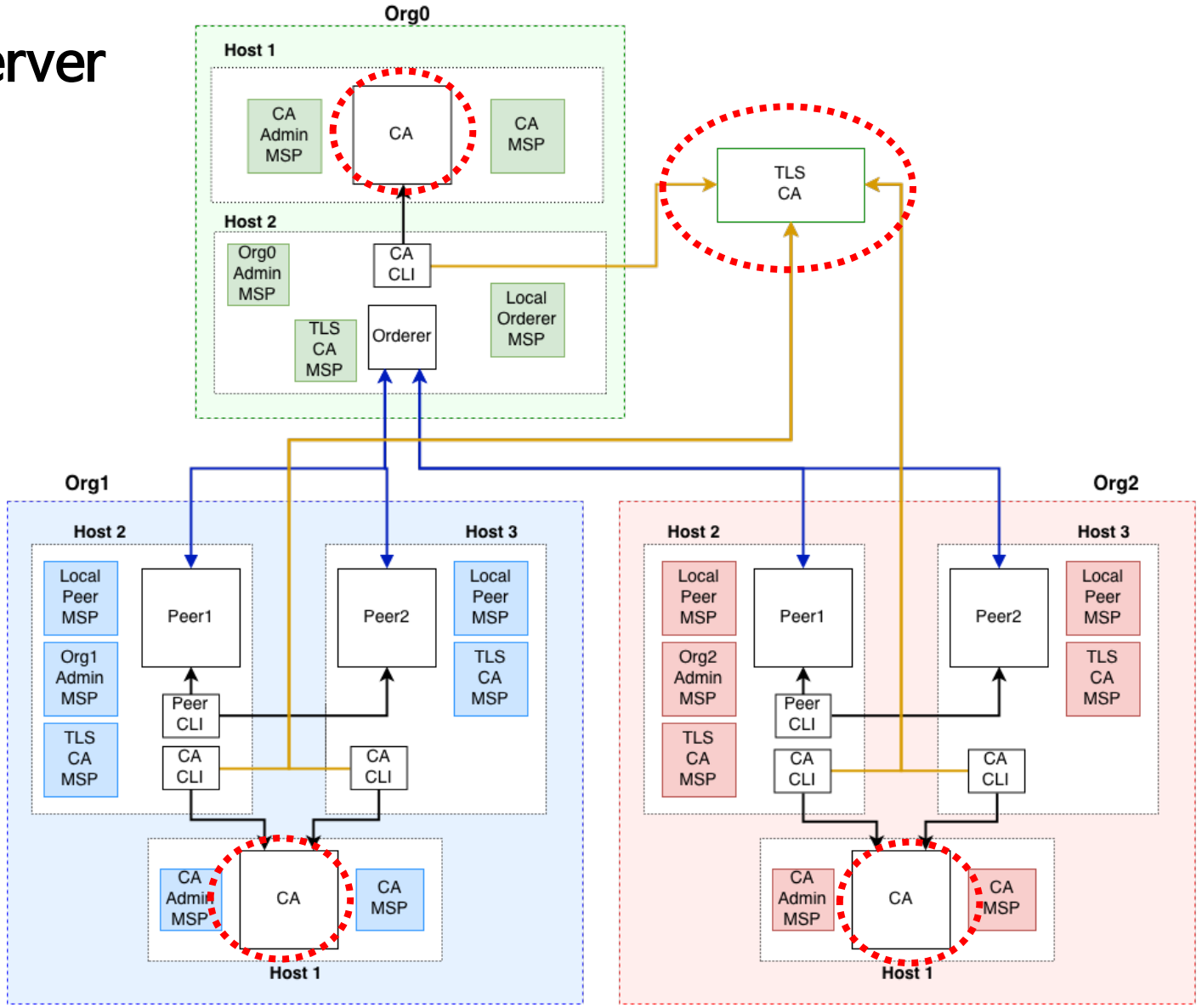
1. 하이퍼레저 패브릭 구조

- Fabric CA Server
  - Optional (하지만 대부분 사용)
  - 다음 키 들을 관리 (register, enroll, revoke)
    - 사용자 키: 트랜잭션 서명에 사용
    - Orderer/Peer 키: MSP로 사용
    - TLS 키: TLS 통신에 사용
  - 키 생성이 완료되면, 오프라인 상태가 되어도 상관 없음
  - 테스트에서는 MSP, TLS 키는 cryptogen 으로 생성, 사용자만 Fabric CA 사용
  - 프로덕션에서는 모두 Fabric CA 사용 권장



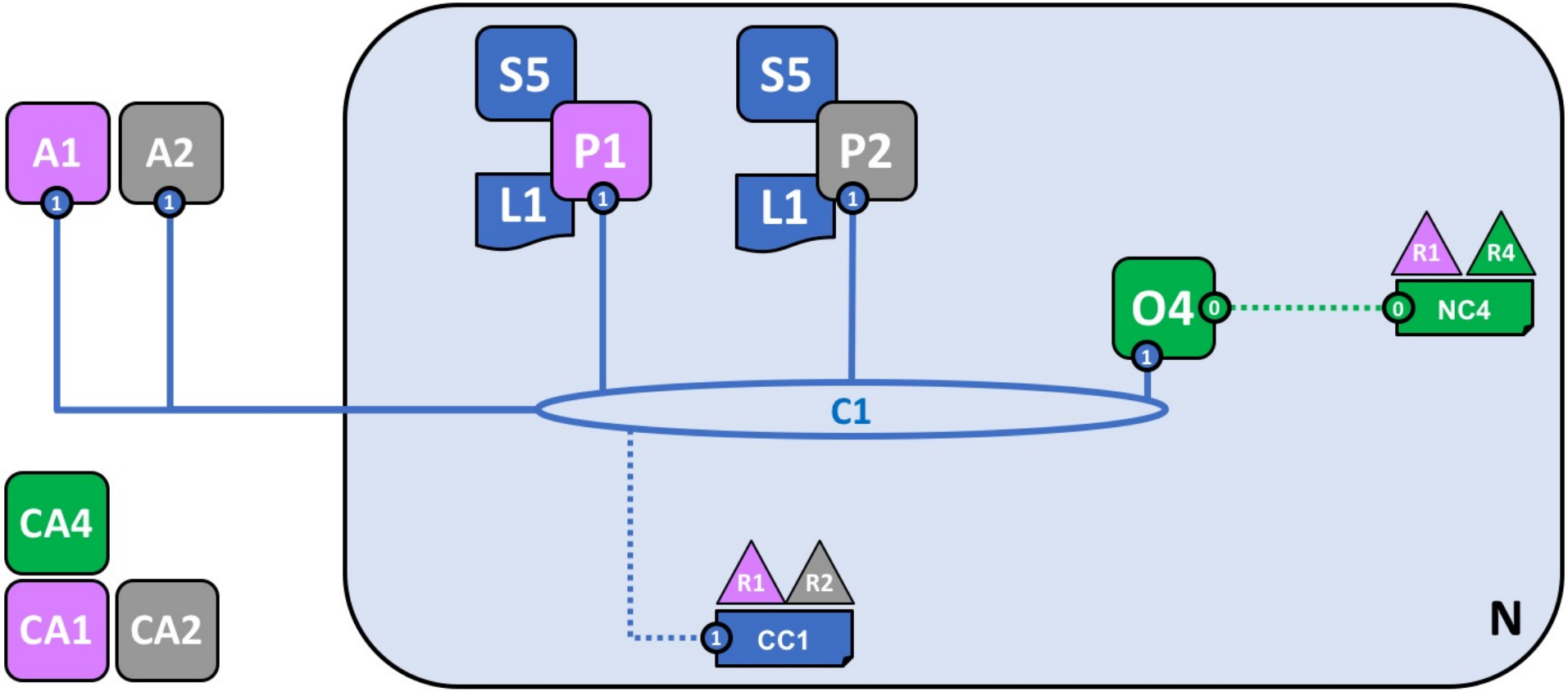
1. 하이퍼레저 패브릭 구조

- Fabric CA Server



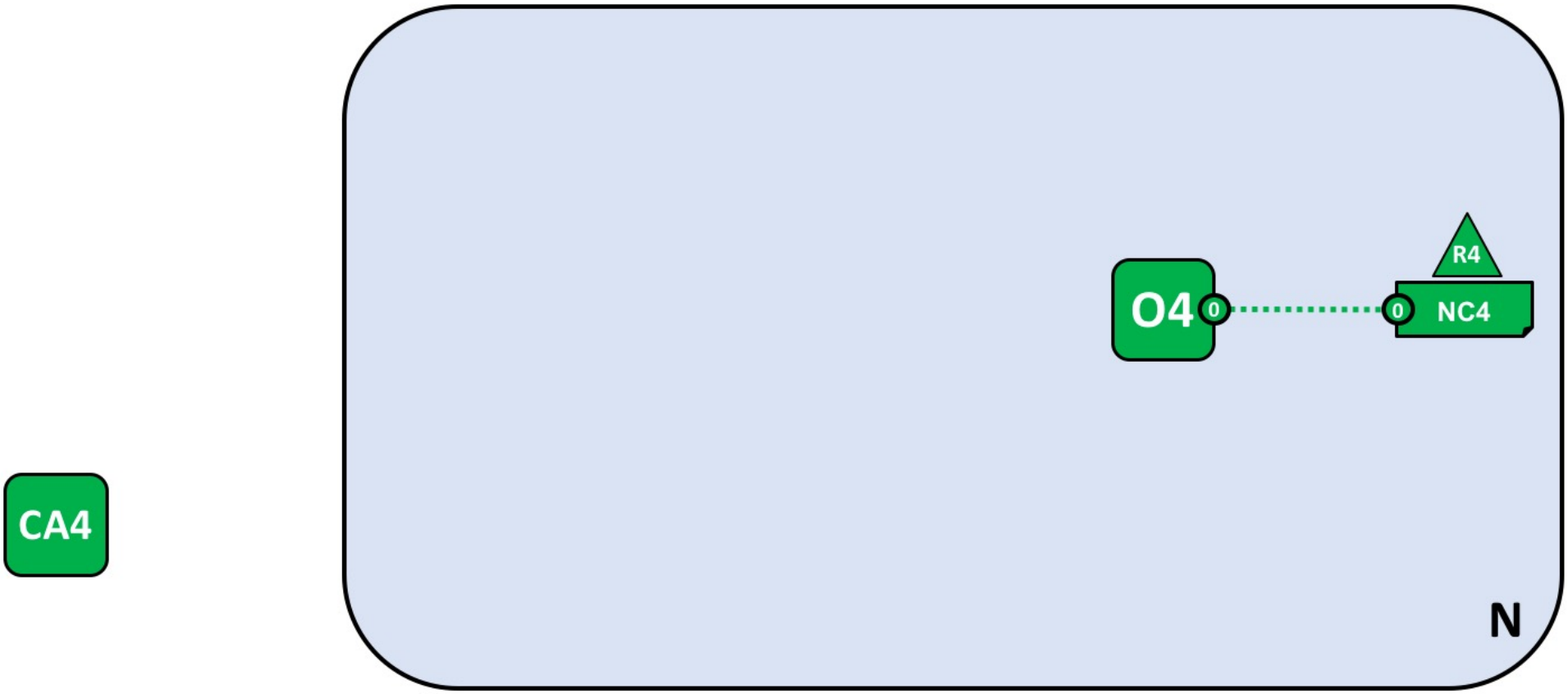
2. 하이퍼레저 패브릭 네트워크 구동 시나리오

- 최종 네트워크



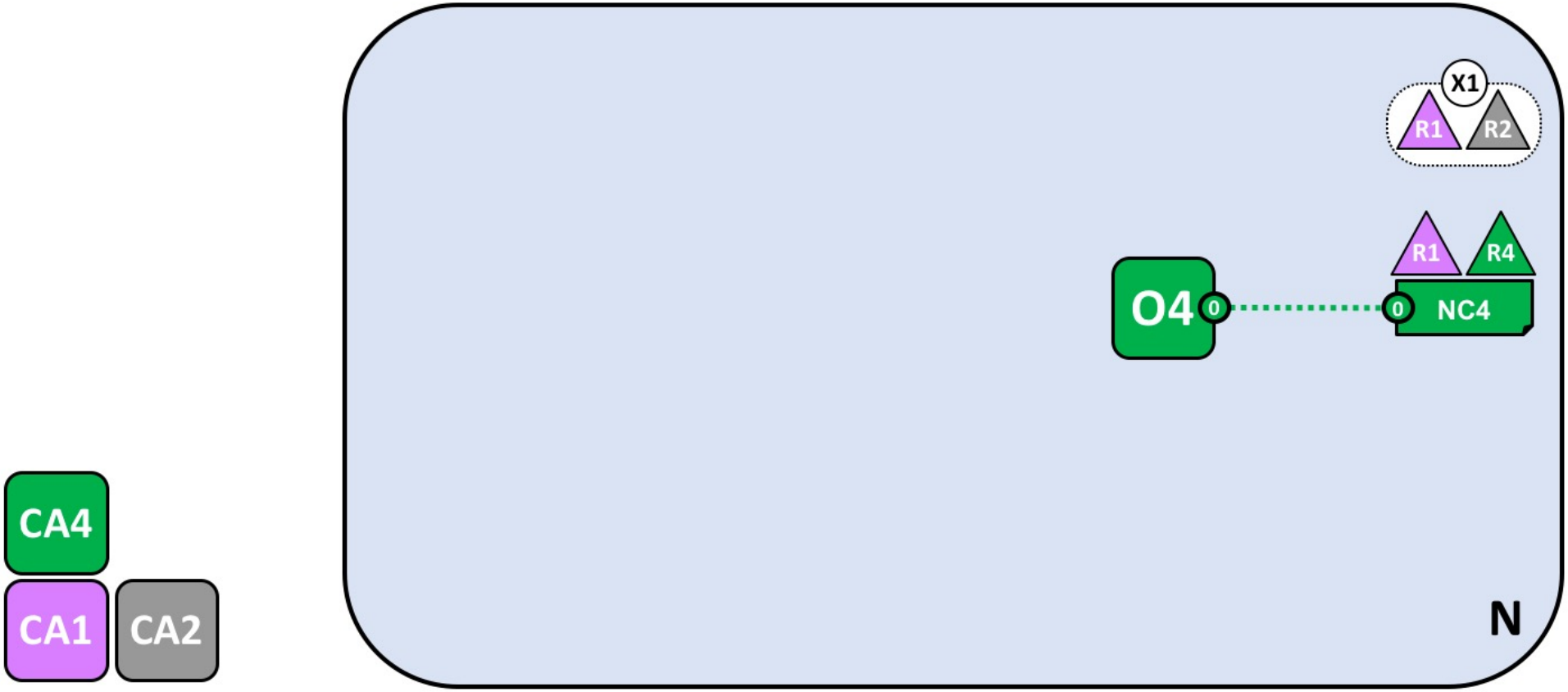
2. 하이퍼레저 패브릭 네트워크 구동 시나리오

- Orderer 구동



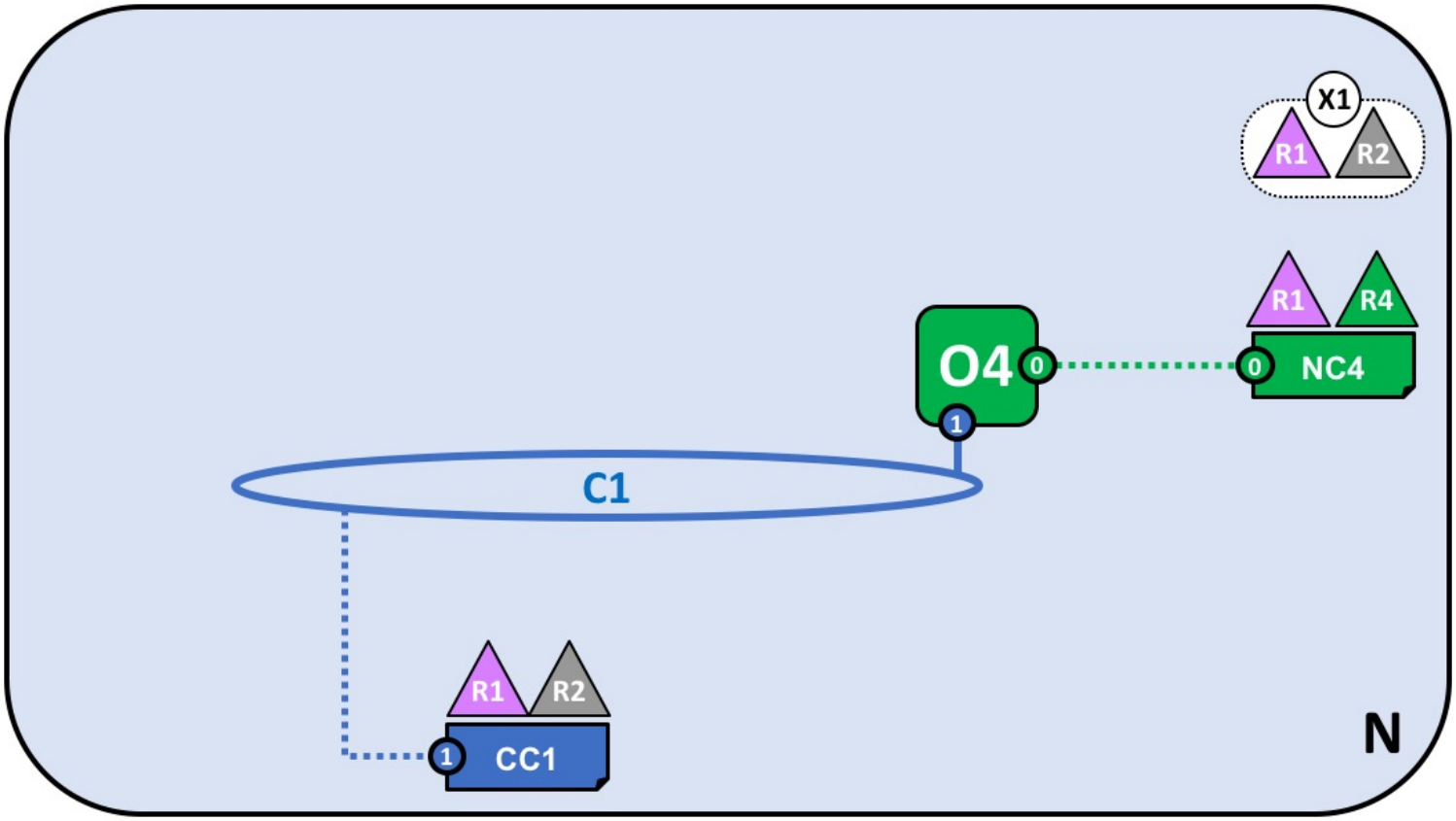
2. 하이퍼레저 패브릭 네트워크 구동 시나리오

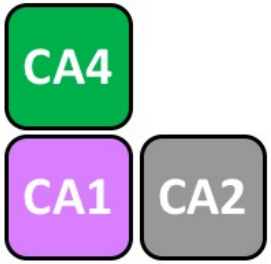
- 콘소시움 정의



2. 하이퍼레저 패브릭 네트워크 구동 시나리오

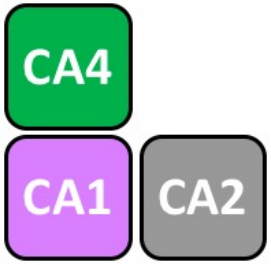
- 채널 생성

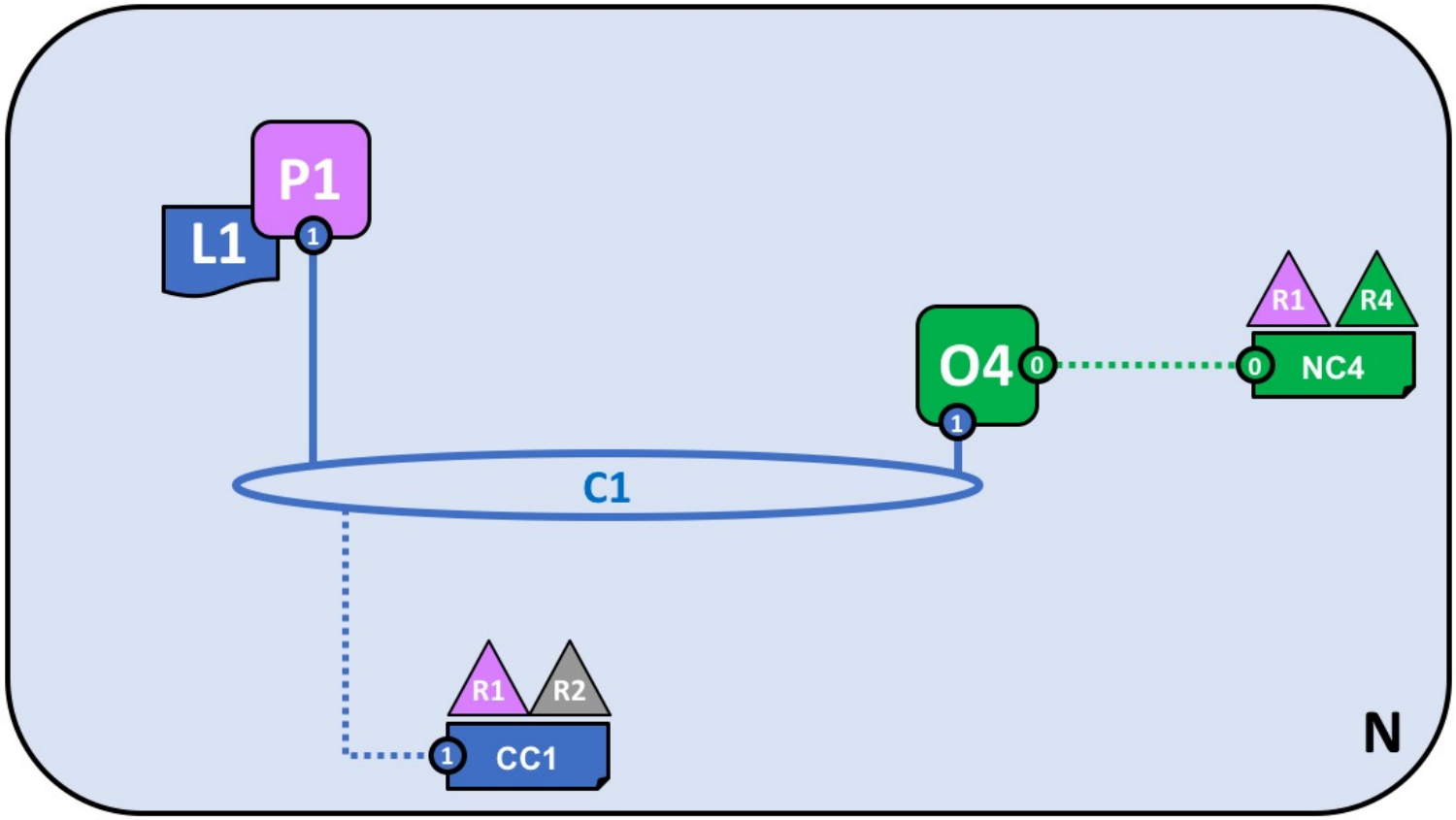




2. 하이퍼레저 패브릭 네트워크 구동 시나리오

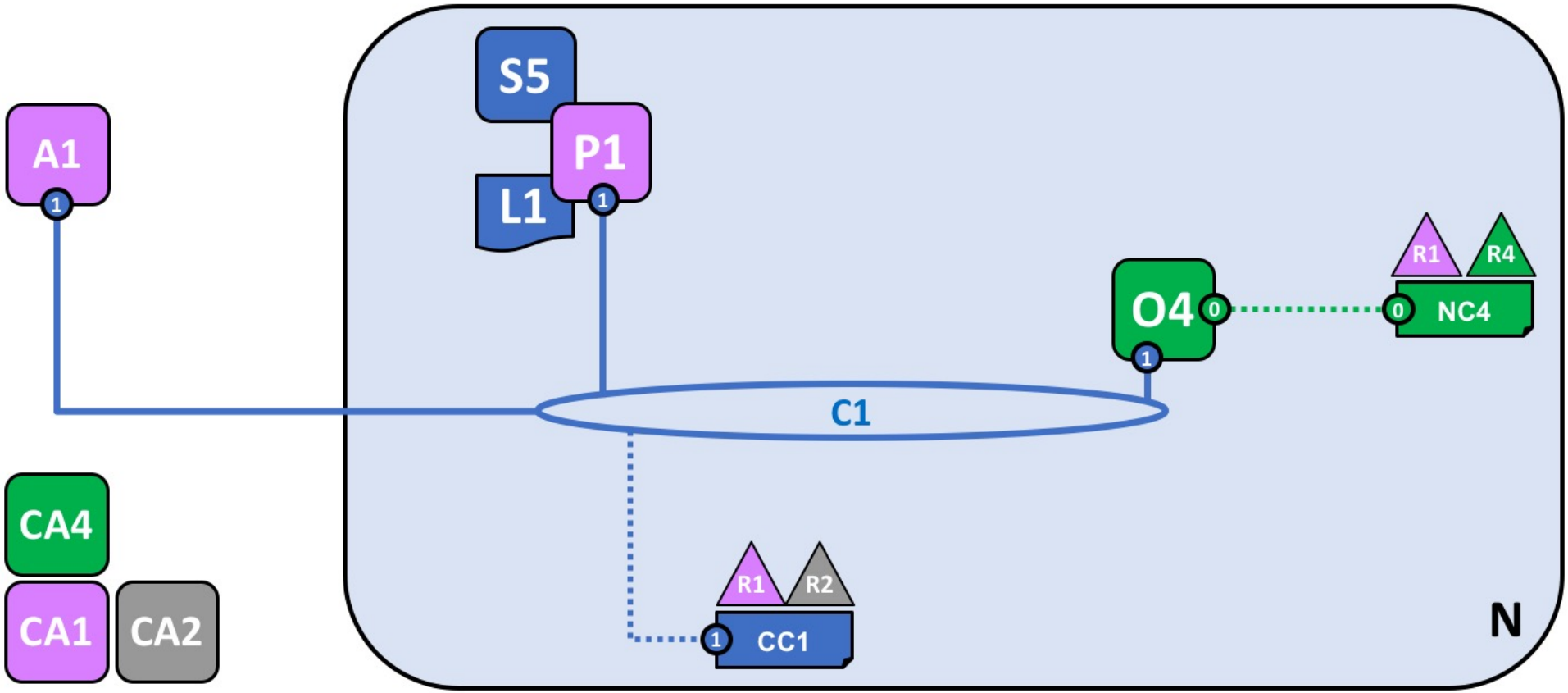
- Org1의 Peer1 채널 조인





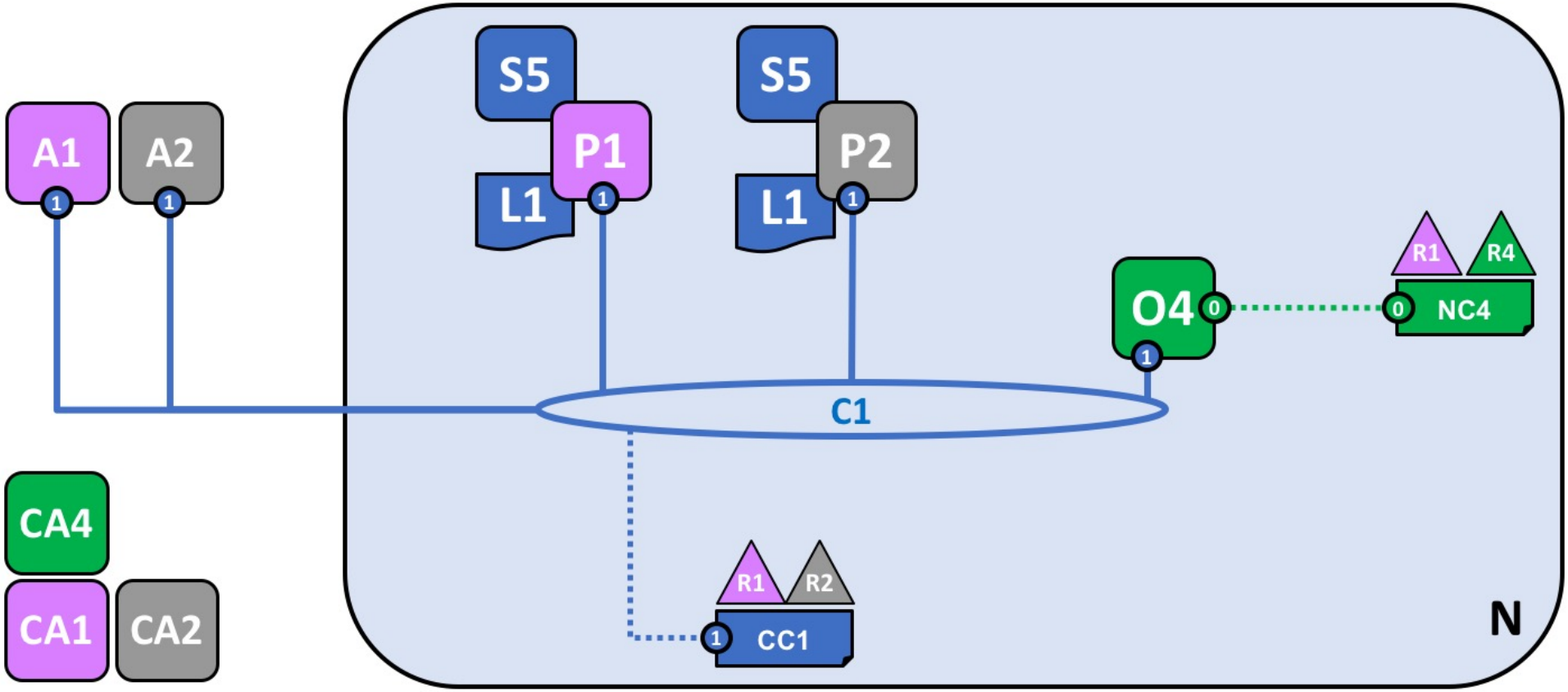
2. 하이퍼레저 패브릭 네트워크 구동 시나리오

- Org1의 Peer1에 체인코드 배포하고, Client Server 구동

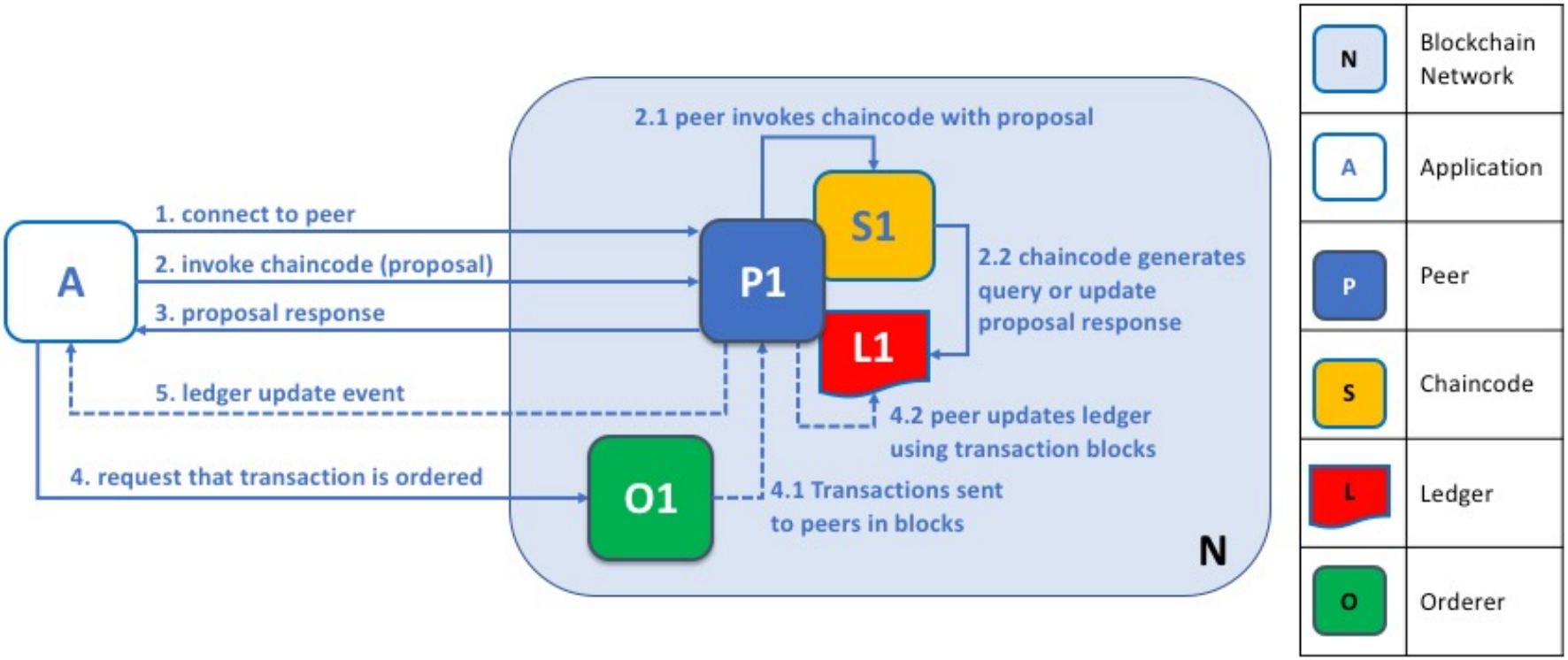


2. 하이퍼레저 패브릭 네트워크 구동 시나리오

- Org2에 대해 채널 조인, 체인코드 배포, Client Server 구동

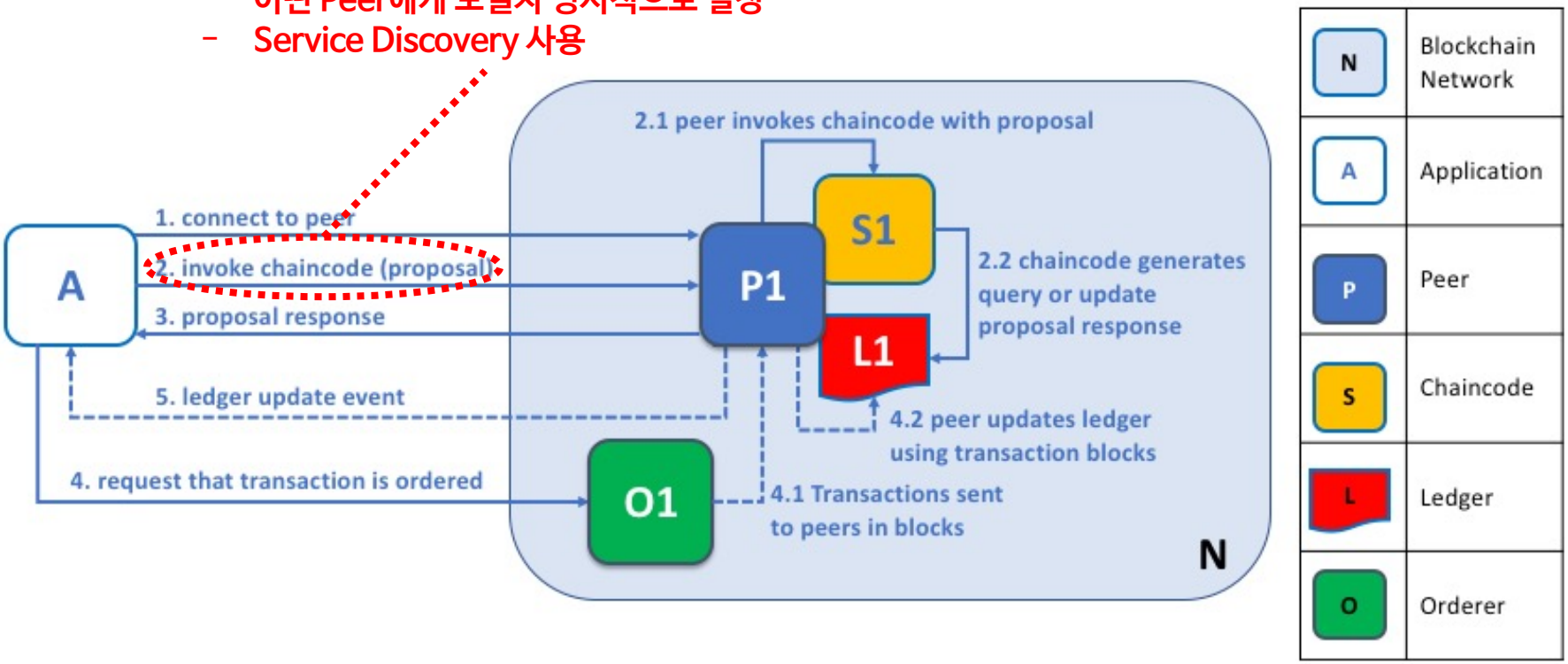


3. 트랜잭션 전송 시나리오



3. 트랜잭션 전송 시나리오

- 어떤 Peer에게 보낼지 명시적으로 결정
- Service Discovery 사용



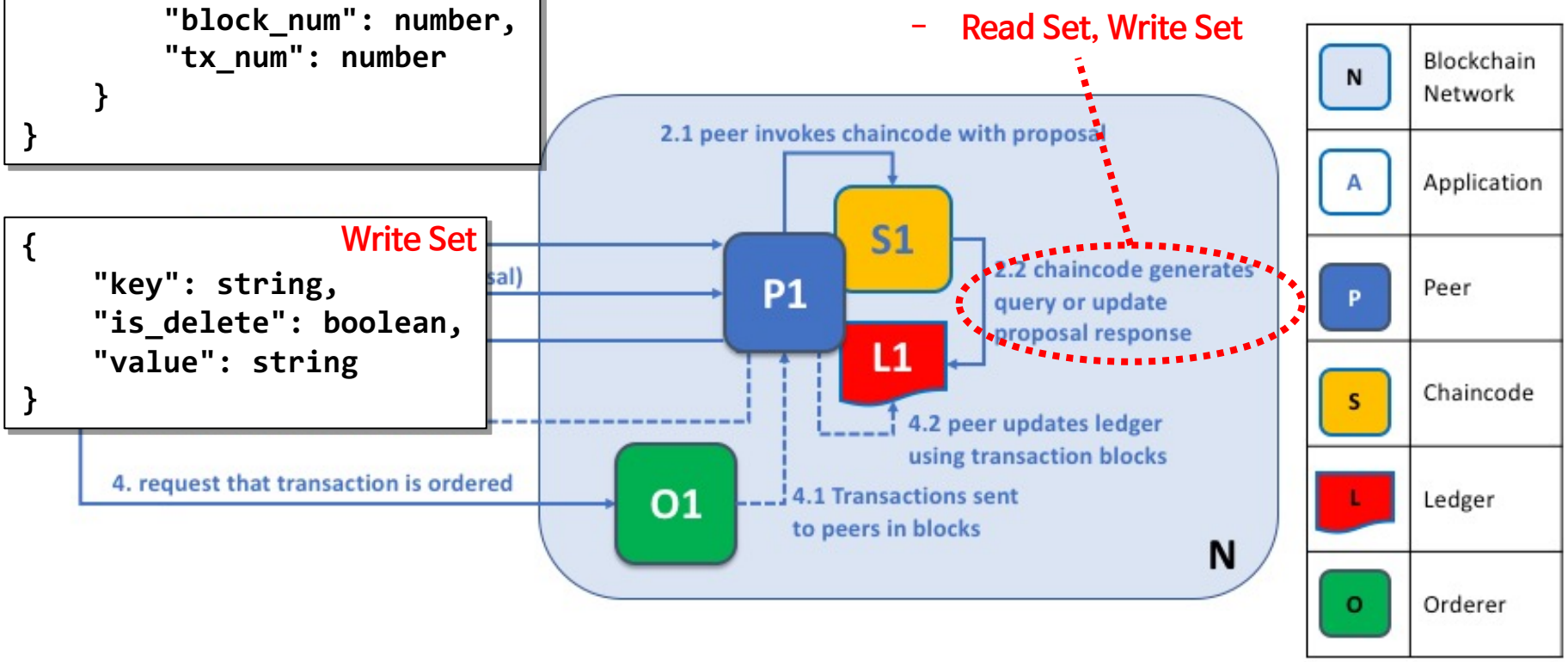
3. 트랜잭션 전송 시나리오

Read Set

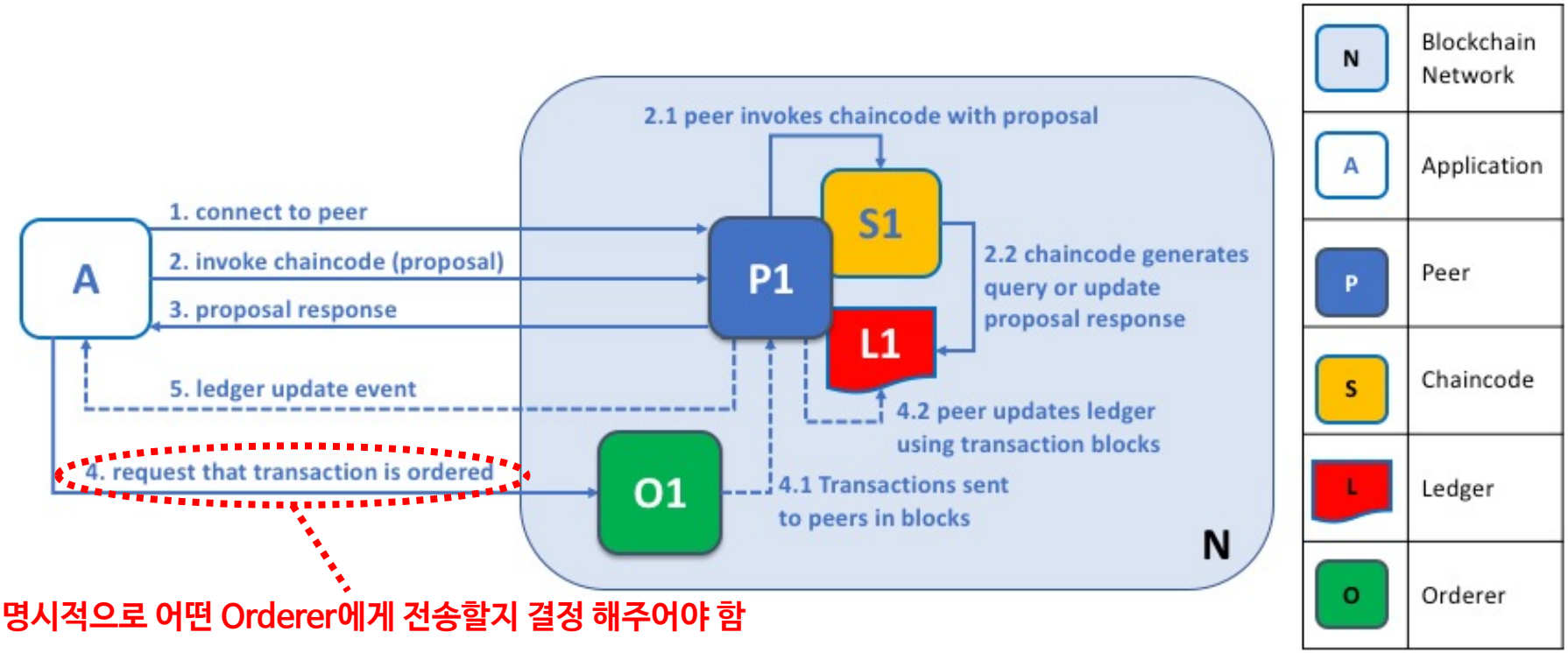
```
{
    "key": string,
    "version": {
        "block_num": number,
        "tx_num": number
    }
}
```

Write Set

```
{
    "key": string,
    "is_delete": boolean,
    "value": string
}
```



3. 트랜잭션 전송 시나리오



- 명시적으로 어떤 Orderer에게 전송할지 결정 해주어야 함

블록체인 기반 소프트웨어 시스템 개발

2022-02-15

빅픽처랩(주)

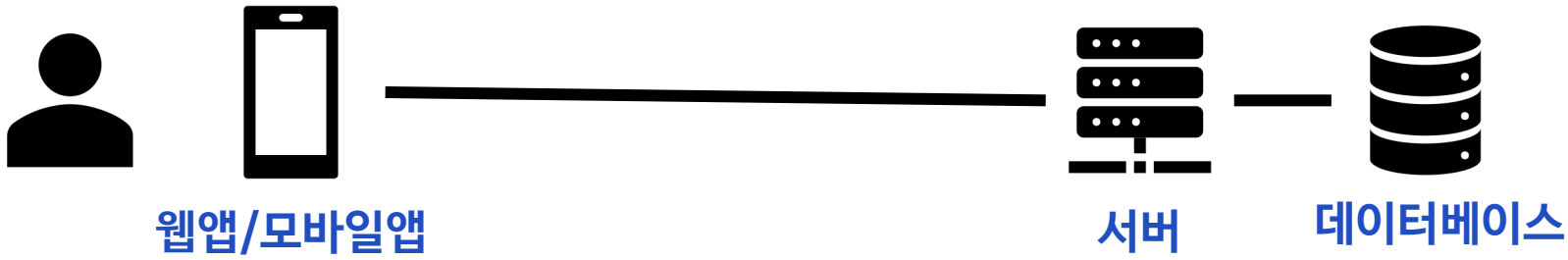
안휘

목차

1. 블록체인 기반 소프트웨어 시스템
2. Client-Server-Blockchain 시스템 개발
3. 블록체인과 일반 데이터베이스 개발의 차이
4. 스마트컨트랙트 설계 원칙

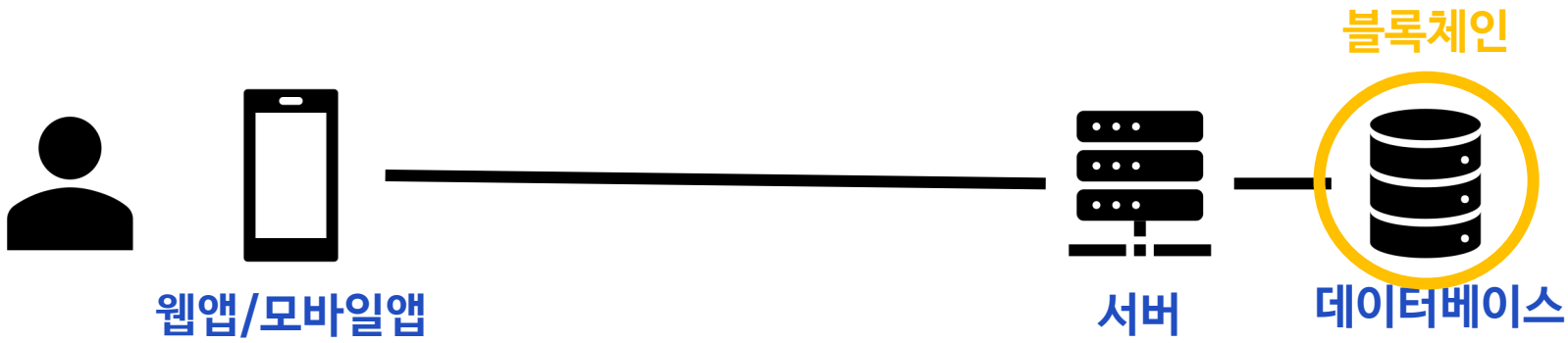
1. 블록체인 기반 소프트웨어 시스템

- 소프트웨어 시스템: 소프트웨어 서비스를 가능케 하는 체계



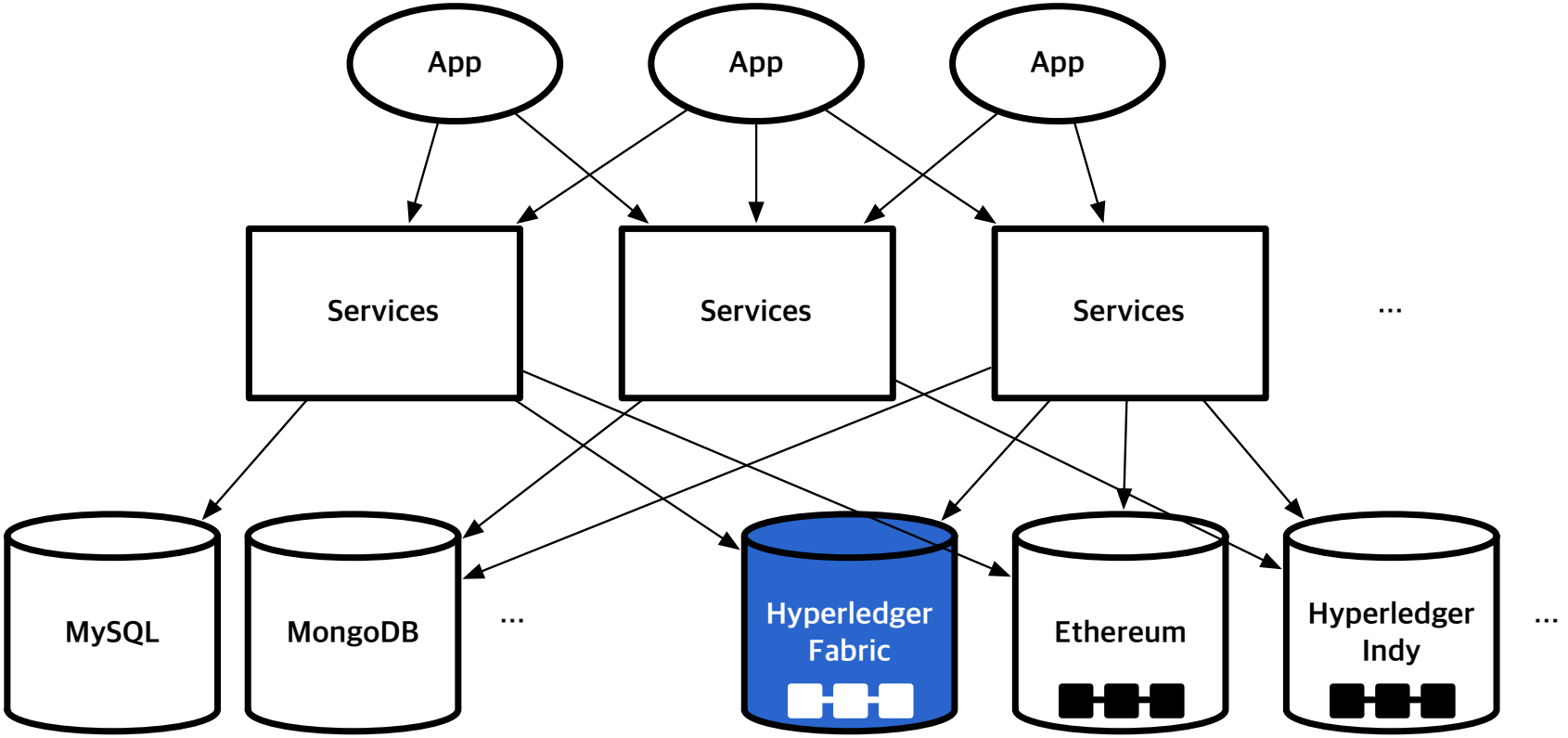
1. 블록체인 기반 소프트웨어 시스템

- 소프트웨어 시스템: 소프트웨어 서비스를 가능케 하는 체계



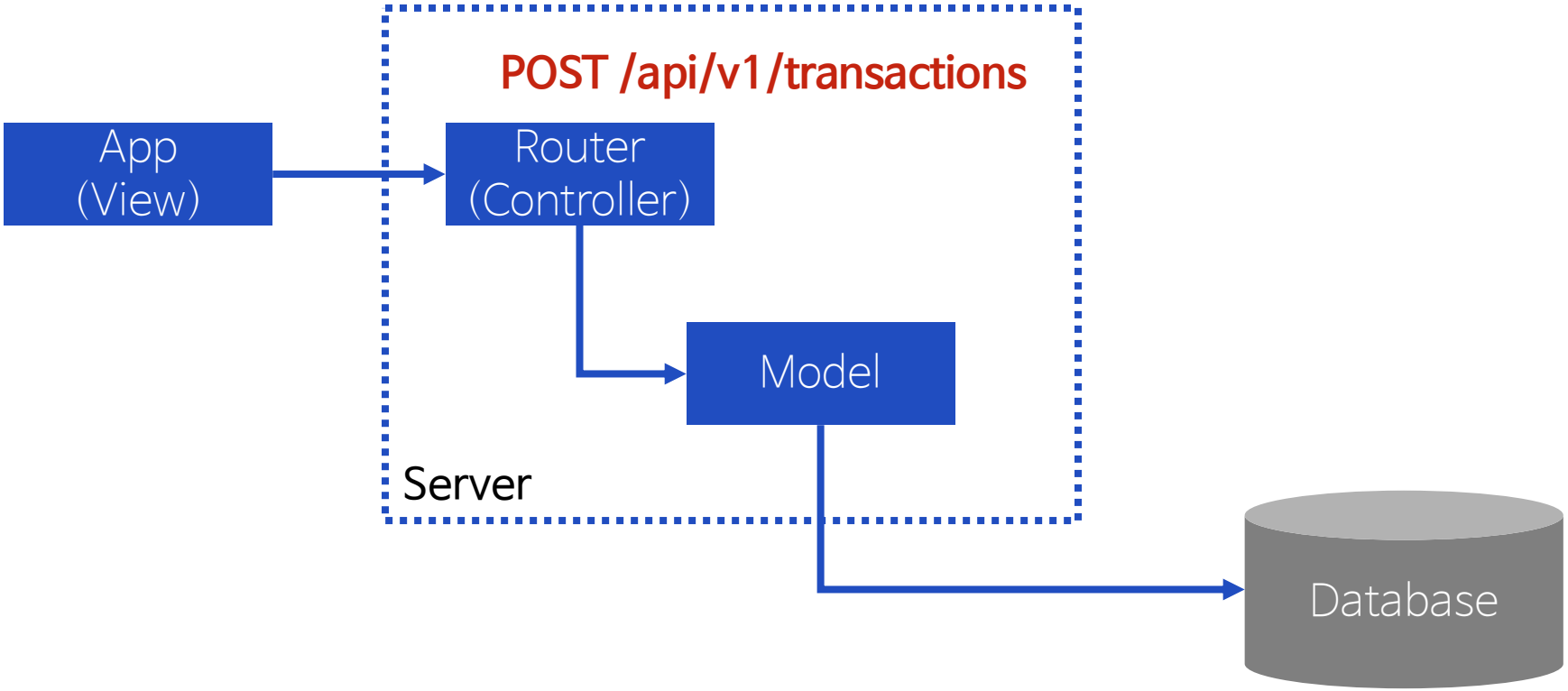
우리가 개발하는 **전체 시스템**에서
데이터를 저장하는 부분 중 하나가
블록체인인 시스템

1. 블록체인 기반 소프트웨어 시스템



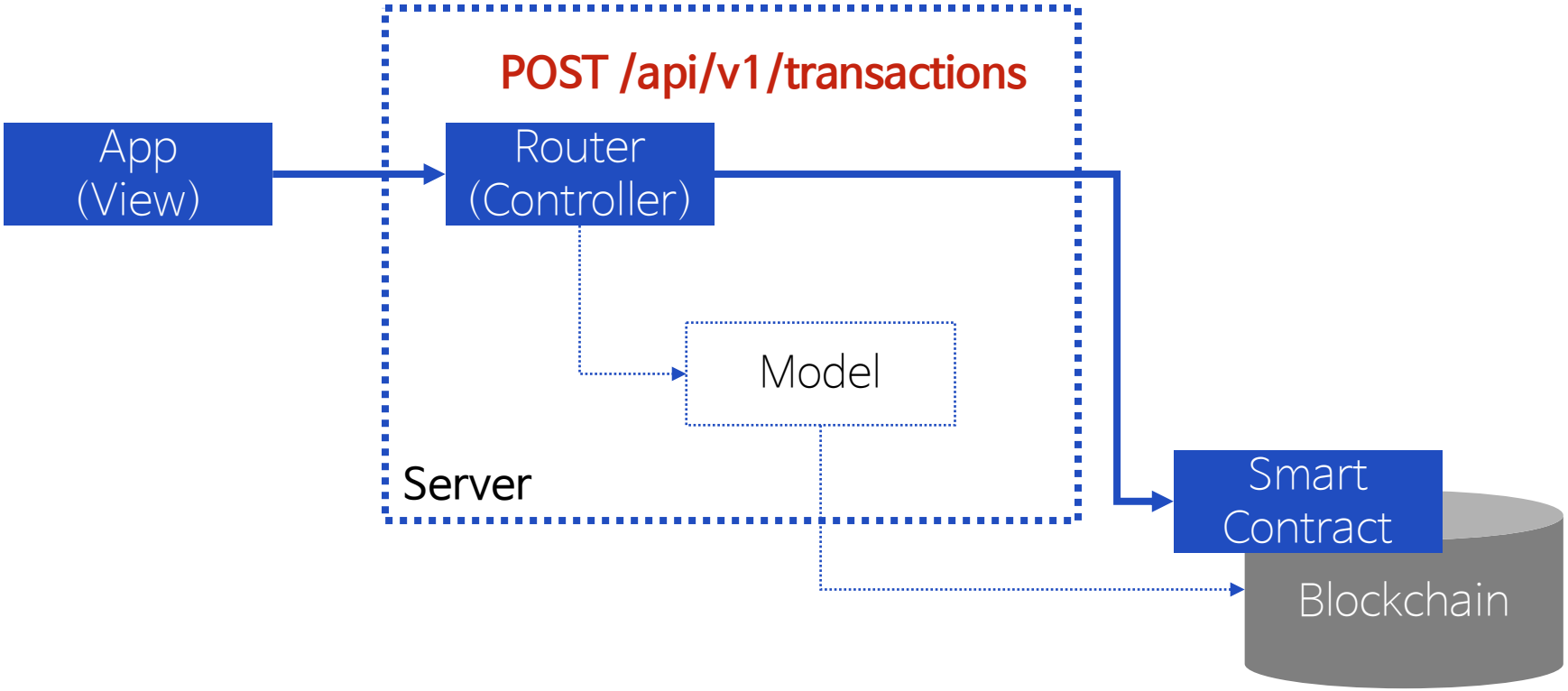
2. Client-Server-Blockchain 시스템 개발

- 일반적인 Client-Server-Database 시스템의 서버 개발



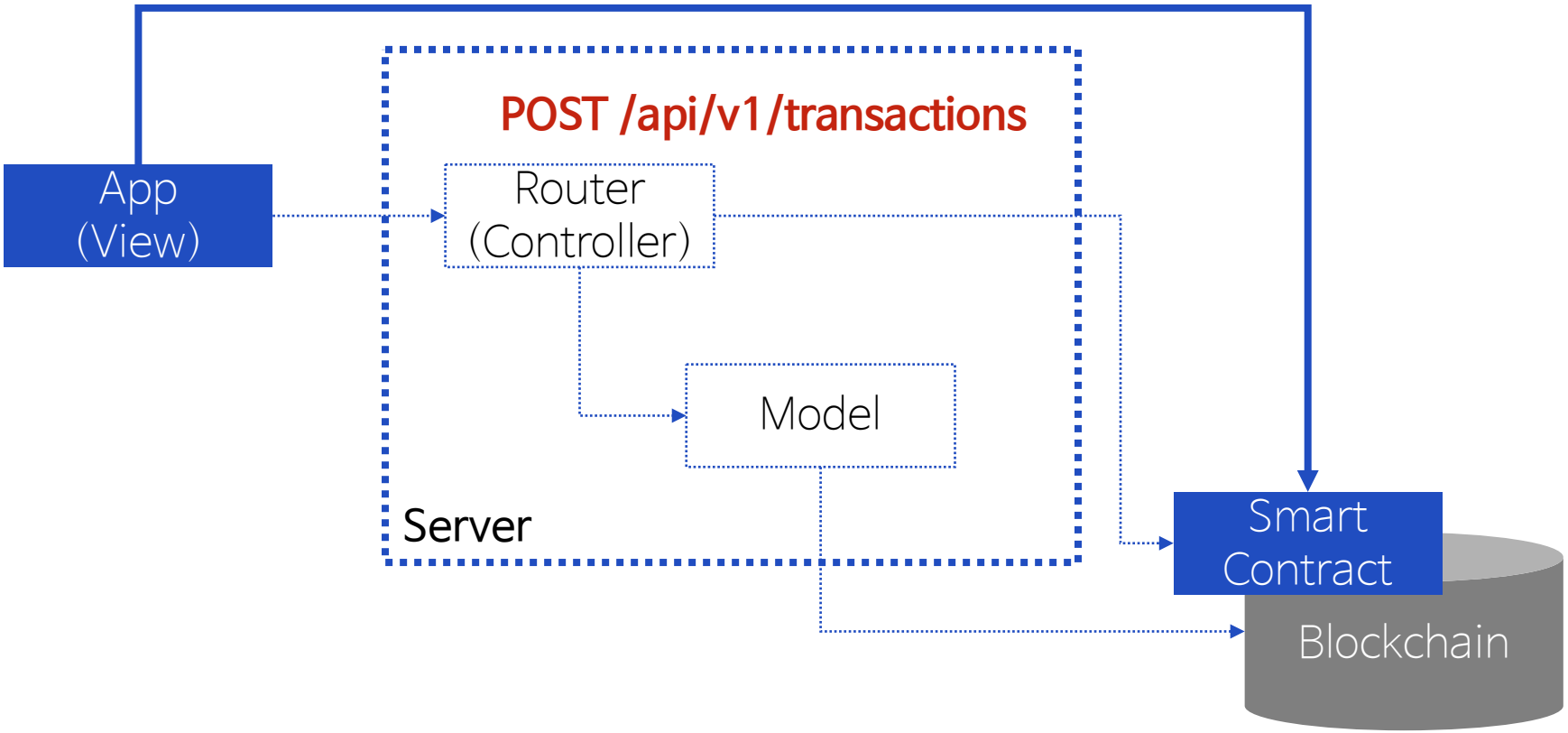
2. Client-Server-Blockchain 시스템 개발

- Client-Server-Blockchain 시스템의 서버 개발 (Hyperledger Fabric)



2. Client-Server-Blockchain 시스템 개발

- Client-~~Server~~-Blockchain 시스템의 서버 개발 (Ethereum)



2. Client-Server-Blockchain 시스템 개발

- **Client-Server-Database 시스템의 서버 개발**
  - REST API를 위한 Router 정의
  - DB 연결 코드 작성
    - DB 전용 SDK 사용
    - db.connect(...) 이런식으로 호출
    - 보통 서버가 실행될 때 1번 호출되어, DB와 연결을 수립
  - Model 코드 작성
    - DB에 저장되는 객체의 모습 정의 (Schema)
    - DB에 저장된 객체를 불러오는 함수들 작성 (a.k.a CRUD 함수)
    - “서버의 로직은 Model에 담긴다”

2. Client-Server-Blockchain 시스템 개발

- **Client-Server-Blockchain 시스템의 서버 개발**
  - REST API를 위한 Router 정의
  - DB 연결 코드 작성
    - DB 전용 SDK 사용 → fabric-sdk, web3.js
    - ~~db.connect(...) 이런식으로 호출~~
    - ~~보통 서버가 실행될 때 1번 호출되어, DB와 연결을 수립~~
    - **연결, 트랜잭션 관리 등을 모두 직접 해주어야 할 가능성이 높음**
  - ~~Model 코드 작성~~ → **스마트컨트랙트** 코드 작성
    - DB(블록체인)에 저장되는 객체의 모습 정의 (Schema)
    - DB(블록체인)에 저장된 객체를 불러오는 함수들 작성 (a.k.a CRUD 함수)
    - "서버의 로직은 ~~Model~~ **스마트컨트랙트**에 담긴다"

2. Client-Server-Blockchain 시스템 개발

- Client-~~Server~~-Blockchain 시스템의 ~~서버~~ 개발
  - ~~REST API를 위한 Router 정의~~
  - DB 연결 코드 작성 → Client-side(e.g. Web App, Mobile App)로 이동
    - DB 전용 SDK 사용 → web3.js
    - ~~db.connect(...) 이런식으로 호출~~
    - ~~보통 서버가 실행될 때 1번 호출되어, DB와 연결을 수립~~
    - **연결, 트랜잭션 관리 등을 모두 직접 해주어야 할 가능성이 높음**
  - ~~Model 코드 작성~~ → **스마트컨트랙트** 코드 작성
    - DB(블록체인)에 저장되는 객체의 모습 정의 (Schema)
    - DB(블록체인)에 저장된 객체를 불러오는 함수들 작성 (a.k.a CRUD 함수)
    - "서버의 로직은 ~~Model~~ **스마트컨트랙트**에 담긴다"

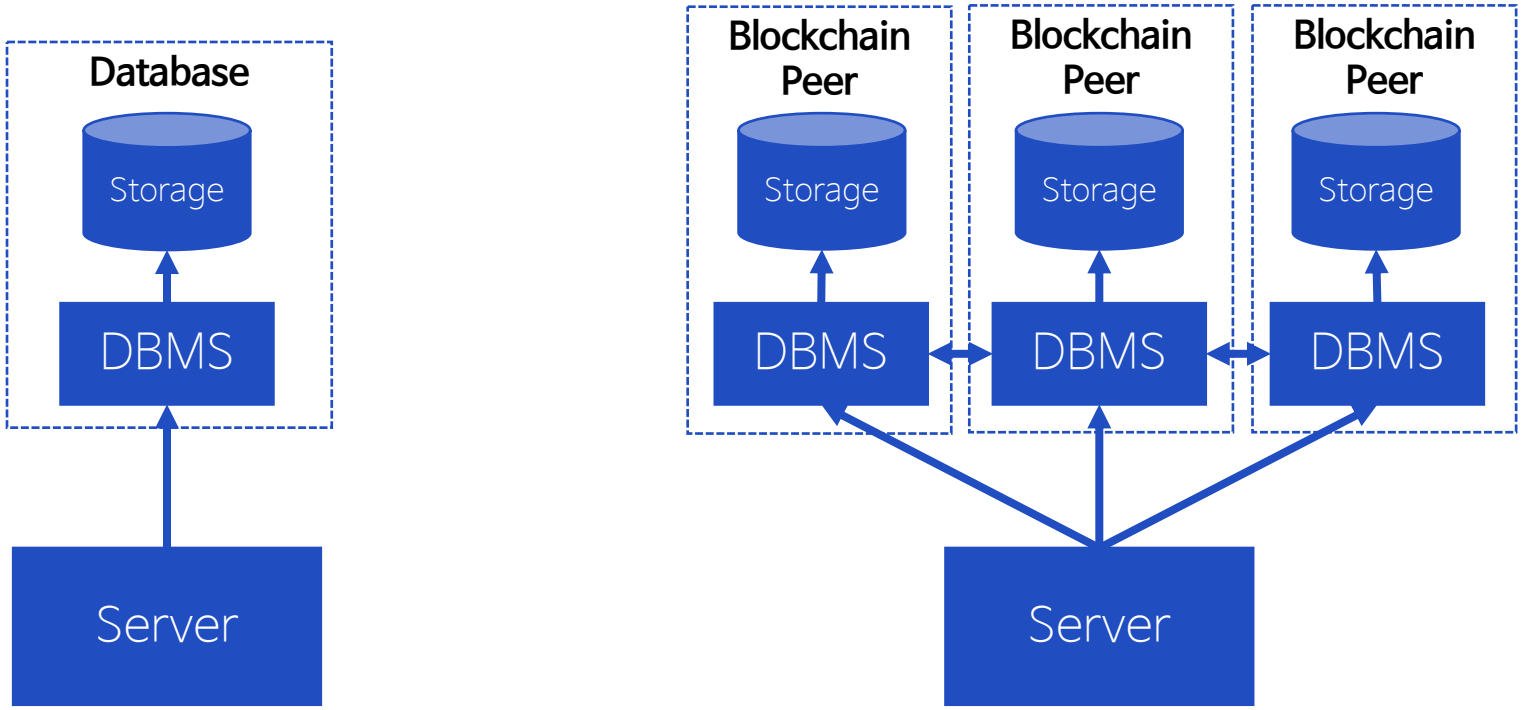
3. 블록체인과 일반 데이터베이스 개발의 차이

- **고수준의 DBMS 부재**
  - DBMS: **D**ata**B**ase **M**anagement **S**ystem
  - DB 자체도 서버: 다중 접속 제어
  - DB 내 데이터에 대한 무결성(데이터 무결성)을 보장함
    - 저장된 데이터들의 일관성을 보장
    - 쏟아지는 트랜잭션들을 잘 정렬&조절하여 데이터 변경의 일관성을 보장
    - 트랜잭션 실패 시 데이터의 원상복구를 보장
    - a.k.a **트랜잭션 관리**
  - db.put(...) 과 같은 high-level 함수 속에 저런 기능들이 모두 보장되고 있음

데이터베이스로 통하는 모든
트랜잭션이 일단 통과해야 하는
중앙화 된 서버 시스템

3. 블록체인과 일반 데이터베이스 개발의 차이

- **고수준의 DBMS 부재**
  - 블록체인에는 **중앙화 된 DBMS가 없음**
  - 트랜잭션 처리 시 **동시성 문제**가 발생할 수 있음
    - 읽는 도중 다른 곳에서는 데이터를 써서 같은 키에 대해 서로 다른 값을 본다던가...



3. 블록체인과 일반 데이터베이스 개발의 차이

- **트랜잭션을 그냥 보내기만 하면 DBMS 가 알아서 데이터 저장을 잘 해줄것이라는 믿음을 버려야 함**
- **MVCC Read Conflict**
  - 먼저 보낸 트랜잭션이 미처 반영되기 전에 같은 키에 대해 또다른 트랜잭션을 보냄
- **Phantom Read Conflict**
  - DB에서 Range 검색 시, 처리 도중 다른 트랜잭션의 삽입으로 처음 읽을 때 없던 값들이 추가되거나 있던 값들이 삭제됨

3. 블록체인과 일반 데이터베이스 개발의 차이

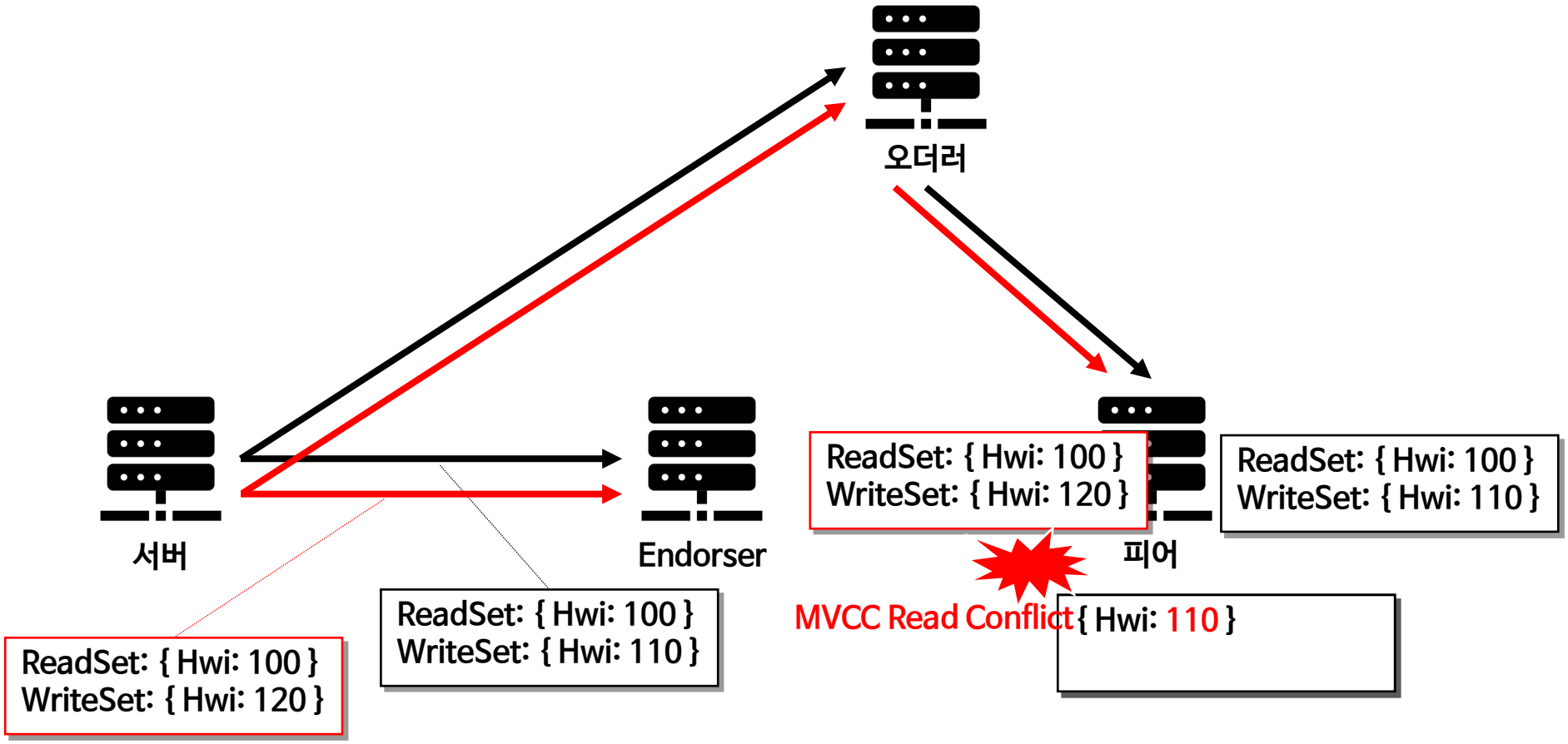
- MVCC: Multi-Version Concurrency Control
  - DBMS에서 **동시성 문제를 해결**하기 위해 나온 기법
  - 가장 기본적인 방법: Lock을 이용
    - 쓰기가 수행되는 동안 Lock을 걸어 다른 요청들을 거부
    - 성능 저하가 심할 수 있음

  - MVCC
    - 저장되는 값에 대해 Version을 부여
    - 읽기를 수행할 때는, 최신 스냅샷으로부터 읽음
    - 쓸 때는 기존 값을 덮어씌우는 대신, 변경 내역을 기반으로 새로운 버전 생성

  - 하이퍼레저 패브릭의 경우, 분산 환경이기 때문에 Lock을 거는 것이 불가능하고, 이 때문에 MVCC 방식으로 State DB를 관리함

3. 블록체인과 일반 데이터베이스 개발의 차이

- MVCC 예



3. 블록체인과 일반 데이터베이스 개발의 차이

- MVCC 해결책
  - 블록 생성 속도 조절
  - 최대한 중복 키 생성을 피하는 설계
  - 요청들을 큐에 넣어 관리
  - 후처리 방식
  - 배칭(Batching)

스마트 컨트랙트 개발

2022-02-15

빅픽처랩(주)

안휘

목차

1. 하이퍼레저 패브릭의 스마트 컨트랙트: 체인코드

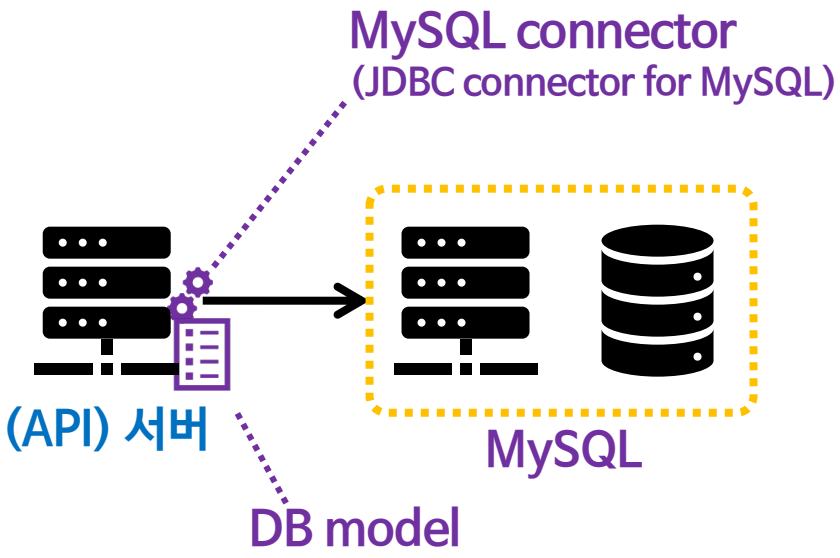
- **스마트 컨트랙트란**
  - DB 입출력 함수들
  - “블록체인”이라는 데이터베이스에 대한 CRUD 프로그램
  - MVC 패턴의 Model 영역에 해당

  - 이더리움
    - 이더리움의 스마트 컨트랙트는 그 자체가 일종의 서비스 서버
    - 이더리움이 인터넷에 준하는 거대한 공용 데이터베이스이기 때문

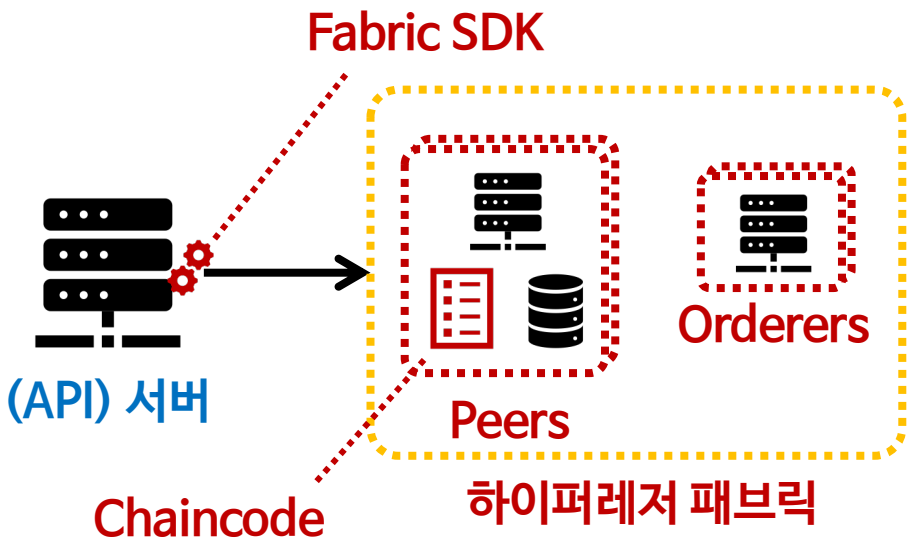
  - 그렇다면, 프라이빗 블록체인인 하이퍼레저 패브릭에서의 스마트 컨트랙트란?

1. 하이퍼레저 패브릭의 스마트 컨트랙트: 체인코드

일반적인 Server – DB 구조

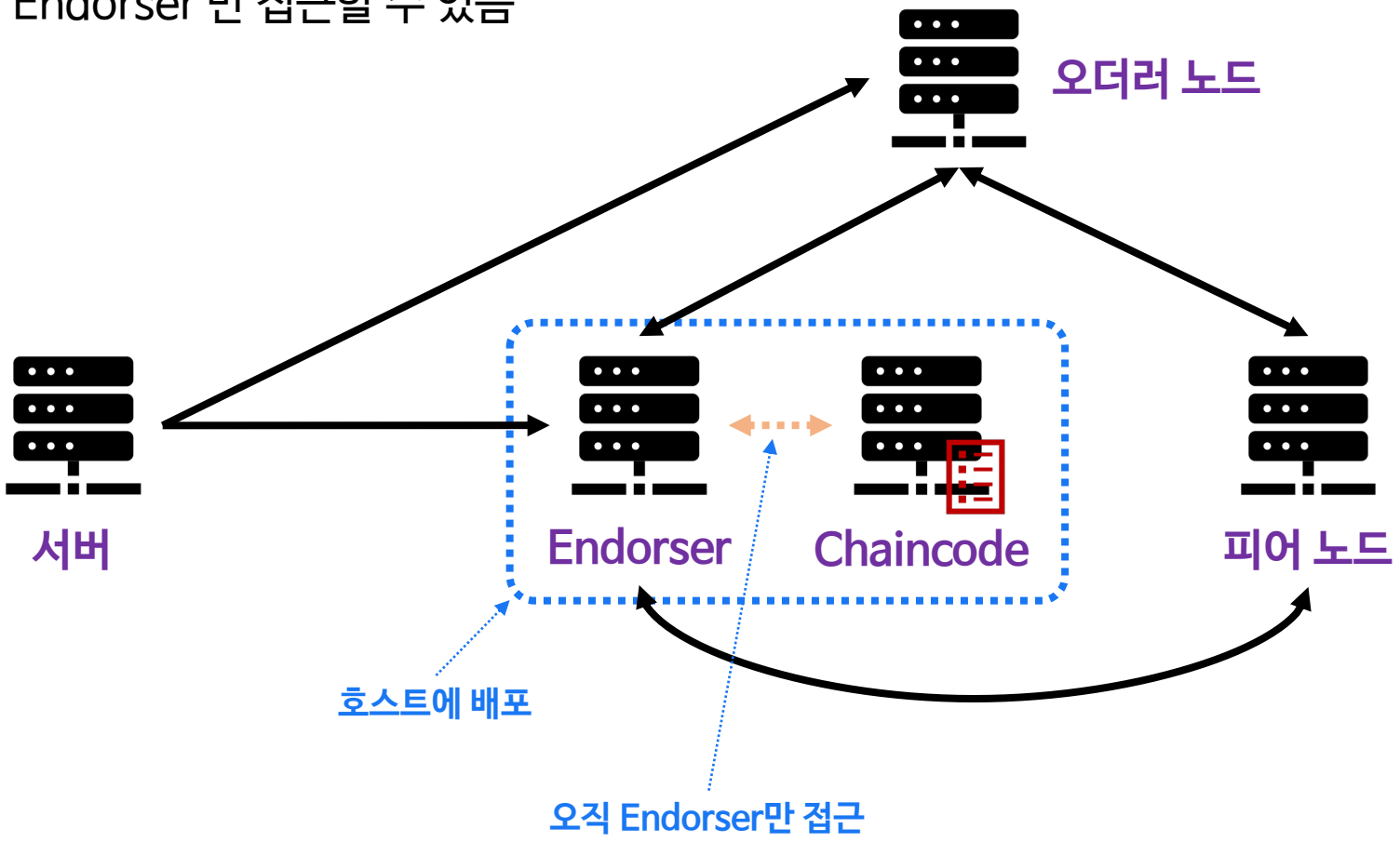


하이퍼레저 패브릭 Server – DB 구조



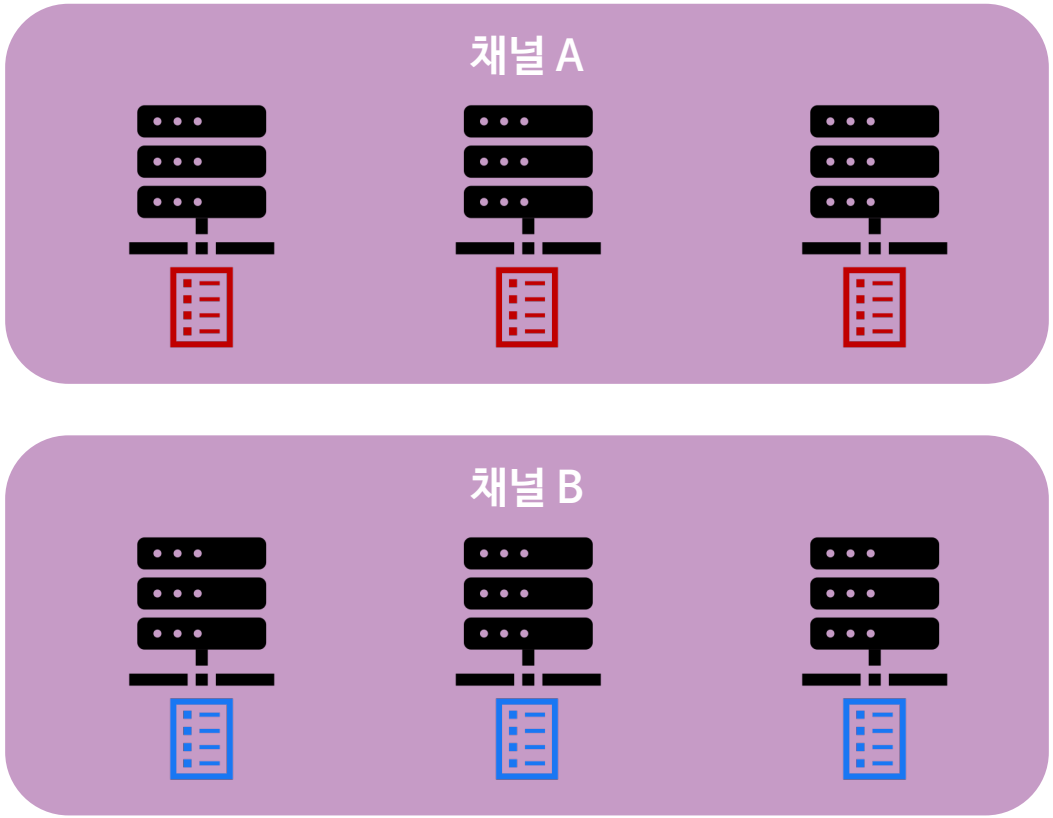
1. 하이퍼레저 패브릭의 스마트 컨트랙트: 체인코드

- 체인코드 특징
  - Endorser 만 접근할 수 있음



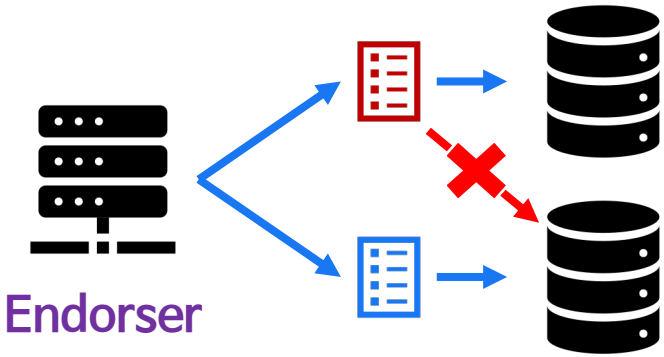
1. 하이퍼레저 패브릭의 스마트 컨트랙트: 체인코드

- **체인코드 특징**
  - 체인코드는 채널 별로 설치됨



1. 하이퍼레저 패브릭의 스마트 컨트랙트: 체인코드

- **체인코드 특징**
  - 체인코드는 고유한 데이터 콜렉션을 갖음
    - 같은 채널이어도 다른 체인코드는 해당 데이터에 접근할 수 없음
  - 물리적으로는 한 개의 LevelDB에 저장되지만, 논리적(테이블)으로는 구분되어 다른 체인코드의 접근을 허용하지 않음



  - 예를 들어, User 체인코드와 Car 체인코드가 있다고 할 때:
    - Car 체인코드 함수가 User 체인코드가 관리하는 사용자 정보를 읽어올 수 없음

1. 하이퍼레저 패브릭의 스마트 컨트랙트: 체인코드

- **체인코드 종류**
  - 사용자 체인코드
    - 여러분이 개발하고, 배포, 운영하는 체인코드
  - 시스템 체인코드
    - 하이퍼레저 패브릭에서 필요한 여러 기능들도 체인코드로 구현되어 있음
    - 이들은 피어 노드에 기본적으로 활성화되어 있음
    - **_lifecycle**: v2.0 이후 체인코드 생명주기를 관리
    - **Configuration system chaincode (CSCC)**: 채널 설정 변경을 관리
      - BatchTimeout, 인증서 업데이트 등
    - **Query system chaincode (QSCC)**: State DB에 대한 입출력 함수를 실행
      - 각종 메트릭 정보도 가져올 수 있음 (채널 숫자, 피어 상태 등)
    - **Endorsement system chaincode (ESCC)**: Endosring 단계 수행
    - **Validation system chaincode (VSCC)**: 최종 단계에서 트랜잭션 검증 수행

2. 체인코드 개발

- **하이퍼레저 패브릭 SDK**
  - Go, JavaScript, Java
    - 모든 언어들이 동일한 개념 모델을 공유함: Context, Gateway, Transaction 등
    - 한 언어를 익히면, 나머지도 동일한 방법으로 사용할 수 있음

  - 두 종류: 체인코드 SDK, 클라이언트 서버 SDK
    - 체인코드 SDK
      - 하이퍼레저 패브릭의 State DB에 대한 입출력(CRUD) 함수를 만들기 위한 SDK
      - GetState, PutState, DelState, …
      - Context 객체를 통해 QSCC의 함수들을 stub 형태로 제공

    - 클라이언트 서버 SDK
      - sendTransaction, evaluateTransaction
      - 연결 유지를 위한 함수들 (DB로 치면, connect 함수 같은 것들)

2. 체인코드 개발

- **체인코드 개발 유의사항**
  - Query를 철저하게 고려한 Key 구조 설계가 필요
    - 저장된 데이터의 속성값으로 검색이 안됨
      (예: 나이가 20 이상인 사용자들 검색, 이메일이 "…"인 사용자 검색 등)
    - Query 에서 요구하는 속성값들을 모두 Key에 넣어두어야 함
    - LevelDB는 정렬도 안됨 (Key 값을 알파벳 순으로 정렬 해놓기만 함)
      - 인덱스 같은건 꿈도 꾸지 말지어다…
  - Random 또는 Endorser마다 달라질 수 있는 값을 생성하는 코드는 절대 삽입 금지
    - ex) time.Now().Unix()

2. 체인코드 개발

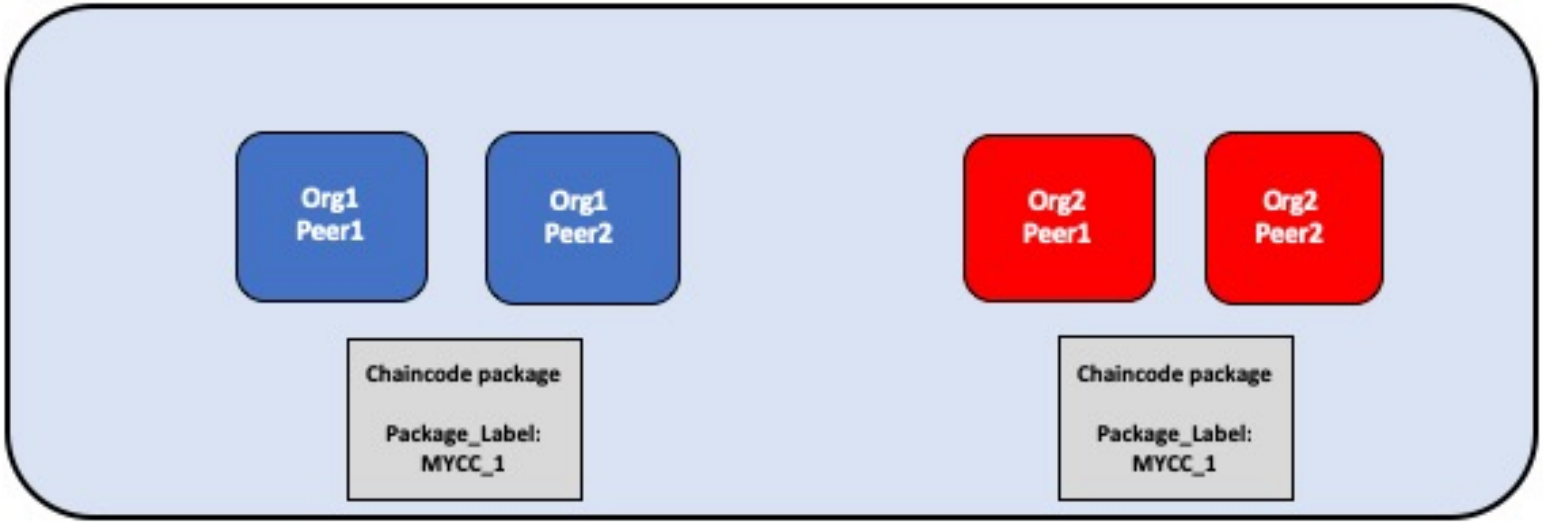
- **사실 체인코드는 별것이 다 됩니다.**
  - 외부로 HTTP 콜도 보낼 수 있음
  - 다른 체인코드도 호출 할 수 있음 (근데 Query만 가능)

  - …사실 Go로 가능한건 대부분 됩니다

하지만 하지 맙시다. 커밋단계에서 깨질 가능성이 너무 랜덤하게 높아집니다.

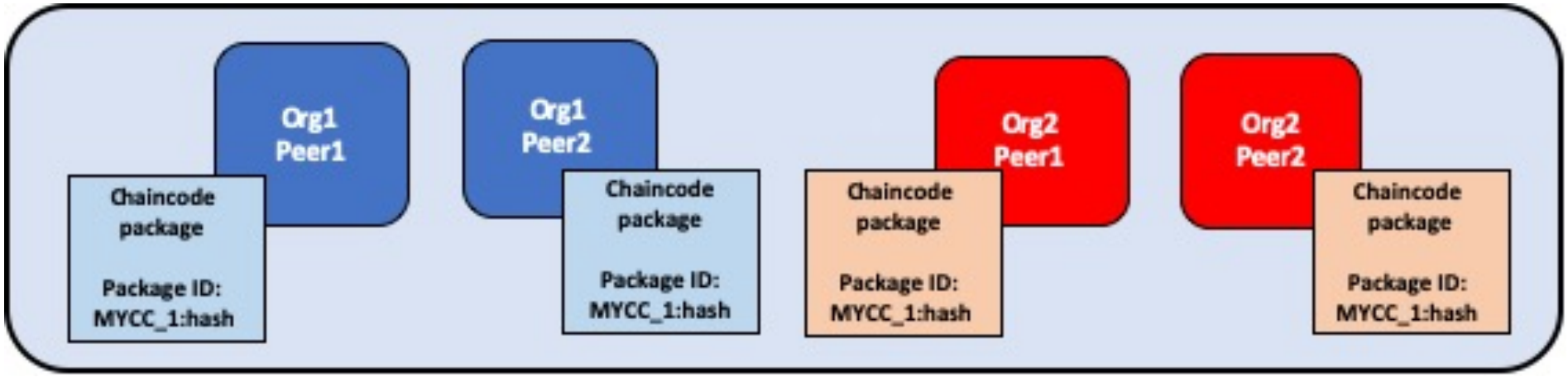
3. 체인코드 운영

- **체인코드 배포**
  - 체인코드 패키지 생성
    - 체인코드 소스코드를 .tar.gz 형태로 압축
    - 하이퍼레저 패브릭에서 제공한 CLI 프로그램을 이용



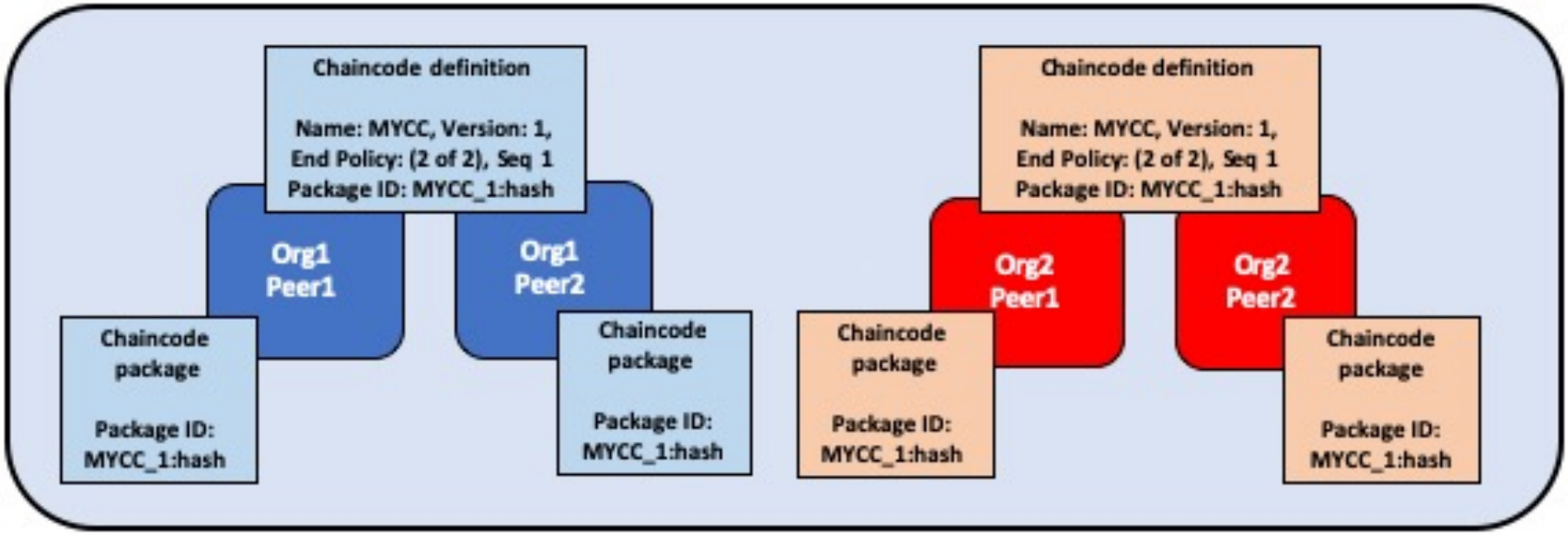
3. 체인코드 운영

- **체인코드 배포**
  - 체인코드 설치
    - 각 피어에 직접 설치 (peer CLI 이용)



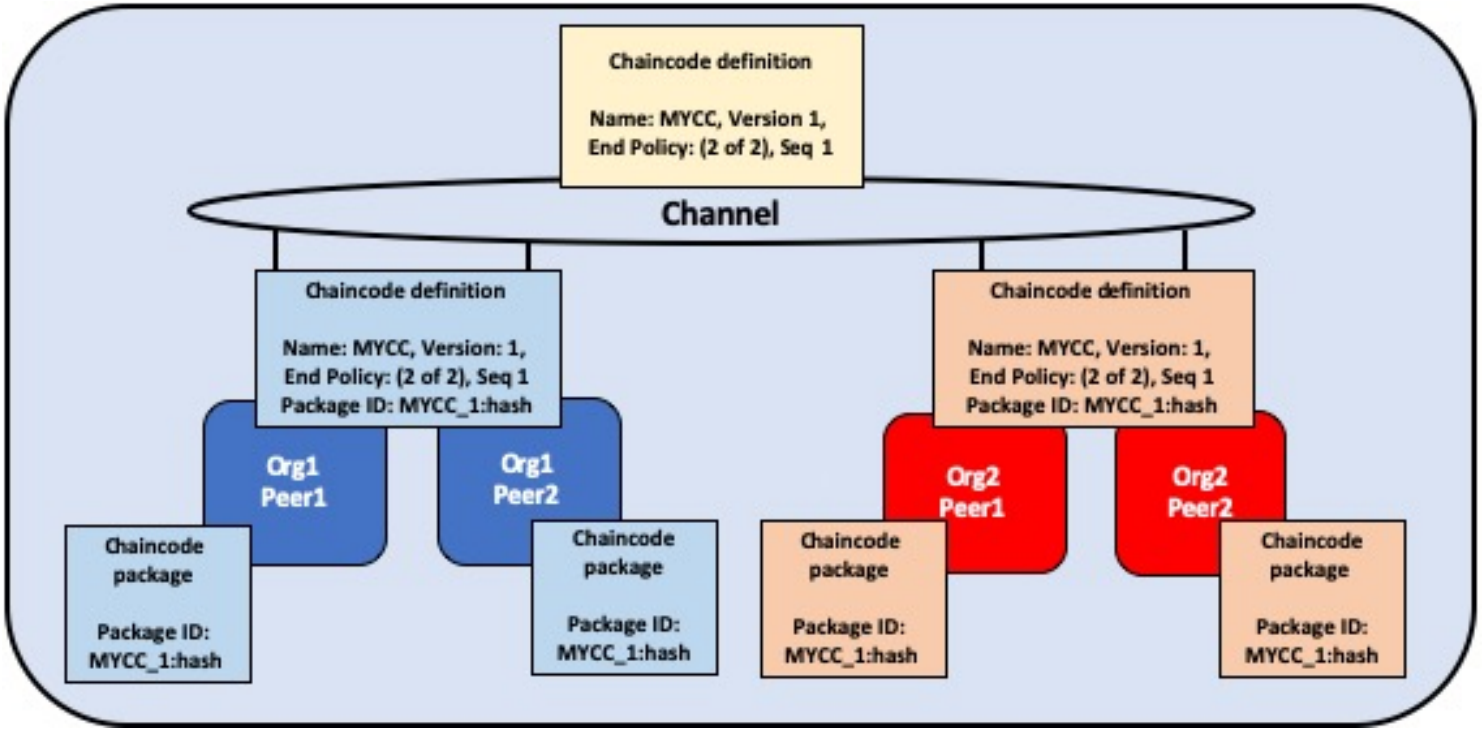
3. 체인코드 운영

- **체인코드 배포**
  - 체인코드 승인(Approve)
    - 각 조직 별로 수행
    - Name, Version, Sequence, Endorsement Policy, …



3. 체인코드 운영

- **체인코드 배포**
  - 체인코드 커밋(Commit)
    - 채널에서 한번 수행하면 됨
    - 체인코드 도커 컨테이너가 실행되며, 가동이 준비됨
    - 기존 체인코드 컨테이너는 중지됨 (예전엔 수동으로 꺼줘야 했음…)



4. 스마트컨트랙트 설계 원칙

- **일반적인 “Client-Server-Database” 시스템의 DB모델 설계 원칙과 같음**
  - ER 다이어그램으로 DB 구조 설계 잘하고,
  - 서버 보안 잘 챙기고,
  - 앱은 UI/UX 잘 만들고, … 등등

그러나…

4. 스마트컨트랙트 설계 원칙

- 블록체인은 다른 데이터베이스에 비해 ...

1) 매우 느리고,

2) 용량을 많이 차지하고,

3) 고수준의 DBMS가 없습니다.

=〉 그럼에도 여기에
저장해야하는 것은 무엇인가

4. 스마트컨트랙트 설계 원칙

- **무엇을 저장할지 결정**
  - **저장할 만한 데이터** (AND 조건)
    - 위변조를 반드시 막아야 하는 데이터
    - 참여자들이 모두 함께 공유해야 하는 데이터
    - 예: 투표 결과, 돈 거래 내역, 물건의 운송 이력, 이해관계자들 사이의 중요 문서(계약서 등)

  - **저장하면 안되는 데이터**(OR 조건)
    - 실시간성을 요구하는 데이터는 안됨
    - 복잡한 Query가 중요한 데이터 (End-User의 앱 서비스에 직접 연관되는 데이터)
      - JOIN 등을 요구하는 복잡한 구조의 데이터
    - 공유되어서는 안되는 개인 데이터
    - 예: 주민등록번호 같은 개인정보, 로그

4. 스마트컨트랙트 설계 원칙

- **NoSQL의 DB 설계 원칙과 동일**
  - 하이퍼레저 패브릭과 이더리움 모두 기본적으로 NoSQL 데이터베이스
    - 둘다 LevelDB를 저장소로 사용함
      - 구글에서 만든 빠르고 가볍지만 불편한 DB

  - Key-Value 데이터베이스인데, Key 구조를 잘 설계하는 것이 중요
    - Query 에 맞춰 Key를 설계
    - LevelDB는 FindByAttributes 이런 거 안됨
    - Index 도 없음 → 대신 Key 를 알파벳 순서로 정렬은 해줌

4. 스마트컨트랙트 설계 원칙

- **블록체인은 언제나 서브 DB**
  - 블록체인을 서비스를 위한 메인 DB로 쓰지 마라
    - 사용자 앱의 첫 페이지를 띄우기 위한 데이터를 fetching 해오는 DB로 쓰면…

  - 블록체인은 올라운드 플레이어가 아니라, **특정 영역에서 아무도 못하는 일을 해주는 스페셜 플레이어**임
  - 사용자가 블록체인이 뒤에서 동작하는 것을 모르는 것이 최고의 블록체인 기반 소프트웨어 서비스