Session I : Klaytn Architecture

이병철 / Ground X

Klaytn's Network Architecture and Consensus

이병철, **Andy Lee**

Software Engineer, Platform & SDK Team, Ground X
- Consensus & Governance
- Klaytn Main-net (Cypress)

CTO of Spes.io, R&D division of Emurgo, Japan
- Dapp development on Ethereum

Senior Software Engineer, Samsung Electronics
- Developed various pilot projects
- Home media sharing, Search engine, Etc.

TABLE OF CONTENTS

Network Architecture

- Former decentralized network topology
- Layered architecture of Klaytn

Network Topology

기존 Decentralized network

장점

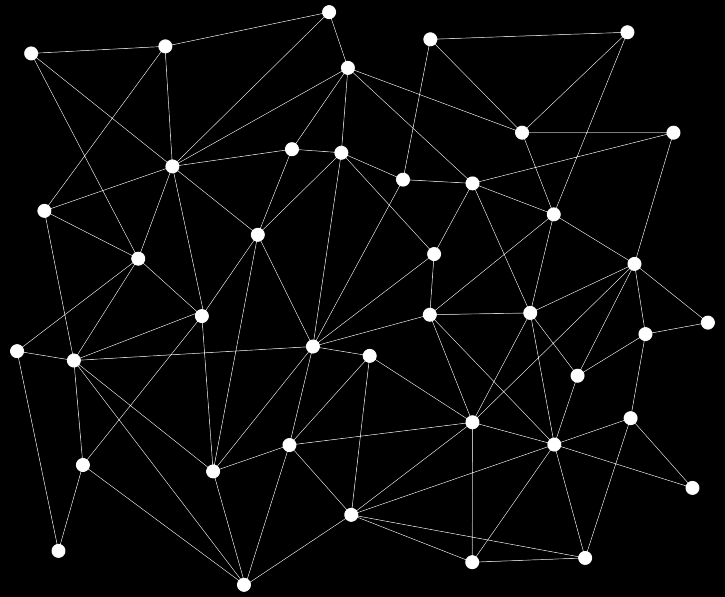
- 강력한 검열 저항성
- 누구나 블록 생성에 참여할 수 있음
- 공격 대상 특정이 어려움

단점

- 낮은 **Throughput**
- 불확실한 **Latency**

의문

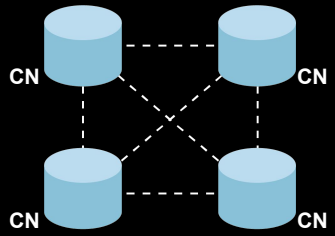
- 어떻게 일정 수준의 성능을 안정적으로 확보할 수 있을까?
- 모든 노드는 과연 동등한가?



Three Layers of Nodes

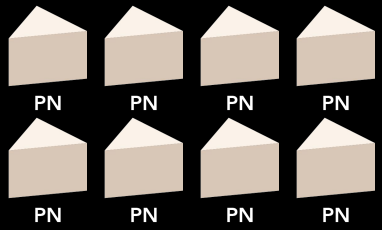
Consensus Node

- 높은 수준의 하드웨어 / 네트워크 요구사양
- 블록 생성을 전담



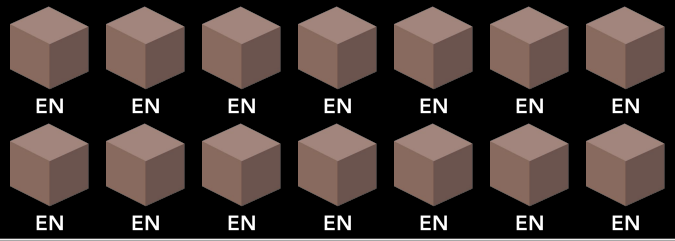
Proxy Node

- Endpoint Node의 트랜잭션을 CN에 전송
- CN이 만든 블록을 Endpoint 노드로 전송



Endpoint Node

- 사용자/서비스를 위한 API 제공
- 사용자/서비스의 트랜잭션을 PN에 전달하고 결과를 수신



Layered Architecture - 3 Layers

Consensus Node Network (CNN)

Consensus Node (CN)

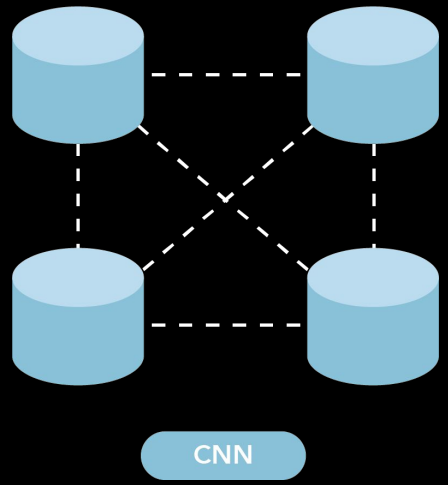
- 컨센서스를 통한 블록 생성을 전담
- 신뢰할만한 노드들만 참여 가능

장점

- 상대적으로 통제되는 환경
  → 일정 수준의 성능 보장

단점

- 누구나 블록 생성에 참여할 수 없음
  → 담합 우려
- 공격 대상의 축소
  → 전통적인 네트워크 보안 필요

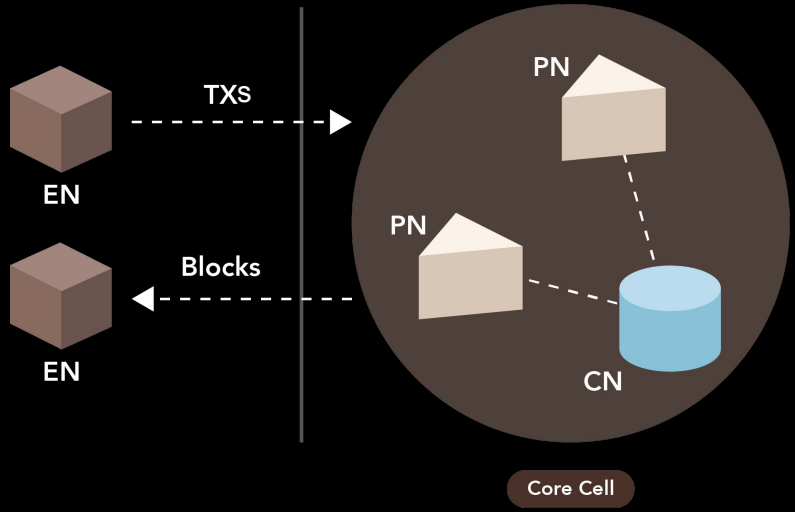


Layered Architecture - 3 Layers

Proxy Node Network (PNN)

Proxy Node (PN):

- 외부 **EN**으로부터 받은 트랜잭션을 **CN** 및 인근 **PN**에 전달하고, **CN**에서 생성된 블록을 **EN**에게 전파
- 일반적으로 **1 CN**은 **2 PN**을 가지며, 이를 묶어 코어셀 **(Core Cell)**로 부름
- **CN**의 부하를 줄이고 **CN**이 외부에 직접 노출되지 않도록 보호

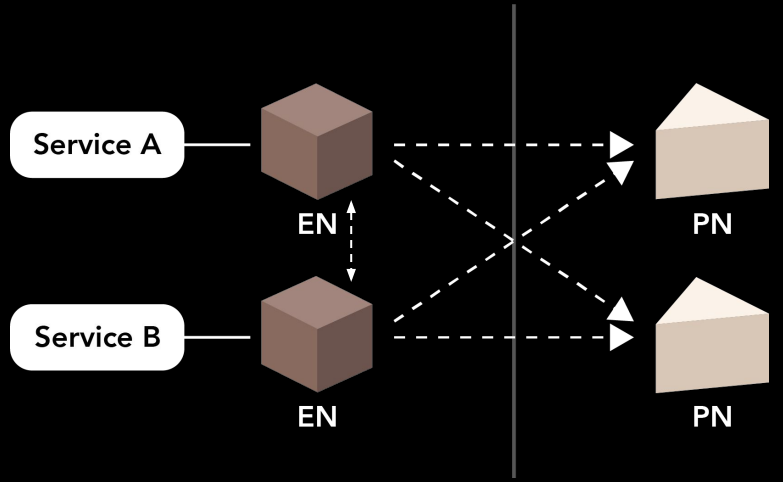


Layered Architecture - 3 Layers

Endpoint Node Network (ENN)

Endpoint Node(EN)

- 서비스/사용자의 트랜잭션을 **PN**들로 보내고, 결과를 받아 사용자에게 제공
- 다양한 **RPC/WS API**를 최종 사용자에게 제공
- 네트워크 부하를 줄이기 위해 **EN** 노드간 **P2P** 전송

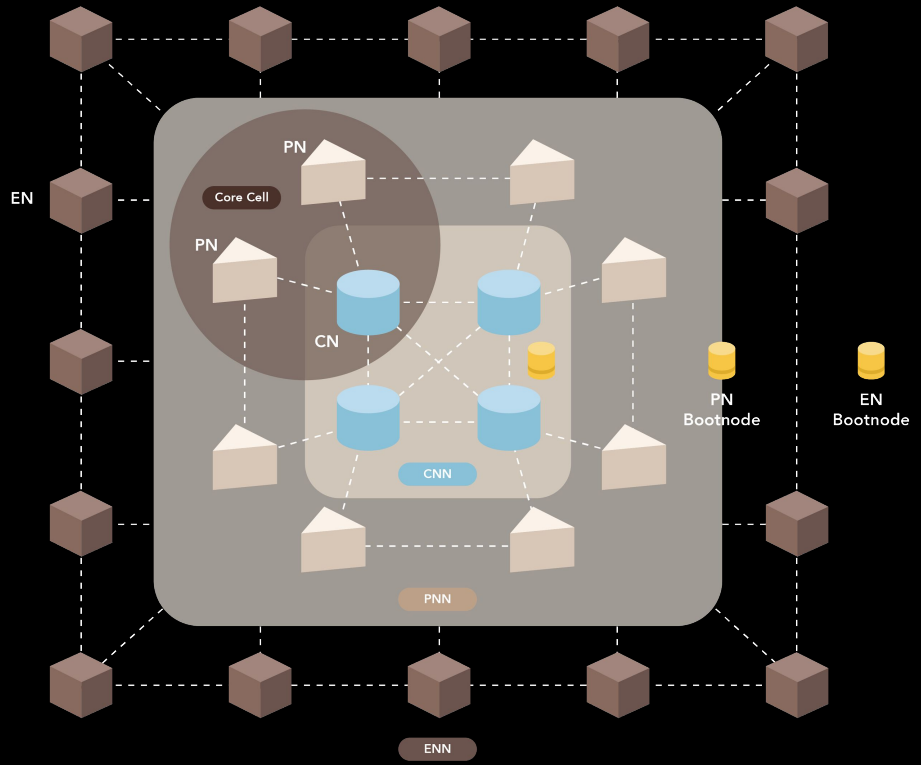


Layered Architecture

Permissioned + Permissionless

Bootnode (BN):

- 현재 네트워크에서 동작중인 노드들의 정보를 가지고 있고, 요청에 따라 연결해야 할 노드 정보를 제공
- EN과 PN은 개별적인 부트노드에 연결 됨
- EN은 주변 EN 및 2개의 PN에 연결됨
- PN은 속해있는 Core Cell의 CN과 다른 Core Cell의 PN에게 연결됨
- Cypress에는 PN과 EN을 위해 각 3개씩의 BN이 존재



Summary - Network Architecture

Permissioned

블록 생성을 장애 없이 빠르게
다양한 주체들로 구성
Enterprise 수준의 네트워크 보호 적용

Throughput 과 Latency 확보

Permissionless

EN을 통한 사용자 / BApp 지원
자유로운 EN 구축 및 연결 가능
전통적인 P2P Network 적용

자유로운 사용성 및 투명성 확보

Consensus Algorithm

- BFT, Council and Committee
- Proposer & Committee selection
- Reward system
- Roundchange
- Future work

Klaytn의 Consensus Algorithm

기존 PoW, PoS의 문제

일정치 않은 블록 생성 시간
Fork 발생 및 그로 인한 Reorg
확률적인 Finality

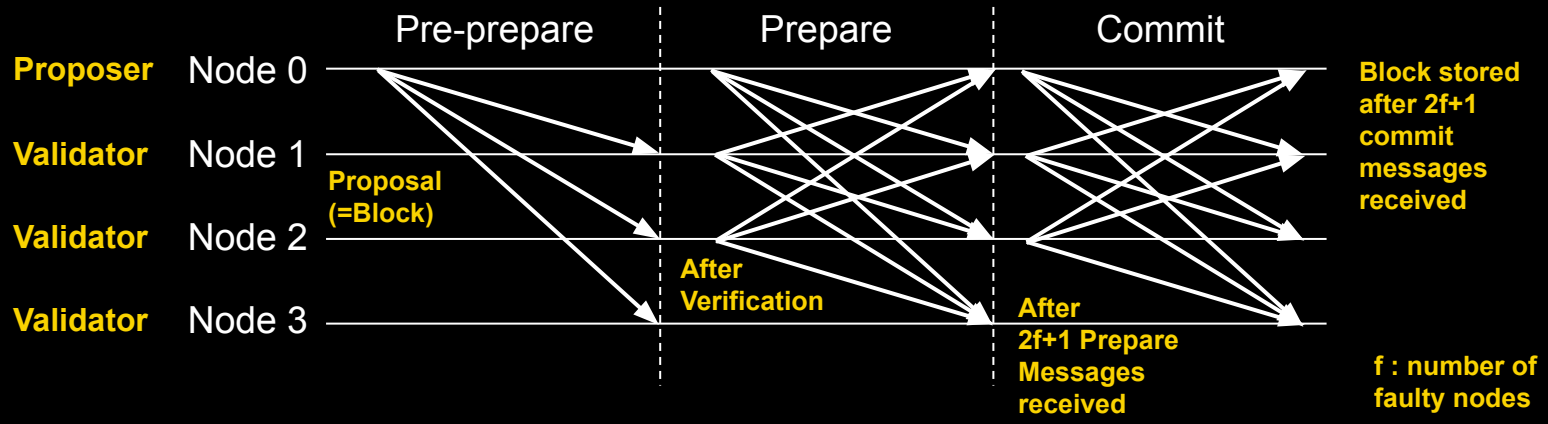
PBFT 적용

Klaytn의 컨센서스

빠른 블록 생성 시간
즉각적인 Finality
Energy Efficiency

BFT? (Byzantine Fault Tolerant)

- **Faulty** 노드는 전체의 ⅓ 미만이어야 함
- 상태마다 많은 메시지 상호 전송
- 합의가 이루어질 경우, 즉각적인 **Finality** 확보



Council and Committee

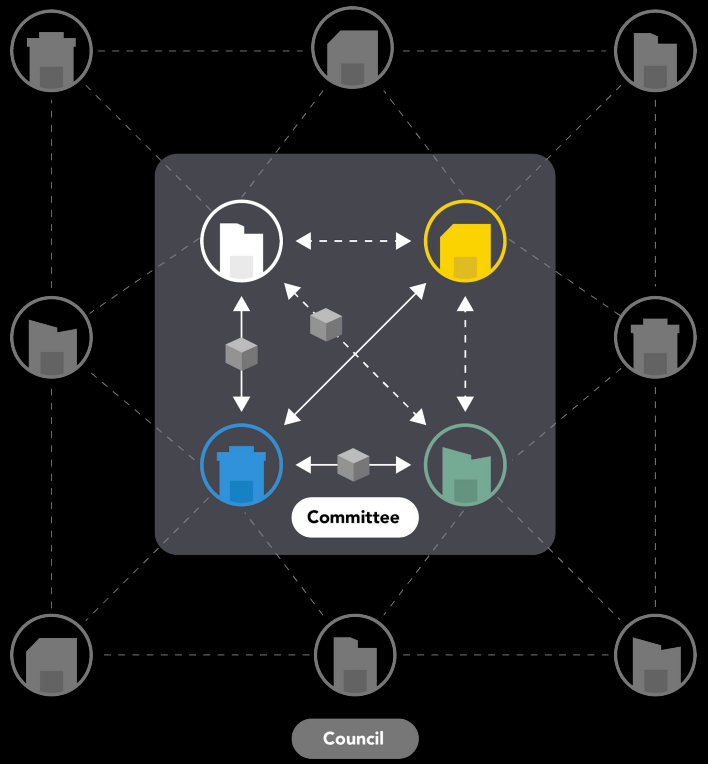
BFT에서 노드수 증가 → 담합 가능성 ↓
BFT에서 노드수 증가 → 메시지 수 ↑

노드 수를 늘려도 메시지 수를 제한할 수 있는 방안 필요

Committee 개념의 도입

- **Council** : 모든 **CN** 노드의 집합
- **Committee:** 한 블록을 검증하기 위해 랜덤하게 선택된 **CN** 노드들

랜덤하게 선택된 소수의 노드가 검증함으로써, 컨센서스 메시지를 줄임
(현재 **Klaytn**의 **Committee** 크기: **22**)



Staking of CNs

Staking

모든 **CN** 노드는 일정 수량 이상의 **KLAY**를 예치하고 유지해야 함

- 이는 **CCO**의 이해관계를 전체 네트워크와 일치시키는 것으로, **KLAY**의 가치를 높일 수 있도록 행동하게 하는 역할을 함
- **KLAY** 예치량에 따라 블록을 생성할 수 있는 기회를 더 받게 됨

불평등 완화 / 네트워크 안정성 강화

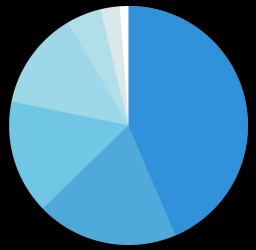
- 지나친 블록 생성 편중을 막기 위해 **Gini** 계수 적용. 이를 통해 기회의 불평등을 완화

$$G = \frac{\sum_{i=0}^{n} \sum_{j=0}^{n} \mathrm{abs}(x_i - x_j)}{2 \cdot n^2 \cdot \mathrm{mean}(x)}$$

- 많은 양을 스테이킹한 노드의 이상 동작시 네트워크의 안정성을 확보하기 위해 극소수의 양만 스테이킹 하고 있더라도 최소 블록 생성 기회를 확보할 수 있도록 보정

Proposer Selection

Klaytn은 스테이킹 기반의 **Proposer** 선정 시스템을 가지고 있음



[Checking Share]

A : 42.7% → 43 slots
B : 21.2% → 21 slots
C : 5% → 5 slots
….
X : 0.1% → **1** slots

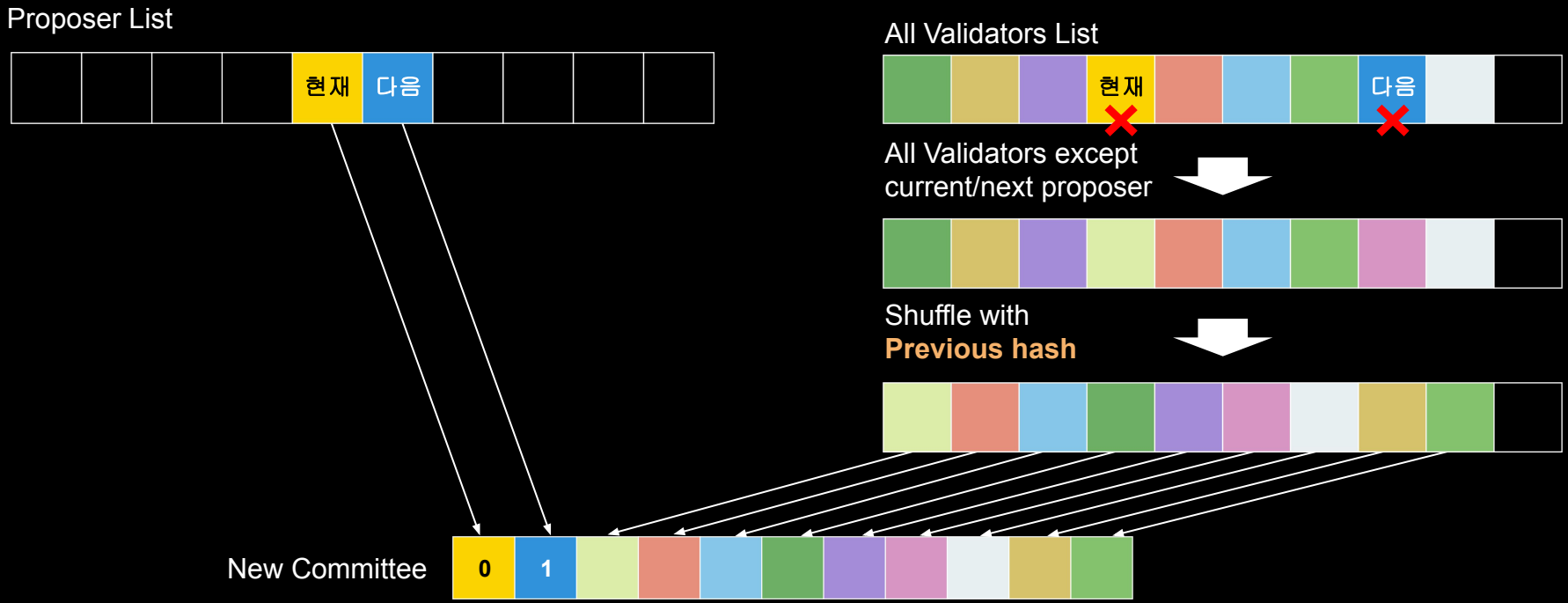
[Share to # of Slots]

[Flatten to an array]

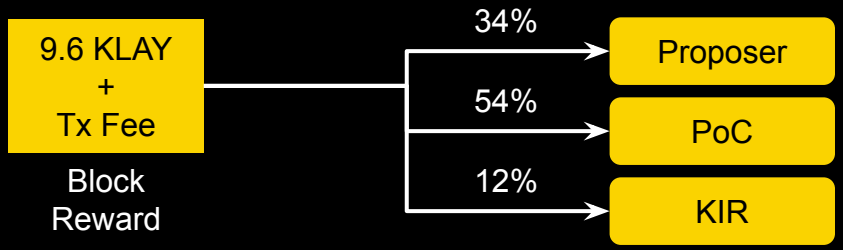
[Shuffle with **previous hash**]

[Sequentially become a proposer]

Committee Selection



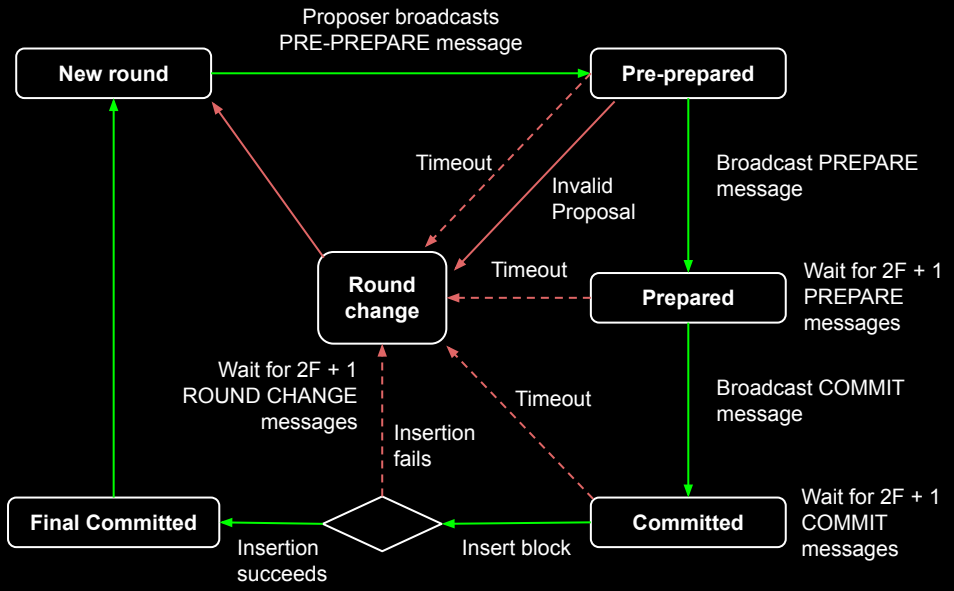
Reward



- 블록을 생성한 **Proposer, Proof of Contribution (PoC), Klaytn Improvement Reserve(KIR)**로 분배
- **PoC**의 경우 생태계 활성화에 기여하는 서비스에 지급, **KIR**은 향후 **Klaytn** 개선을 위한 자금으로 활용
- 이러한 자금의 집행은 **CCO (Core Cell Operator)**의 모임인 **Governance Council**에서 결정
- 위의 배분 비율 및 블록당 주조되는 **KLAY**의 양 등은 **Governance Council**의 투표로 변경될 수 있음
- 이러한 투표를 위한 기능은 이미 **Klaytn**에 포함되어 있음

Roundchange

- 어떠한 문제로 기다리던 동작이 완료되지 않을 경우 발생
    - **Proposer**가 블록을 제안 하지 않거나
    - ⅔ 이상의 **Prepare** 또는 **Commit** 메시지를 받지 못한 경우
- 이 경우 다음 **Proposer**가 블록 제안
- 네트워크의 속도 지연 요인이며, 발생 시 빠르게 복원 가능해야 함



Future work

- *메시지 전송 개선*:
  - 컨센서스 메시지를 다음 **Round**의 **Committee**까지 보내도록 변경
  - 현재 **Committee**에게만 보낼 경우, **Roundchange** 발생했을 때 메시지 불일치 발생 가능
  - 통신량을 일정하게 유지하되, 문제 상황에서 빠르게 벗어나도록 대응

- ***Committee** 구성 개선*:
  - **Roundchange**가 발생했을 때 **Committee**의 변경 폭이 커지도록 변경
  - 내년 상반기 중 적용 예정

- *헤더 크기 개선*:
  - 빠른 블럭 생성으로, 헤더의 크기가 전체 데이터량에 많은 영향을 미침
  - 정보를 압축하여 최소한의 데이터 저장

Summary - Consensus Algorithm

- **PBFT** 기반의 알고리즘을 사용 ⇒ 빠른 블록 생성 속도와 **Finality**를 확보

- **Staking** 기반의 **Proposer Selection** 및 **Reward System** 구현

- **Committee** 구성을 통해 **BFT**의 노드 수 제한을 극복

- 문제 상황에 대한 대응 및 블록 데이터량 개선 예정

THANK YOU

Ground X
27F, 521, Teheran-ro,
Gangnam-gu, Seoul, Republic of Korea