Blockchain #9

Fork and Consensus

Prof. Byung Il Kwak

Review

- Security of Bitcoin
- Attacks on mining pools
- Blacklisting
- Selfish mining
- Attacks on exchange markets

Outline

- Hard Fork vs. Soft Fork
- Byzantine Fault Tolerance Model
- Federated Byzantine Agreement
- Proof of Elapsed Time

CONTENTS

- Hard Fork vs. Soft Fork

Hard Fork vs. Soft Fork

- 탈중앙화 시스템은 유지 보수와 업데이트 문제를 항상 가지고 있음

Why?

Hard Fork vs. Soft Fork

- 블록체인 시스템의 업데이트 2가지 종류

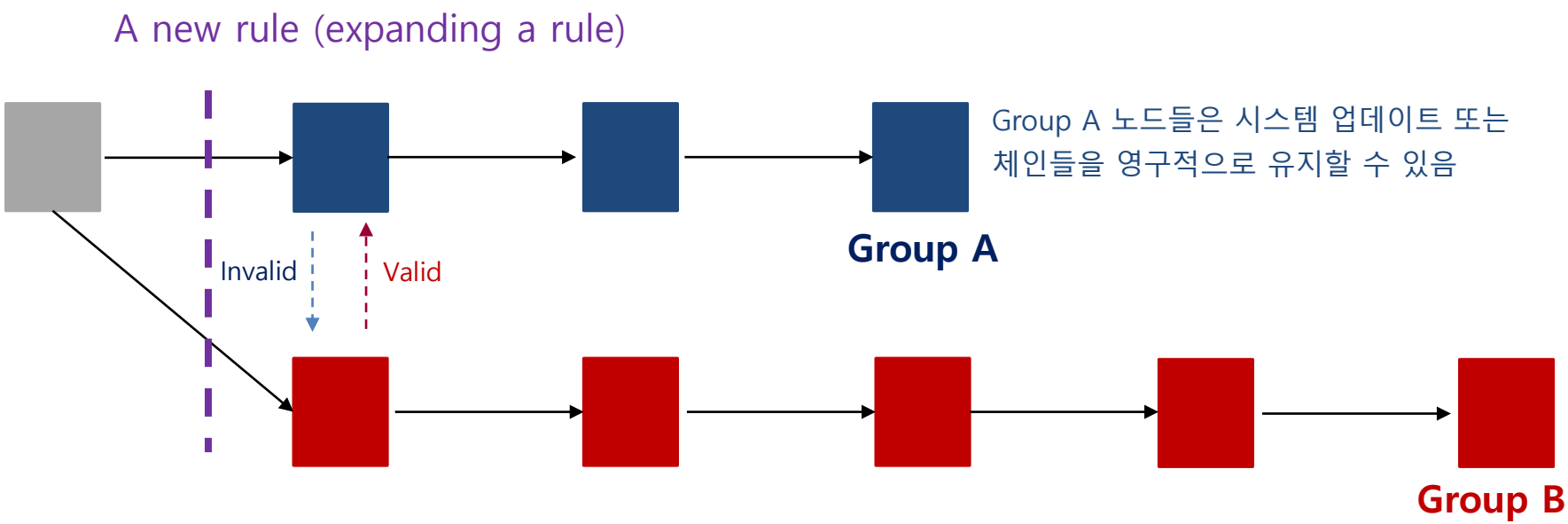
Soft vs Hard Forks

Soft Fork	Hard Fork
Tightening the rules (eg, 1MB -> 0.5MB)	Expanding the rules (eg, 1MB -> 2MB)
Backwards compatible	Not backwards compatible
Old nodes accept new blocks	Old nodes don't accept new blocks

		Update Group	No Update Group
Hard fork	Case 1	Hash power > 50%	Hash power < 50%
	Case 2	Hash power < 50%	Hash power > 50%
Soft fork	Case 3	Hash power > 50%	Hash power < 50%
	Case 4	Hash power < 50%	Hash power > 50%

Hard Fork vs. Soft Fork

- (Case 1) Hard Fork (e.g., 1MB -> 2MB)
  - Group A: 기존 블록체인 노드 (해시 파워 < 50%)
  - Group B: 업데이트된 블록체인 노드 (해시 파워 > 50%)

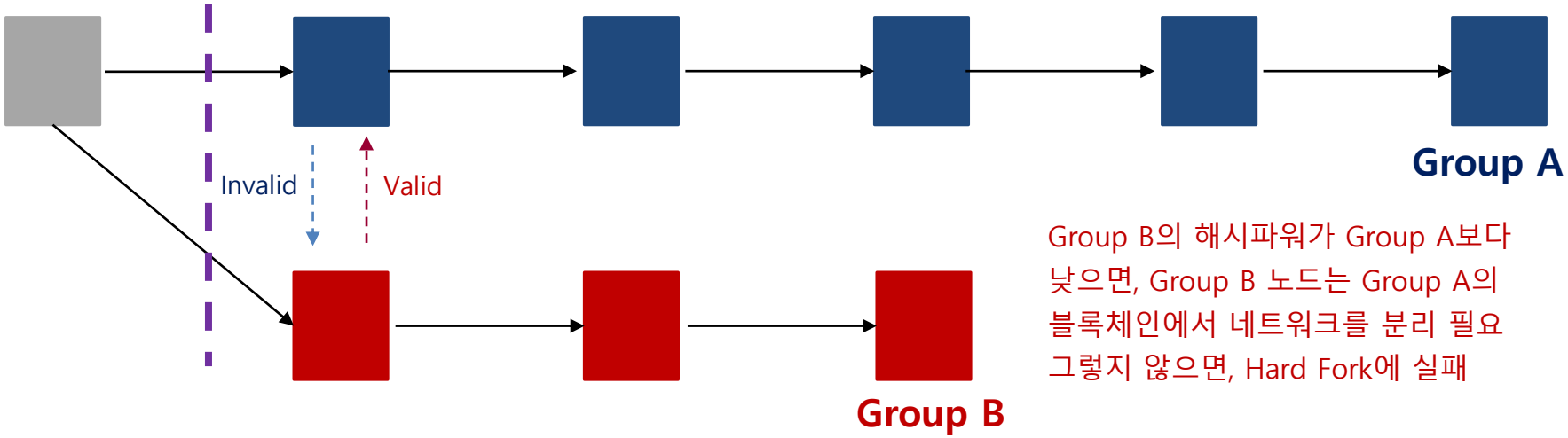


Hard Fork: 모든 노드들은 포크 합의를 위해 시스템 업데이트를 해야함

Hard Fork vs. Soft Fork

- (Case 2) Hard Fork (e.g., 1MB -> 2MB)
  - Group A: 기존 블록체인 노드 (해시 파워 > 50%)
  - Group B: 업데이트된 블록체인 노드 (해시 파워 < 50%)

A new rule (expanding a rule)

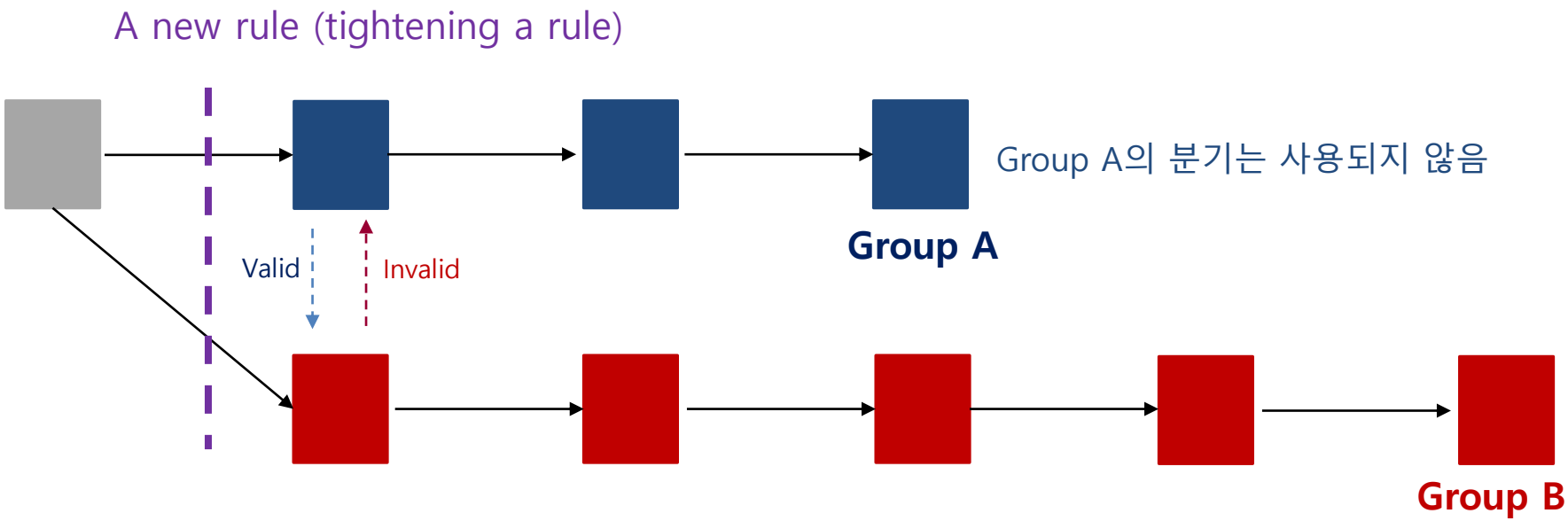


Group B의 해시파워가 Group A보다 낮으면, Group B 노드는 Group A의 블록체인에서 네트워크를 분리 필요
그렇지 않으면, Hard Fork에 실패

Hard Fork: 모든 노드들은 포크 합의를 위해 시스템 업데이트를 해야함

Hard Fork vs. Soft Fork

- (Case 3) Soft Fork (e.g., 2MB -> 1MB)
  - Group A: 기존 블록체인 노드 (해시 파워 < 50%)
  - Group B: 업데이트된 블록체인 노드 (해시 파워 > 50%)

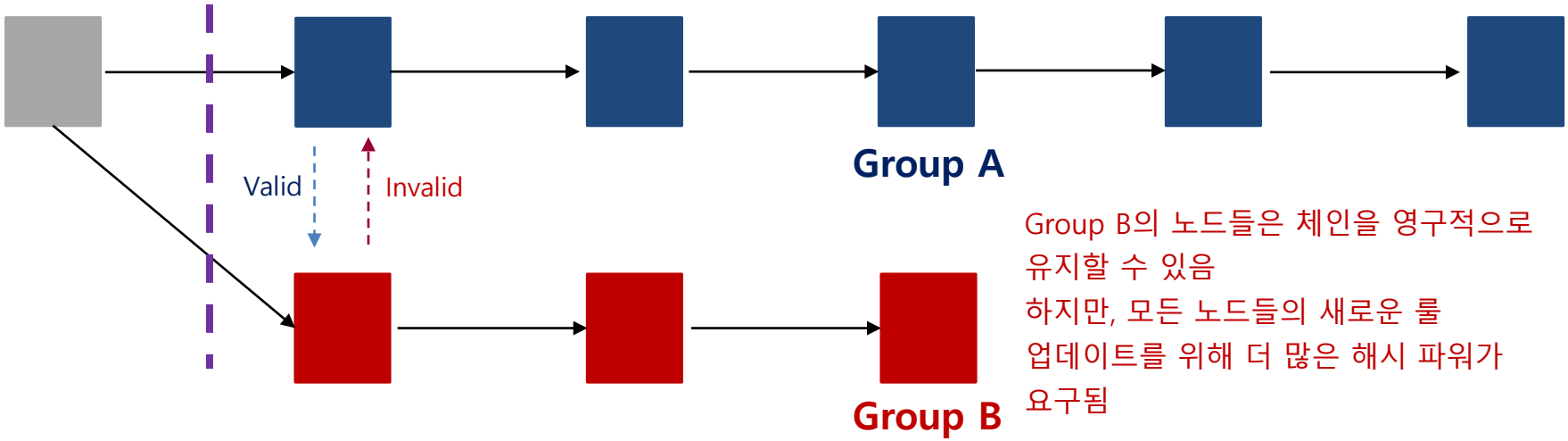


Soft Fork: 모든 노드들은 자연스럽게 새로운 규칙 (Soft Fork)을 따르게 됨
성공적인 Soft Fork를 위해서는 새로운 규칙에 동의하는 많은 채굴자들이 필요

Hard Fork vs. Soft Fork

- (Case 4) Soft Fork (e.g., 2MB -> 1MB)
  - Group A: 기존 블록체인 노드 (해시 파워 > 50%)
  - Group B: 업데이트된 블록체인 노드 (해시 파워 < 50%)

A new rule (tightening a rule)



Group B의 노드들은 체인을 영구적으로 유지할 수 있음
하지만, 모든 노드들의 새로운 룰 업데이트를 위해 더 많은 해시 파워가 요구됨

성공적인 Soft Fork를 위해서는 새로운 규칙에 동의하는 많은 채굴자가 필요

CONTENTS

- Byzantine Fault Tolerance Model

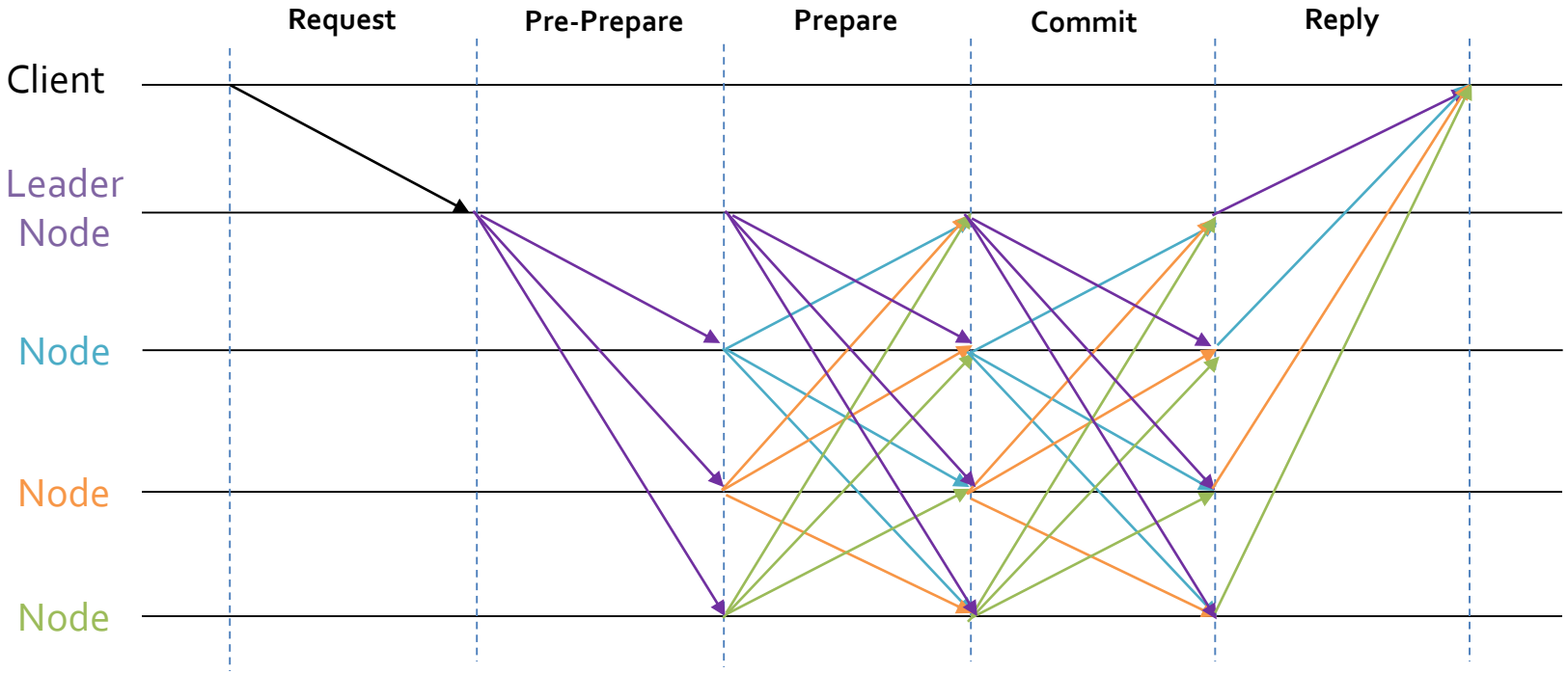
Byzantine Fault Tolerance Model

- Byzantine Fault Tolerance (BFT)
  - 악의적이거나 결함이 있는 노드를 어느 정도 허용
    - N: 검증자의 수
    - Practical Byzantine Fault Tolerance (PBFT) 알고리즘에서는
      - $N \geq 3f + 1, where\ f\ is\ the\ number\ of\ faulty\ node$
    - 즉, 결함이 있더라도, 전체의 1/3을 넘지 않으면, 시스템이 정상 작동하도록 허용 (정상 참여자가 최소 2/3는 되어야 함을 가정)
    - 이때, 검증자는 서로를 알고 있으며, 검증자를 추가하거나 제거하기 위해서는 나머지 검증자들의 승인이 필요
  - BFT는 허가된 블록체인 시스템 (예: HyperLedger Fabric)에 적용됨

** **PBFT**는 BFT 합의 알고리즘이 동기식 네트워크에서만 합의 가능한 문제를 비동기 네트워크에서도 합의를 이룰 수 있도록 개선한 알고리즘

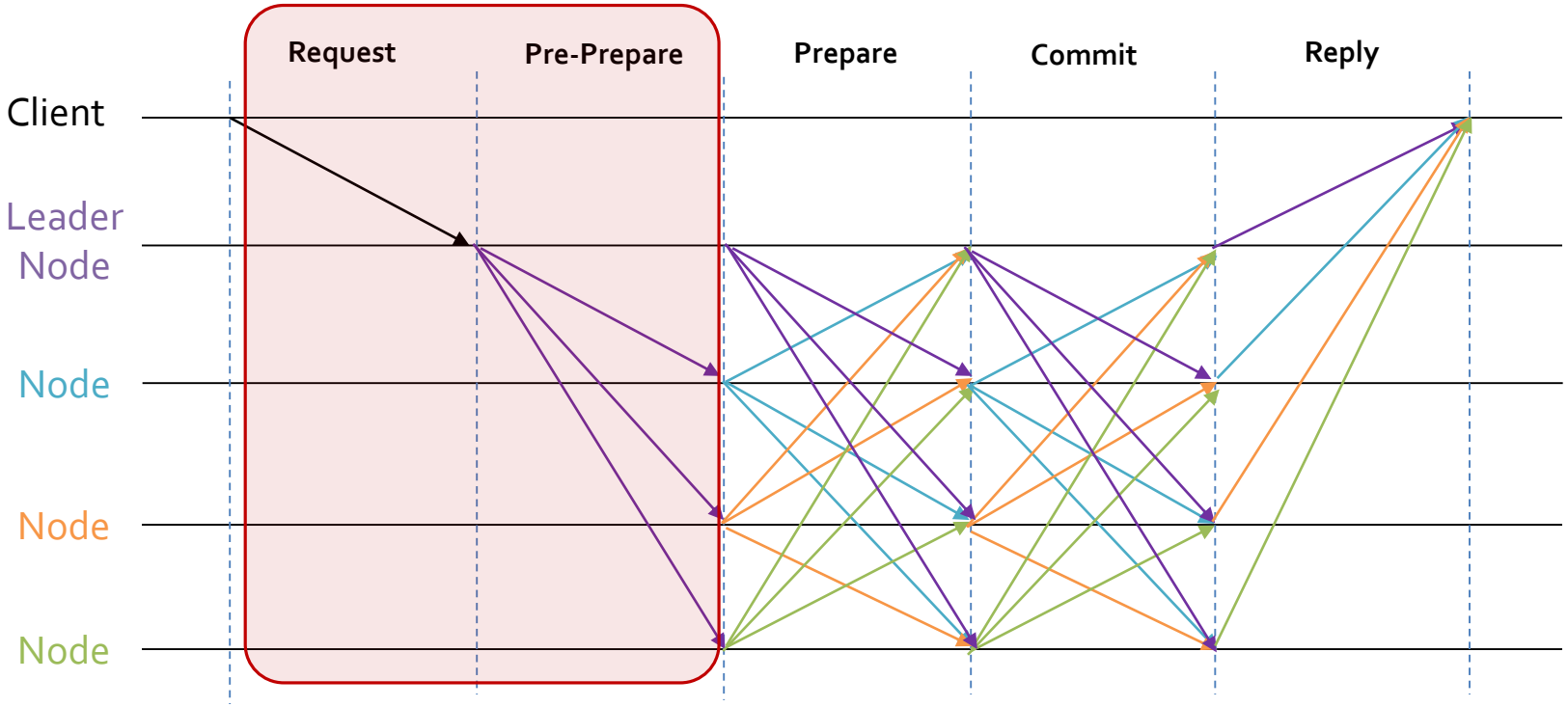
Byzantine Fault Tolerance Model

- Practical BFT 알고리즘
  - $N \geq 3f + 1, where\ f\ is\ the\ number\ of\ faulty\ node$
  - PBFT는 전체 5가지 절차로 구성됨



Byzantine Fault Tolerance Model

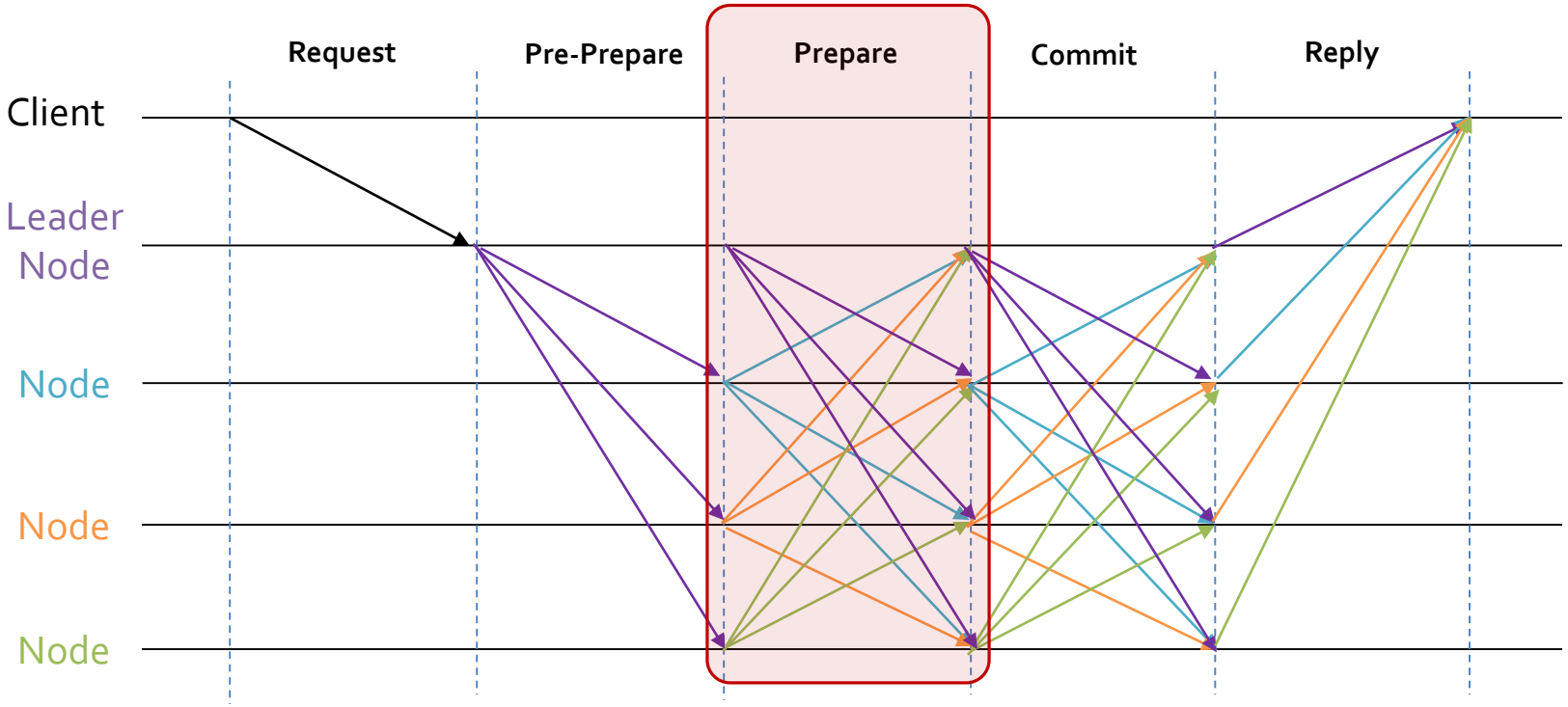
- Practical BFT 알고리즘
  - $N \geq 3f + 1, where\ f\ is\ the\ number\ of\ faulty\ node$
  - PBFT는 전체 5가지 절차로 구성됨



** 1. 리더가 클라이언트들의 요청을 수집하여 실행 결과를 다른 노드들에게 전파
2. 리더의 메시지를 받은 노드들은 다른 노드들에게서 받은 메시지를 다시 한번 나머지 노드들에게 전파
3. 모든 노드들은 자신이 다른 노드들에게서 가장 많이 받은 중복된 메시지가 무엇인지를 다른 노드들에게 전파
4. 모든 노드들은 2/3 이상 동의한, 즉 생성 가능한 블록으로 인정하여 합의를 이룬 같은 데이터를 응답으로 Client에게 전송

Byzantine Fault Tolerance Model

- Practical BFT 알고리즘
  - $N \geq 3f + 1, where\ f\ is\ the\ number\ of\ faulty\ node$
  - PBFT는 전체 5가지 절차로 구성됨



** 1. 리더가 클라이언트들의 요청을 수집하여 실행 결과를 다른 노드들에게 전파

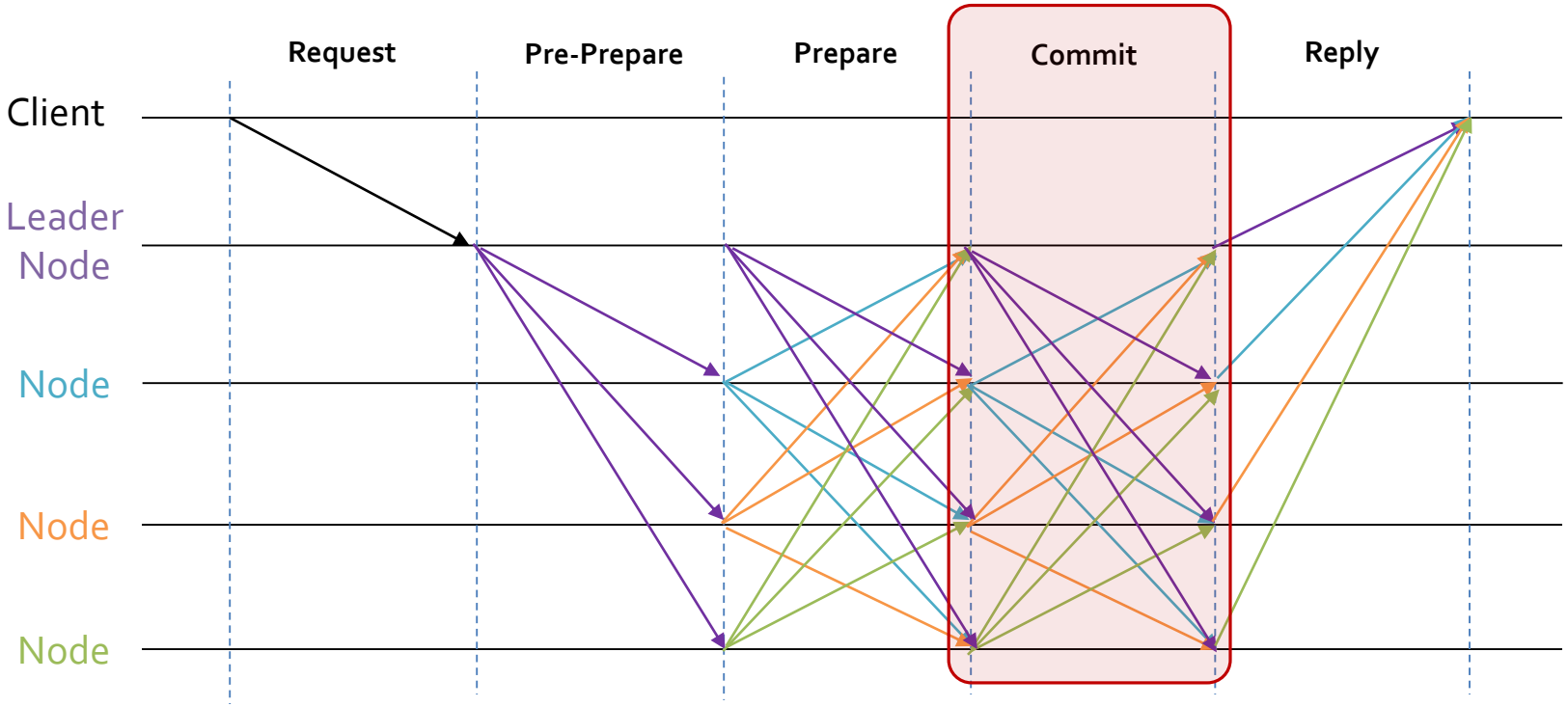
2. 리더의 메시지를 받은 노드들은 다른 노드들에게서 받은 메시지를 다시 한번 나머지 노드들에게 전파

3. 모든 노드들은 자신이 다른 노드들에게서 가장 많이 받은 중복된 메시지가 무엇인지를 다른 노드들에게 전파

4. 모든 노드들은 2/3 이상 동의한, 즉 생성 가능한 블록으로 인정하여 합의를 이룬 같은 데이터를 응답으로 Client에게 전송

Byzantine Fault Tolerance Model

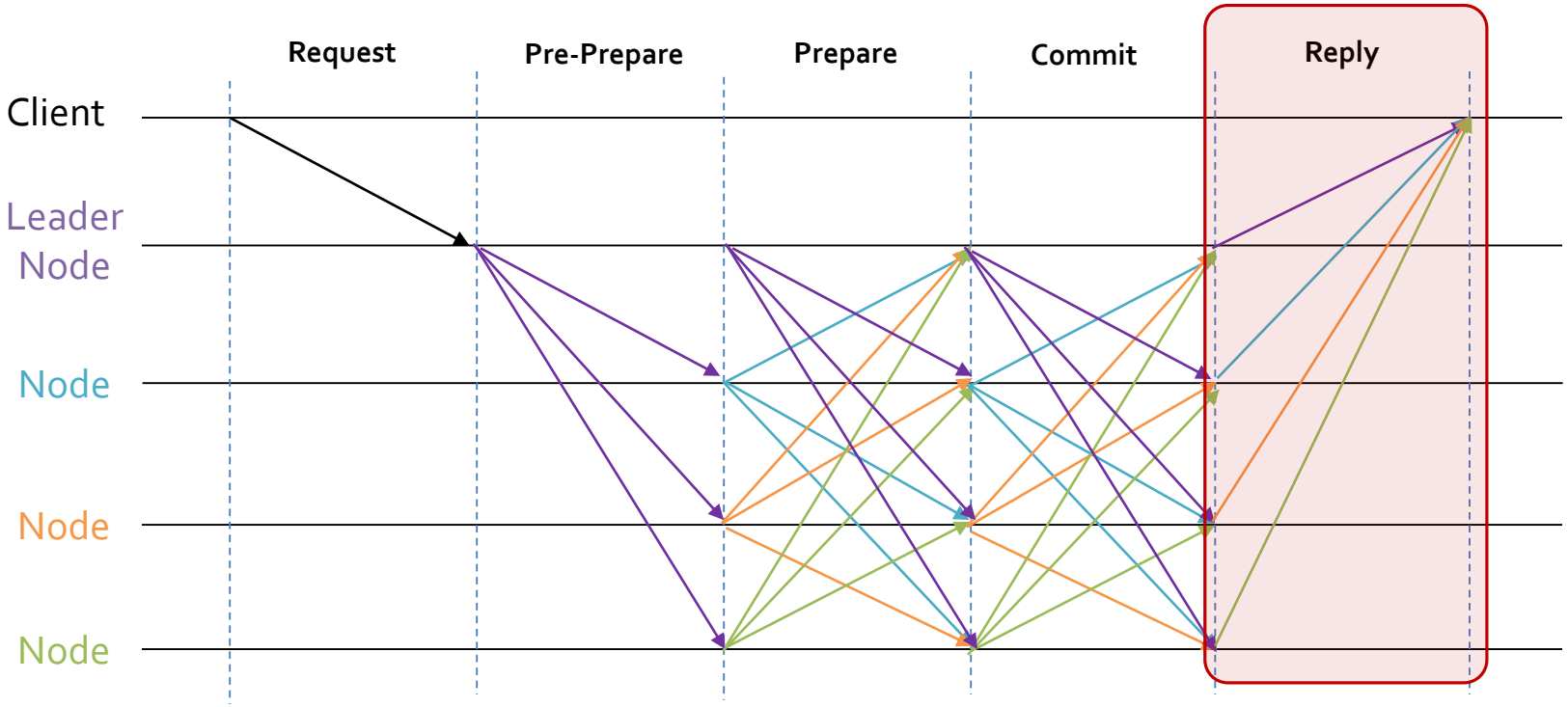
- Practical BFT 알고리즘
  - $N \geq 3f + 1, where\ f\ is\ the\ number\ of\ faulty\ node$
  - PBFT는 전체 5가지 절차로 구성됨



** 1. 리더가 클라이언트들의 요청을 수집하여 실행 결과를 다른 노드들에게 전파
2. 리더의 메시지를 받은 노드들은 다른 노드들에게서 받은 메시지를 다시 한번 나머지 노드들에게 전파
3. 모든 노드들은 자신이 다른 노드들에게서 가장 많이 받은 중복된 메시지가 무엇인지를 다른 노드들에게 전파
4. 모든 노드들은 2/3 이상 동의한, 즉 생성 가능한 블록으로 인정하여 합의를 이룬 같은 데이터를 응답으로 Client에게 전송

Byzantine Fault Tolerance Model

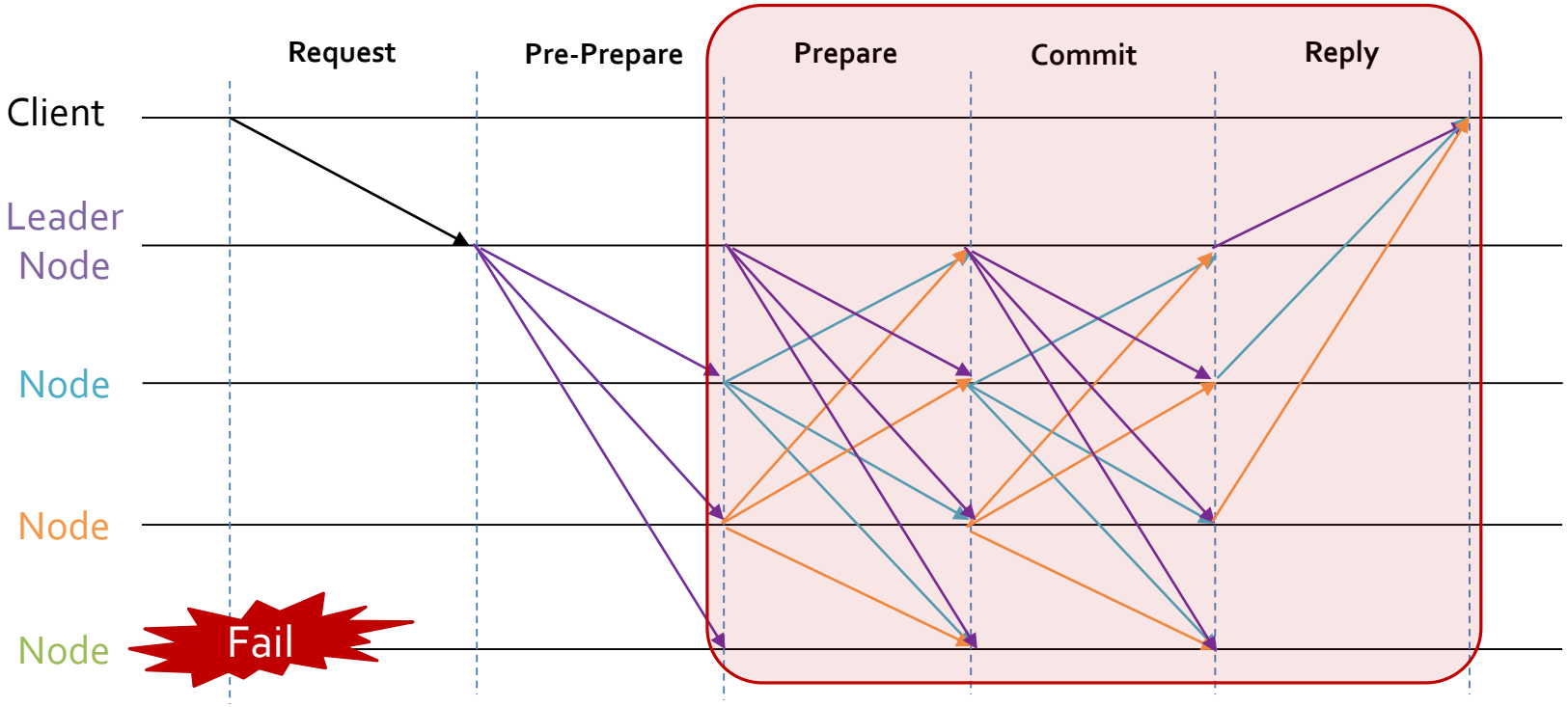
- Practical BFT 알고리즘
  - $N \geq 3f + 1, where\ f\ is\ the\ number\ of\ faulty\ node$
  - PBFT는 전체 5가지 절차로 구성됨



** 1. 리더가 클라이언트들의 요청을 수집하여 실행 결과를 다른 노드들에게 전파
2. 리더의 메시지를 받은 노드들은 다른 노드들에게서 받은 메시지를 다시 한번 나머지 노드들에게 전파
3. 모든 노드들은 자신이 다른 노드들에게서 가장 많이 받은 중복된 메시지가 무엇인지를 다른 노드들에게 전파
4. 모든 노드들은 2/3 이상 동의한, 즉 생성 가능한 블록으로 인정하여 합의를 이룬 같은 데이터를 응답으로 Client에게 전송

Byzantine Fault Tolerance Model

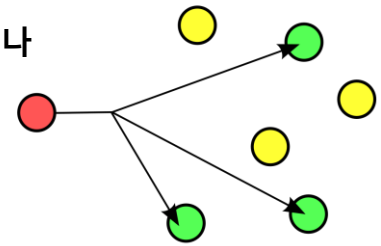
- Practical BFT 알고리즘
  - $N \geq 3f + 1, where\ f\ is\ the\ number\ of\ faulty\ node$
  - PBFT는 전체 5가지 절차로 구성됨



Byzantine Fault Tolerance Model

- Practical BFT 알고리즘
  - Request
    - client가 leader 노드에게 서비스 작업을 호출하기 위한 요청을 보냄 (예: transaction)
  - Pre-Prepare
    - leader 노드는 다른 노드들에게 해당 요청을 멀티캐스트 함

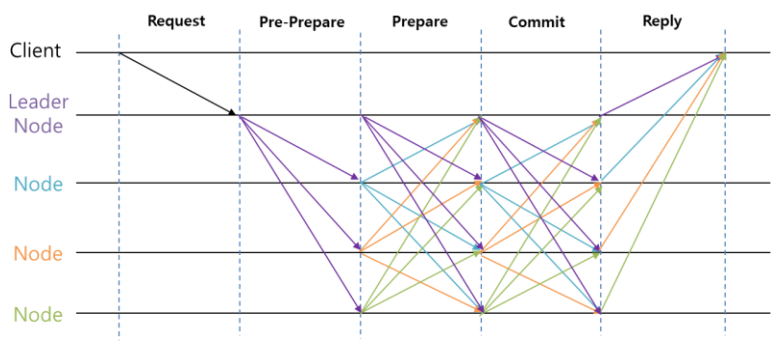
      **** 멀티캐스트 (Multicast)는 컴퓨터 네트워크에서 한번의 송신으로 메시지나 정보를 목표한 여러 컴퓨터에 동시에 전송을 의미**



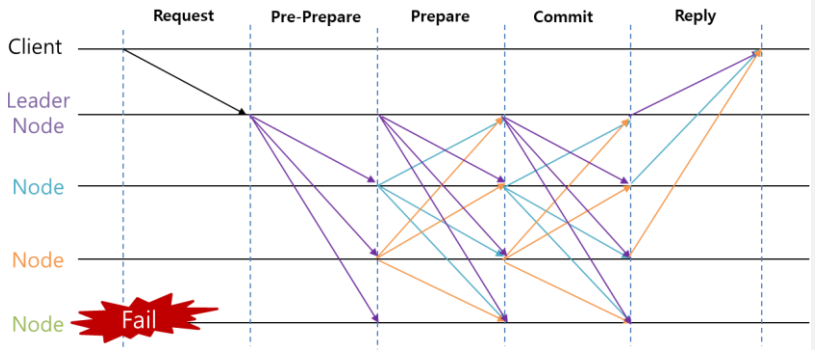
  - Prepare
    - leader 노드의 요청 메시지를 수락하면, 노드는 "prepare" 메시지를 다른 모든 노드들에게 멀티캐스트로 수락함을 알림

Byzantine Fault Tolerance Model

- Practical BFT 알고리즘
  - Commit
    - 모든 노드들은 다른 노드들로부터 $2f$의 정상 "prepare" 메시지를 받으면 다른 노드들에게 "commit" 메시지를 멀티캐스트함
  - Reply
    - 모든 노드는 자신의 "commit" 메시지를 포함한 $2f + 1$ 개의 정상 "commit" 메시지를 수락한 경우, 클라이언트에 응답을 보냄



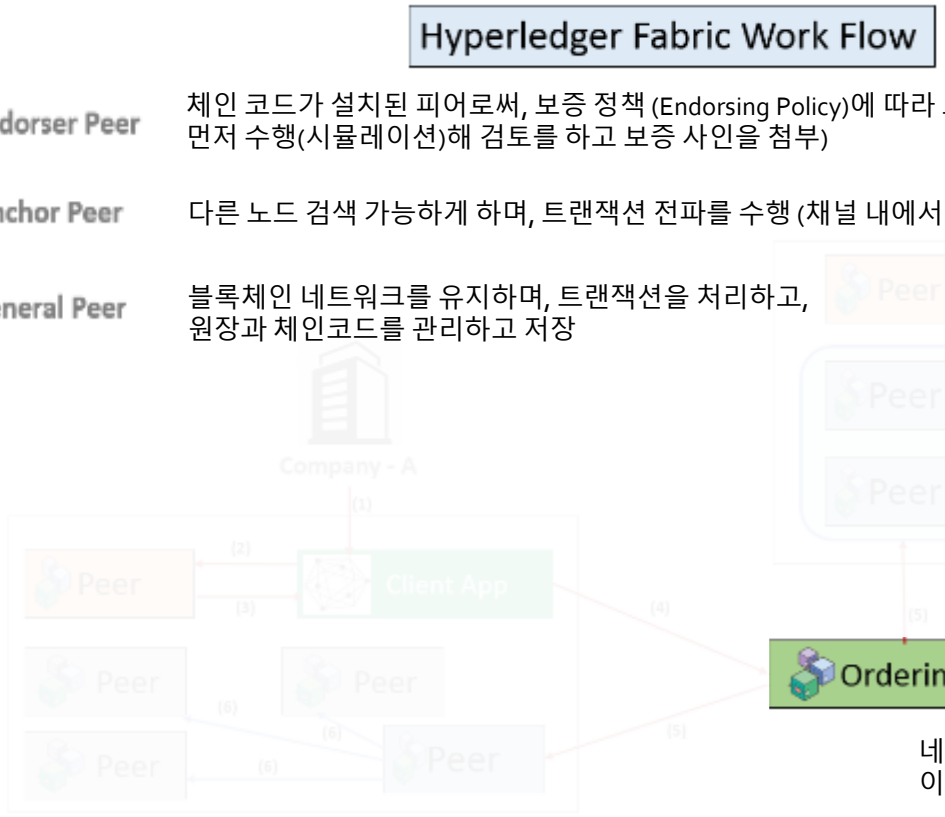
N=4, f=0



N=4, f=1

Byzantine Fault Tolerance Model

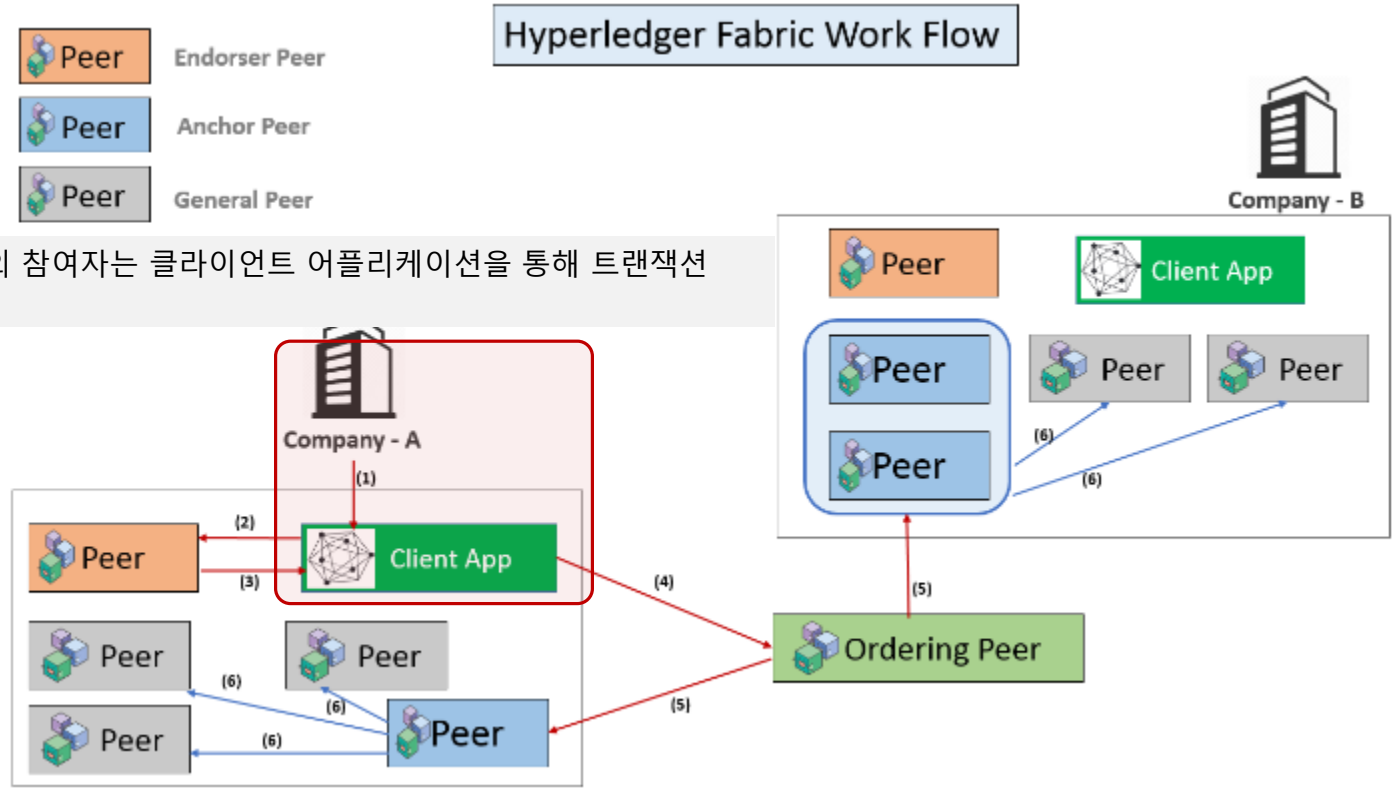
- HyperLedger
  - PBFT와 같은 합의 할고리즘을 기반으로 함



https://medium.com/coinmonks/how-does-hyperledger-fabric-works-cdb68e6o66f5

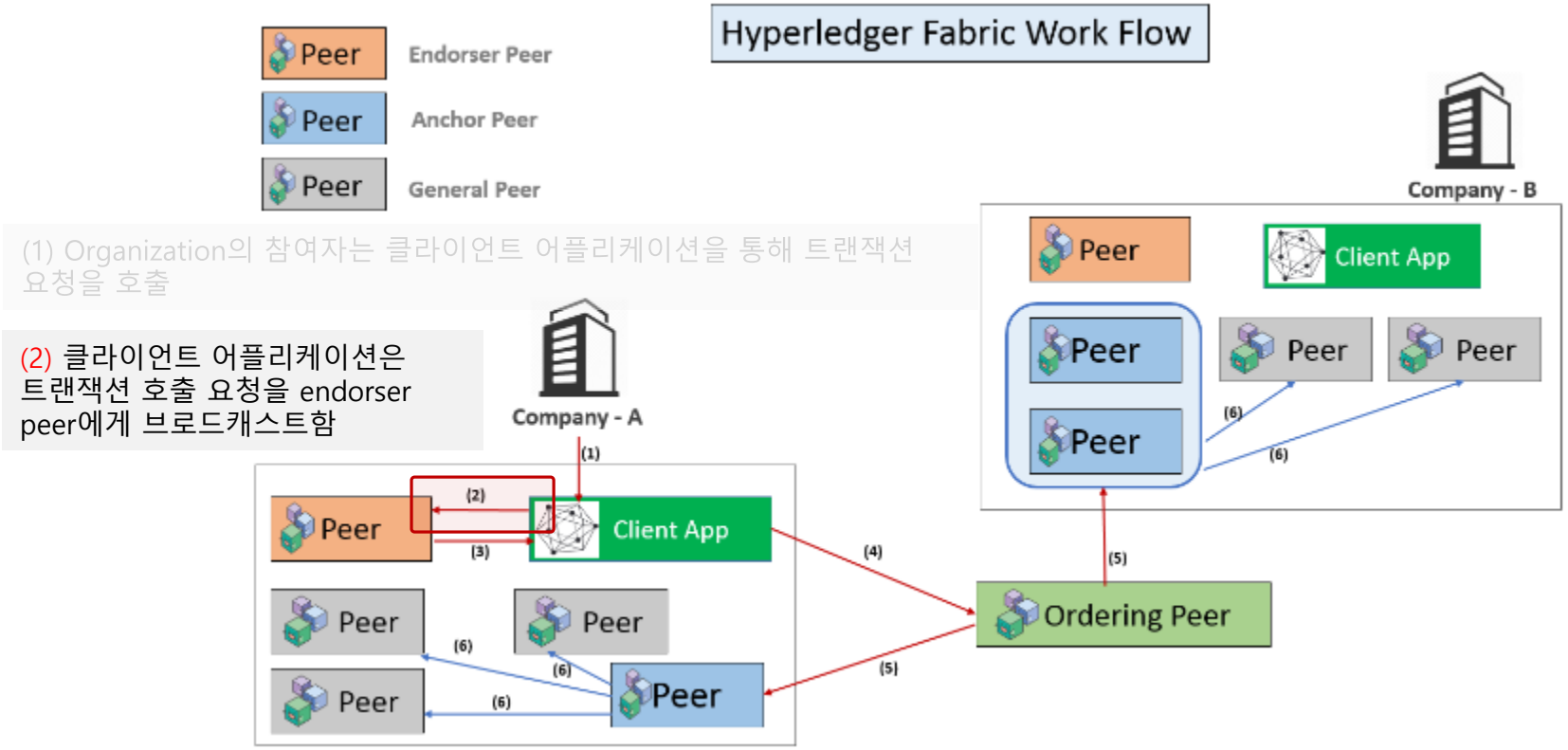
Byzantine Fault Tolerance Model

- HyperLedger
  - PBFT와 같은 합의 알고리즘을 기반으로 함



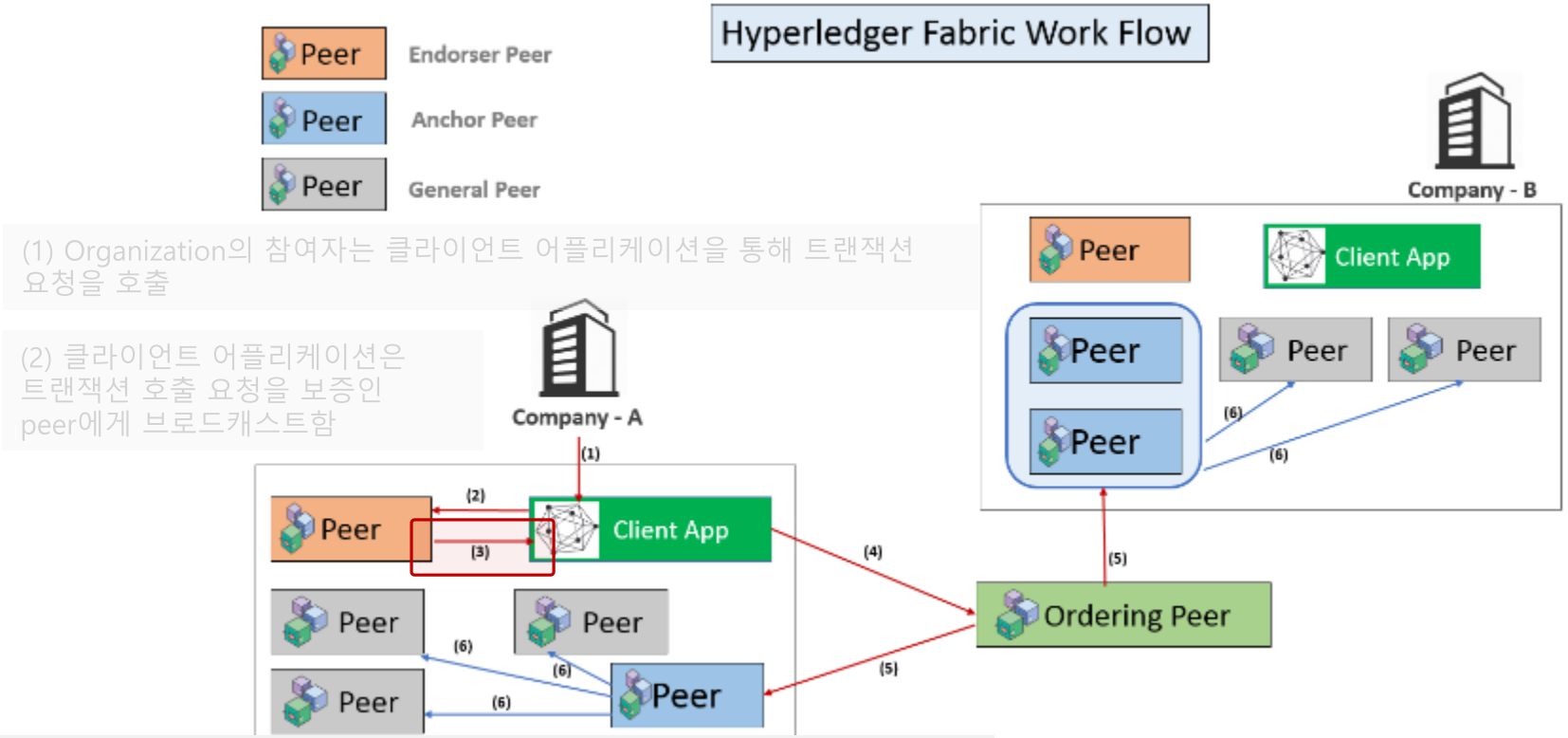
Byzantine Fault Tolerance Model

- HyperLedger
  - PBFT와 같은 합의 할고리즘을 기반으로 함



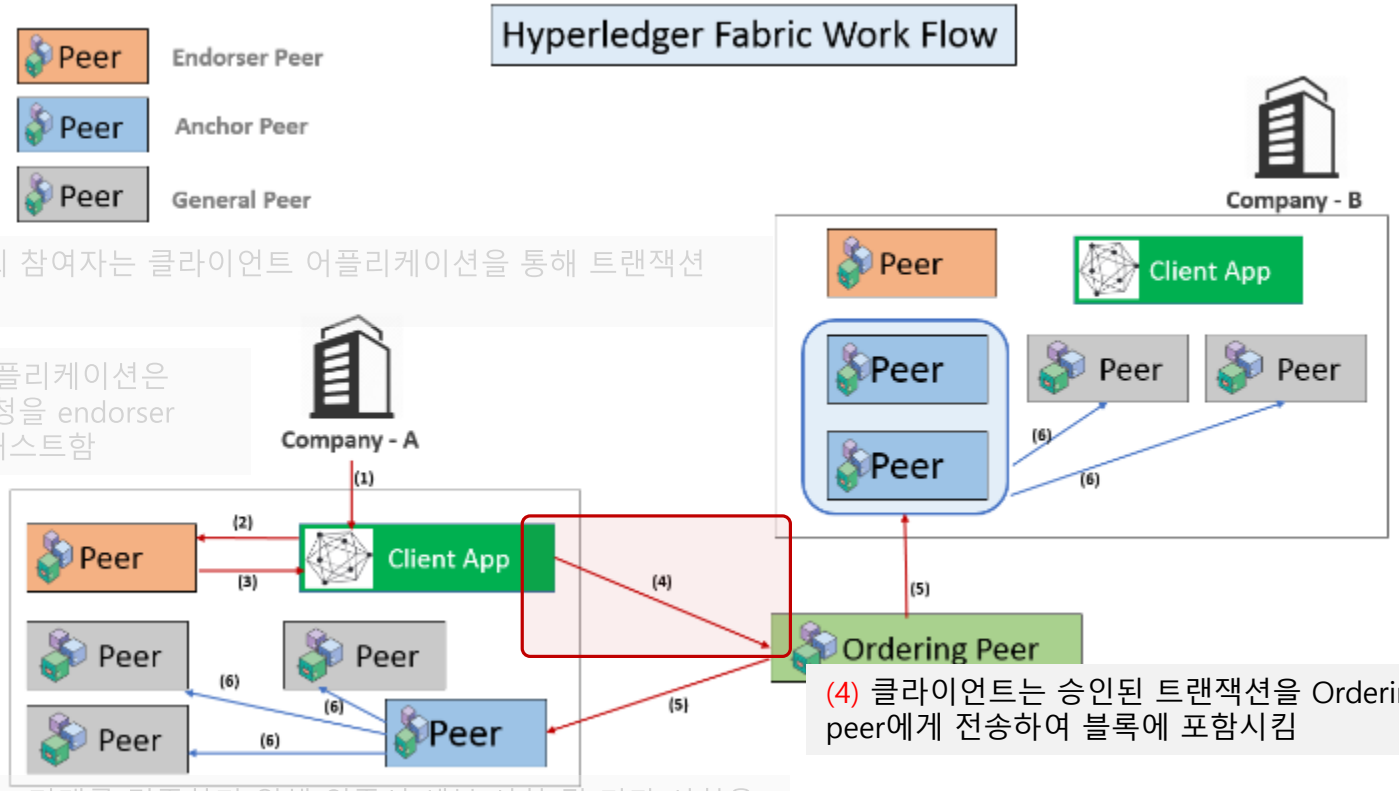
Byzantine Fault Tolerance Model

- HyperLedger
  - PBFT와 같은 합의 할고리즘을 기반으로 함



Byzantine Fault Tolerance Model

- HyperLedger
  - PBFT와 같은 합의 할고리즘을 기반으로 함



(1) Organization의 참여자는 클라이언트 어플리케이션을 통해 트랜잭션 요청을 호출

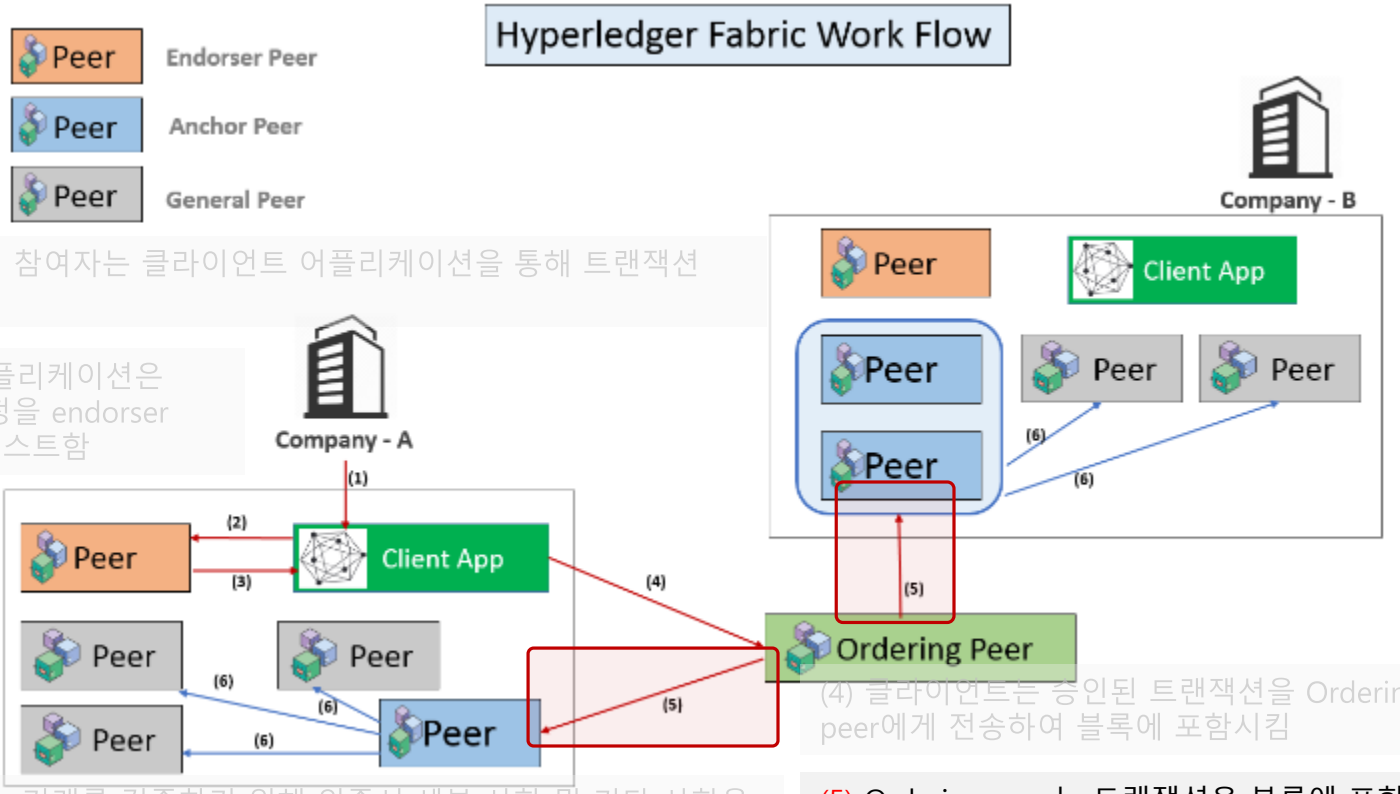
(2) 클라이언트 어플리케이션은 트랜잭션 호출 요청을 endorser peer에게 브로드캐스트함

(3) endorser peer는 거래를 검증하기 위해 인증서 세부 사항 및 기타 사항을 확인하고 (시뮬레이션), 응답으로 트랜잭션 승인 또는 거부를 보냄

(4) 클라이언트는 승인된 트랜잭션을 Ordering peer에게 전송하여 블록에 포함시킴

Byzantine Fault Tolerance Model

- HyperLedger
  - PBFT와 같은 합의 할고리즘을 기반으로 함



(1) Organization의 참여자는 클라이언트 어플리케이션을 통해 트랜잭션 요청을 호출

(2) 클라이언트 어플리케이션은 트랜잭션 호출 요청을 endorser peer에게 브로드캐스트함

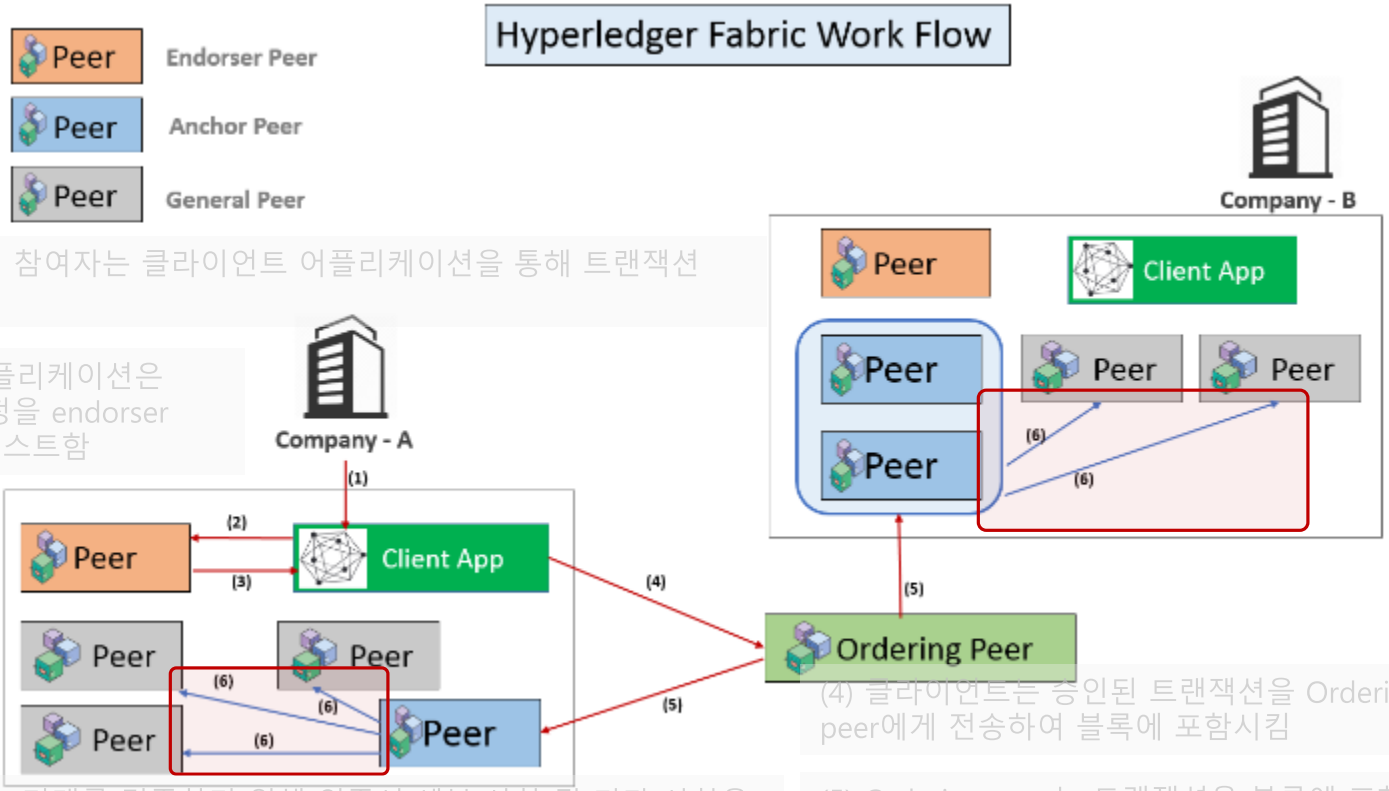
(3) endorser peer는 거래를 검증하기 위해 인증서 세부 사항 및 기타 사항을 확인하고 (시뮬레이션), 응답으로 트랜잭션 승인 또는 거부를 보냄

(4) 클라이언트는 승인된 트랜잭션을 Ordering peer에게 전송하여 블록에 포함시킴

(5) Ordering peer는 트랜잭션을 블록에 포함시키고 해당 블록을 Anchor 노드에게 전달

Byzantine Fault Tolerance Model

- HyperLedger
  - PBFT와 같은 합의 할고리즘을 기반으로 함



(1) Organization의 참여자는 클라이언트 어플리케이션을 통해 트랜잭션 요청을 호출

(2) 클라이언트 어플리케이션은 트랜잭션 호출 요청을 endorser peer에게 브로드캐스트함

(3) endorser peer는 거래를 검증하기 위해 인증서 세부 사항 및 기타 사항을 확인하고 (시뮬레이션), 응답으로 트랜잭션 승인 또는 거부를 보냄

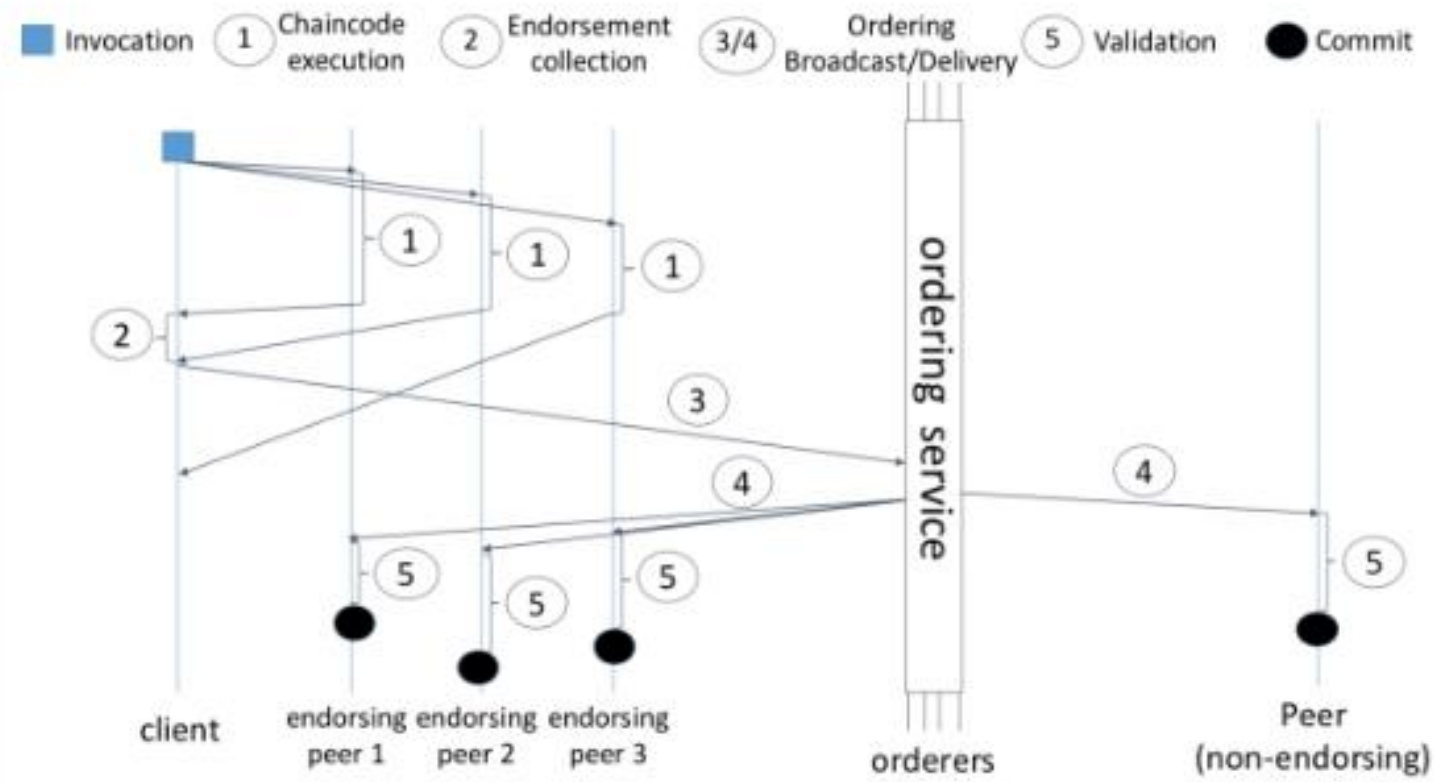
(4) 클라이언트는 승인된 트랜잭션을 Ordering peer에게 전송하여 블록에 포함시킴

(5) Ordering peer는 트랜잭션을 블록에 포함시키고 해당 블록을 Anchor 노드에게 전달

(6) Anchor peer는 블록을 자체 조직내의 다른 general peer들에게 전파하고, general peer들은 로컬 원장을 최신 블록으로 업데이트함 (네트워크들이 동일한 원장 소유)

Byzantine Fault Tolerance Model

서명 검증과 스마트계약 코드 실행



https://blog.acolyer.org/2018/06/04/hyperledger-fabric-a-distributed-operating-system-for-permissioned-blockchains/

CONTENTS

- Federated Byzantine Agreement

Federated Byzantine Agreement

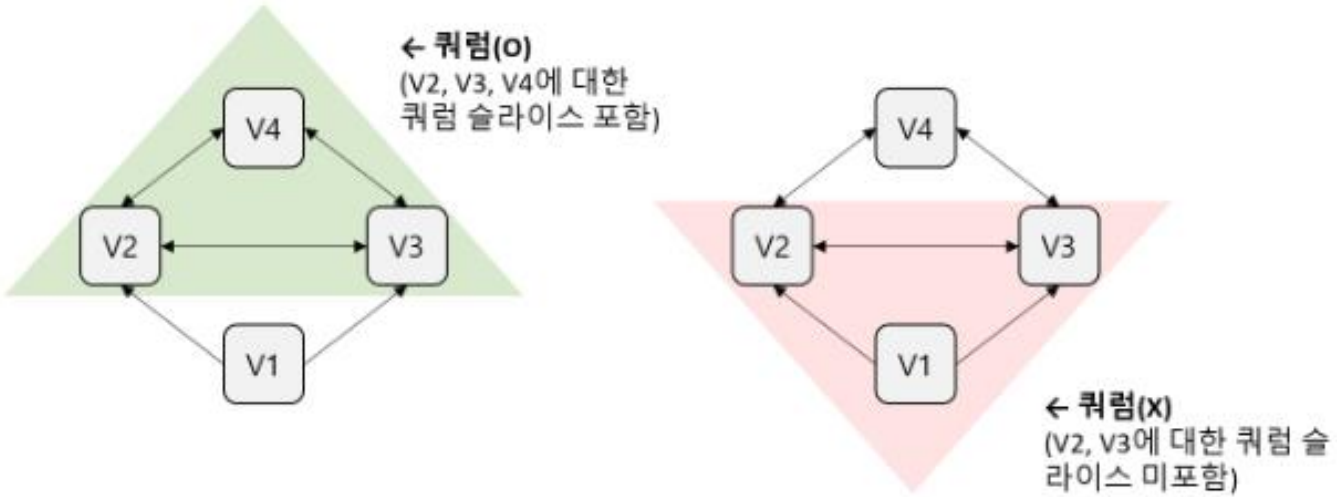
- Federated Byzantine Agreement (FBA)의 등장 배경
  - BFT상에서는 합의에 참여하는 전체 노드의 2/3이 내린 결과에 따라 합의가 이뤄짐
    - BFT하에서 악의적 노드가 전체 노드의 1/3이상이 될 경우, 네트워크 보안이 취약해짐
    - FBA 상에서는 악의적 노드가 정직한 노드로 구성된 *quorum slice*에 포함되지 않는 이상 *quorum slice* 내에서의 합의 과정에 영향을 줄 수 없음
    - 악의적 노드가 네트워크 공격에 성공하기 위해서는, 정직한 노드로 구성된 다수의 *quorum slice*가 다수의 악의적인 계정들을 골고루 편입시키도록 해야함. (공격이 제한적)

Federated Byzantine Agreement

- Quorum
  - 특정한 결과를 합의하기 위한 참가자들의 그룹
  - Quorum의 부분 집합이 quorum slice이며, quorum의 노드 수는 quorum slice의 노드 수 이상
  - Quorum은 quorum안에 속한 quorum slice를 모두 포함

- Quorum slice
  - 특정 노드가 집합의 합의 결과에 동의하도록 설득하는 노드 집합
  - 노드는 자신이 어떤 quorum slice에 들어갈 지 선택할 수 있음

Federated Byzantine Agreement



V1에 대한 quorum slice는 {v1, v2, v3}
V2에 대한 quorum slice는 {v2, v3, v4}

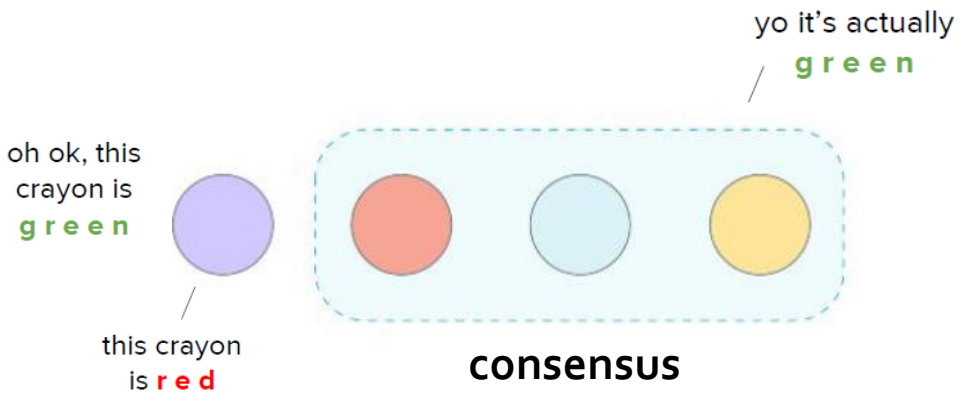
[Source: https://cryptoturtles.substack.com/p/-scp]

Federated Byzantine Agreement

- Problem: 분산 시스템에서 ***quorum (정족수)***를 어떻게 결정할 것인가?

  블록체인 시스템에서 합의를 이루기 위한 최소한의 투표수를 가진 소그룹

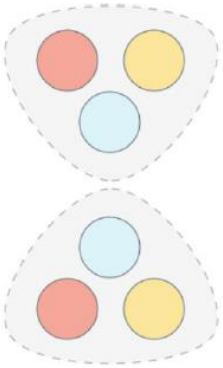
- Solution: **quorum slice**를 도입
  - 특정 합의 노드를 진행할 수 있는 **quorum**의 모임
    - 블록이 유효한지 아닌지를 결정
  - 개별 노드들은 자신들이 들어갈 quorum slice에 들어갈지 선택



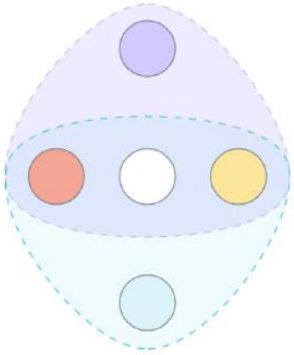
Federated Byzantine Agreement

- Idea: 여러 개의 quorum slice들이 함께 결합되면 어떻게 되는가?

- Quorum intersection
  - Quorum slice들은 **다른 quorum들을 느리게 확신시킬 것이며**, **더 큰 규모의 quorum을 형성**함
  - 그렇지 않으면, 서로 다른 것에 동의하는 **분리된 정족수**를 얻음



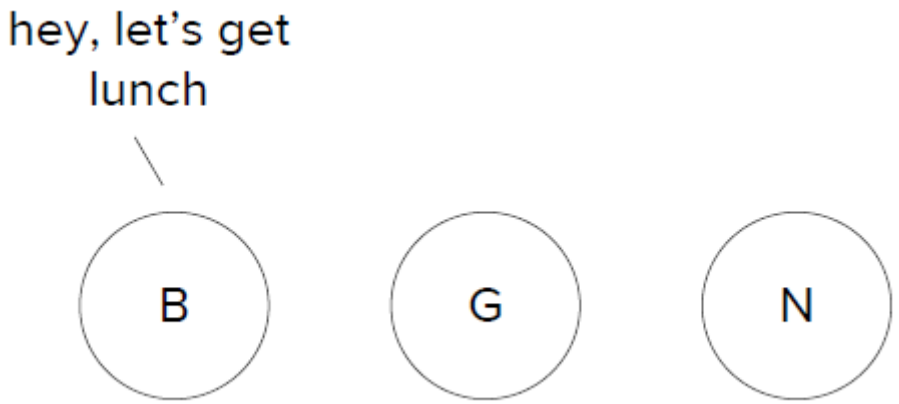
Disjoint quorums



Quorum intersection

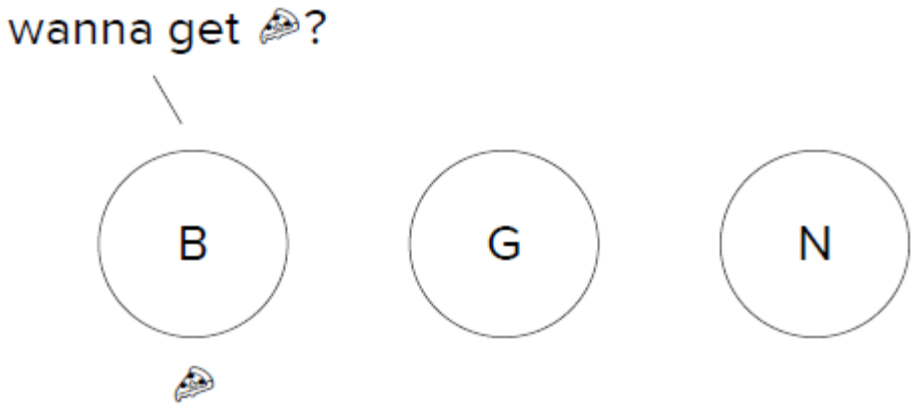
Federated Byzantine Agreement

- Lunchtime Consensus



Federated Byzantine Agreement

- Lunchtime Consensus



Federated Byzantine Agreement

- Lunchtime Consensus

Federated Byzantine Agreement

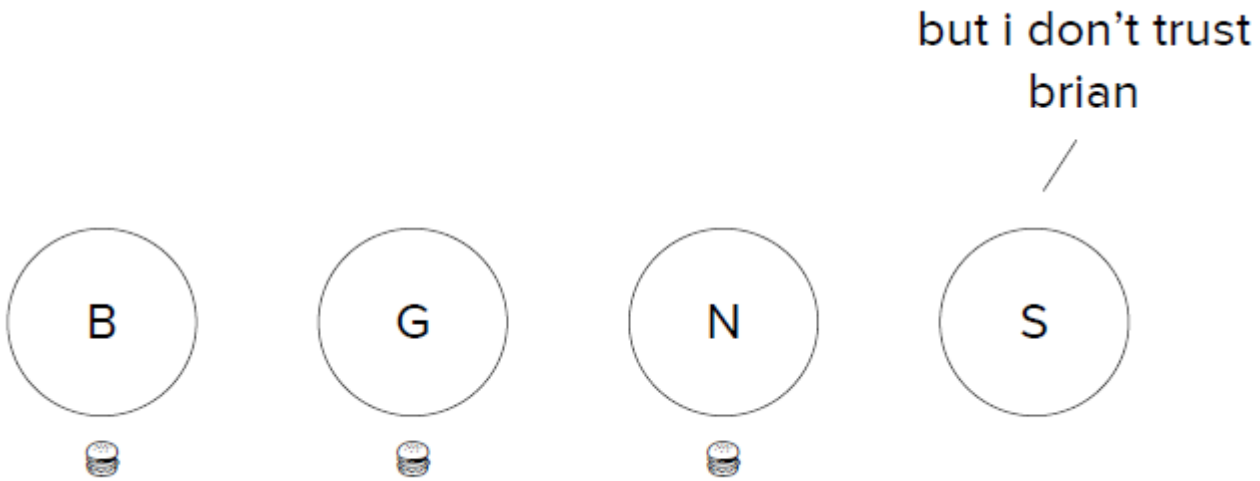
- Lunchtime Consensus

Federated Byzantine Agreement

- Lunchtime Consensus

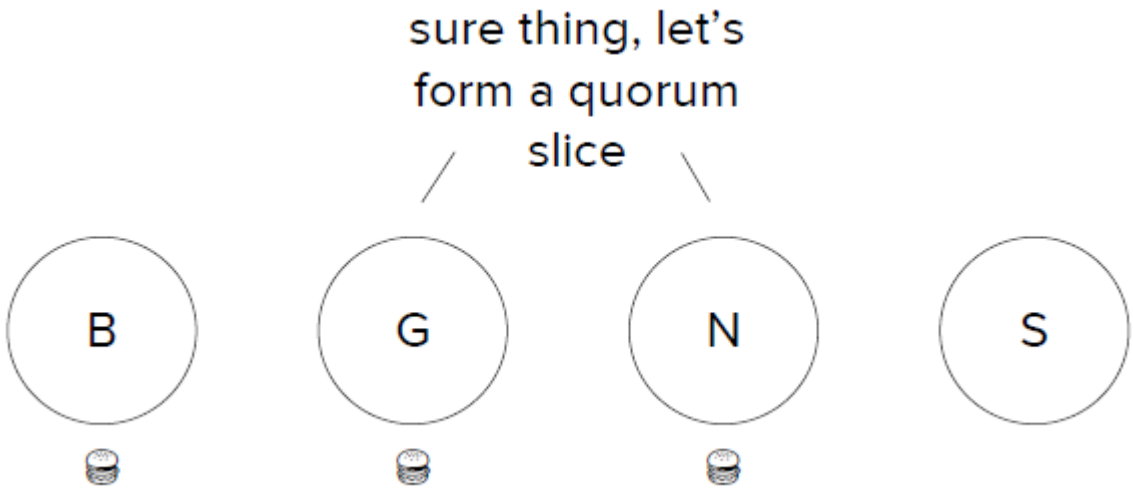
Federated Byzantine Agreement

- Lunchtime Consensus



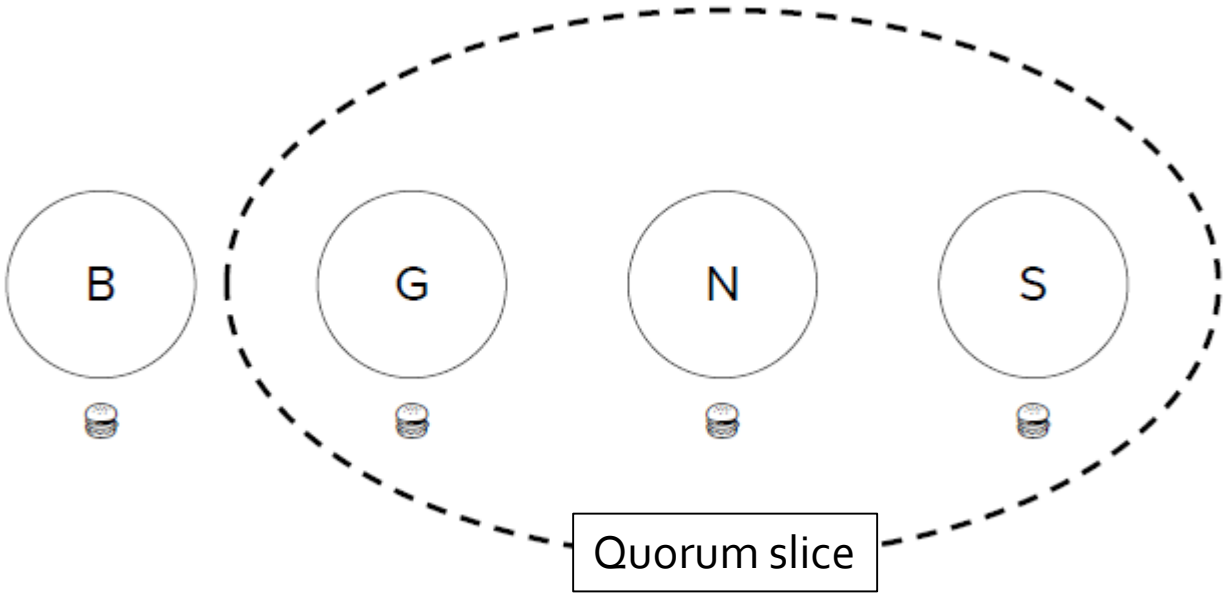
Federated Byzantine Agreement

- Lunchtime Consensus



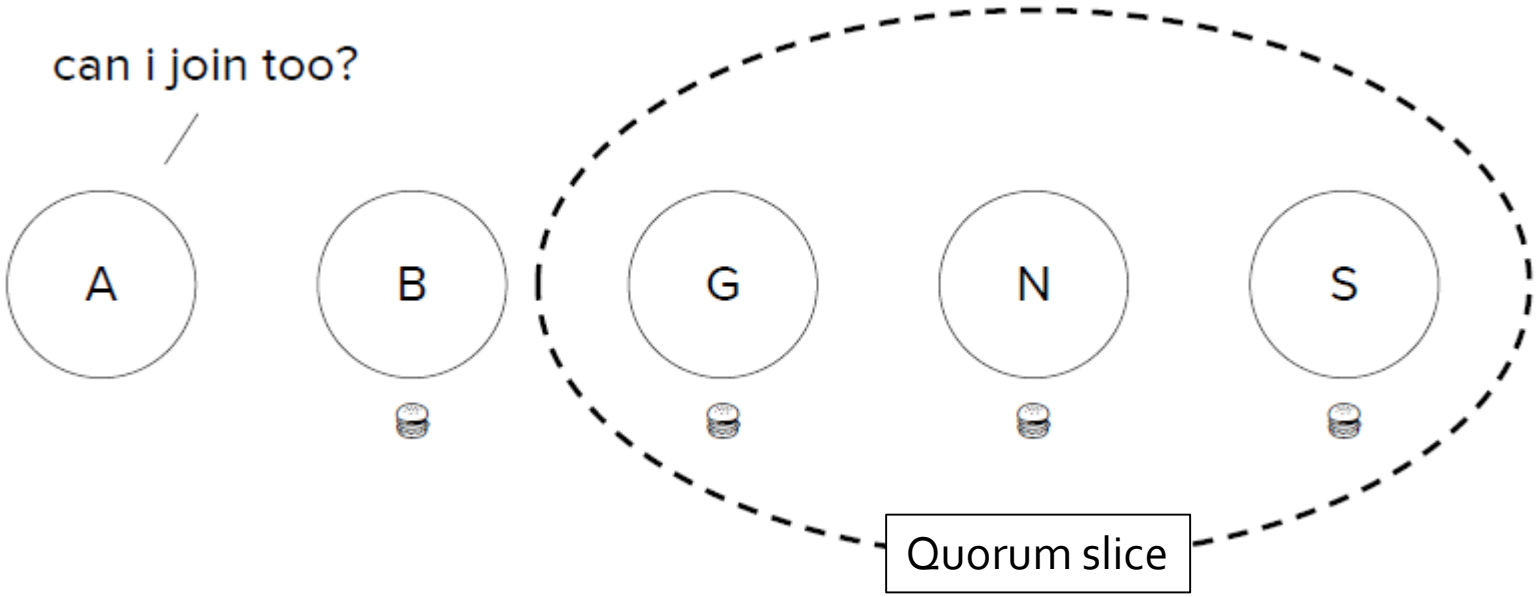
Federated Byzantine Agreement

- Lunchtime Consensus



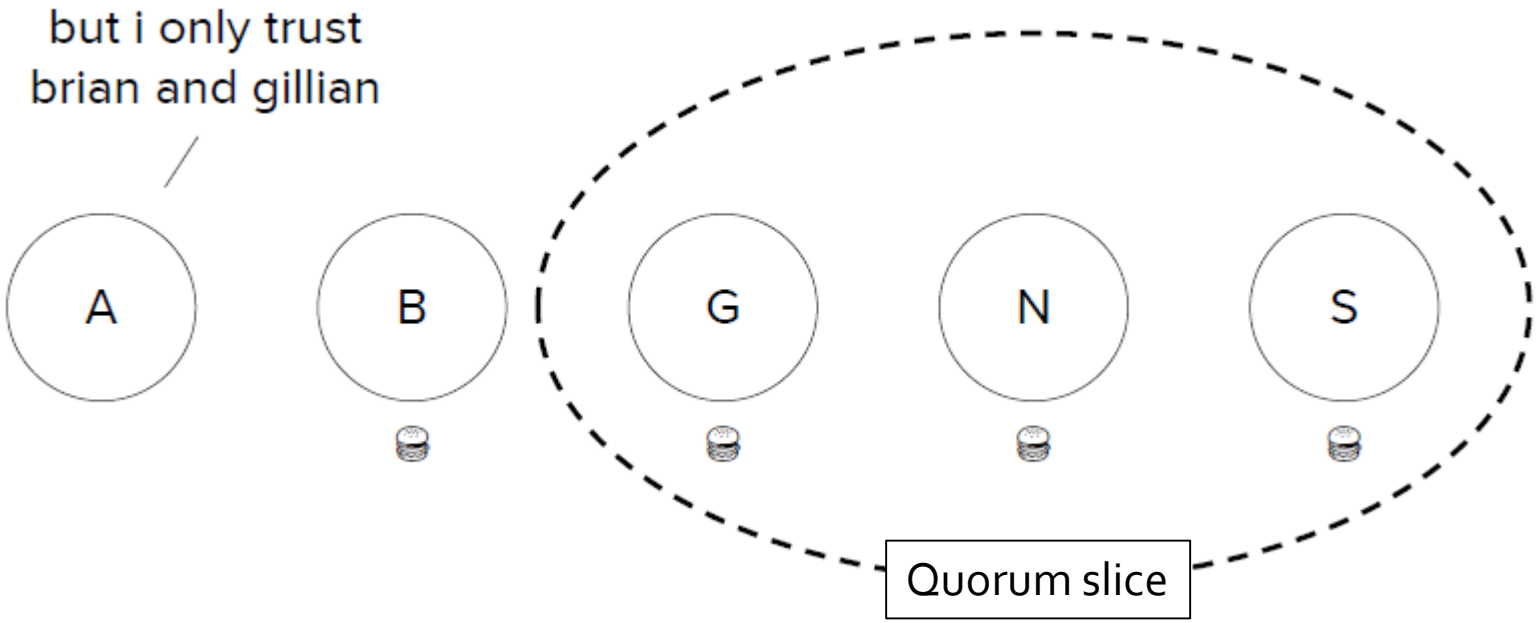
Federated Byzantine Agreement

- Lunchtime Consensus



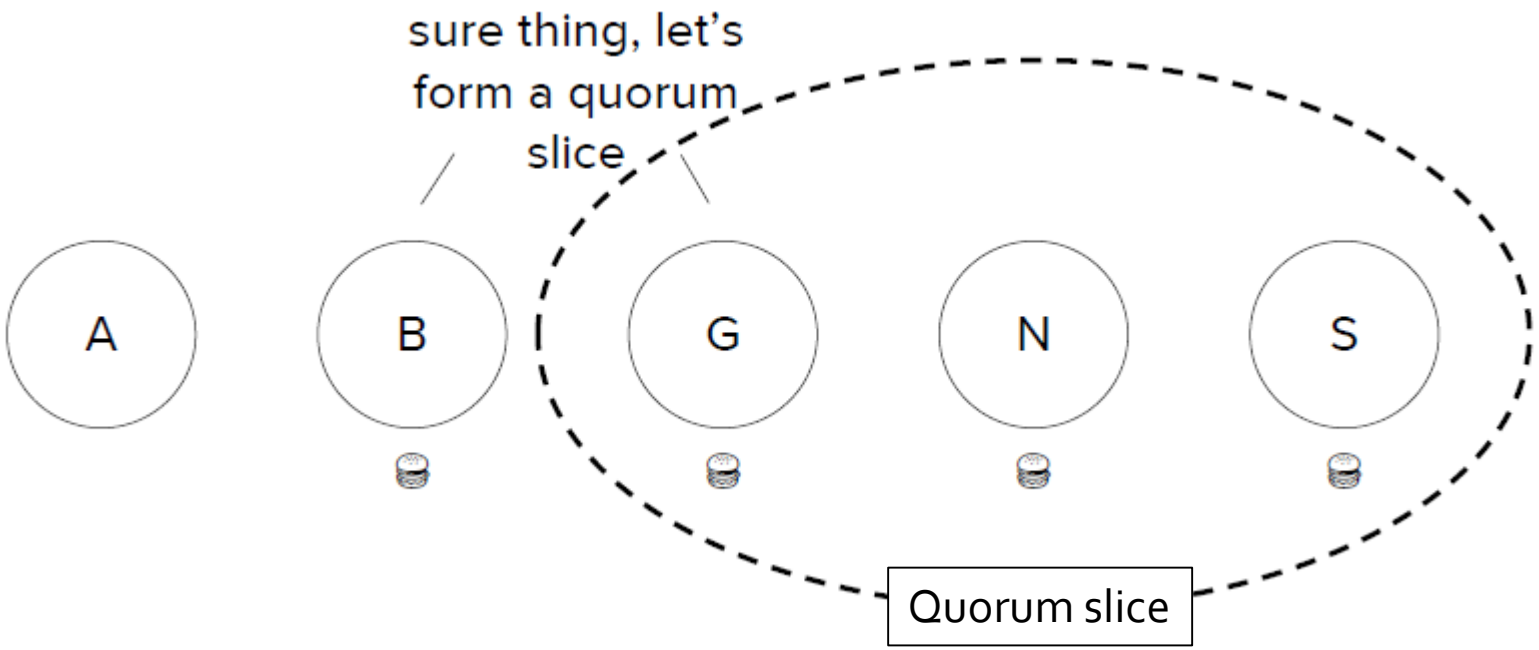
Federated Byzantine Agreement

- Lunchtime Consensus



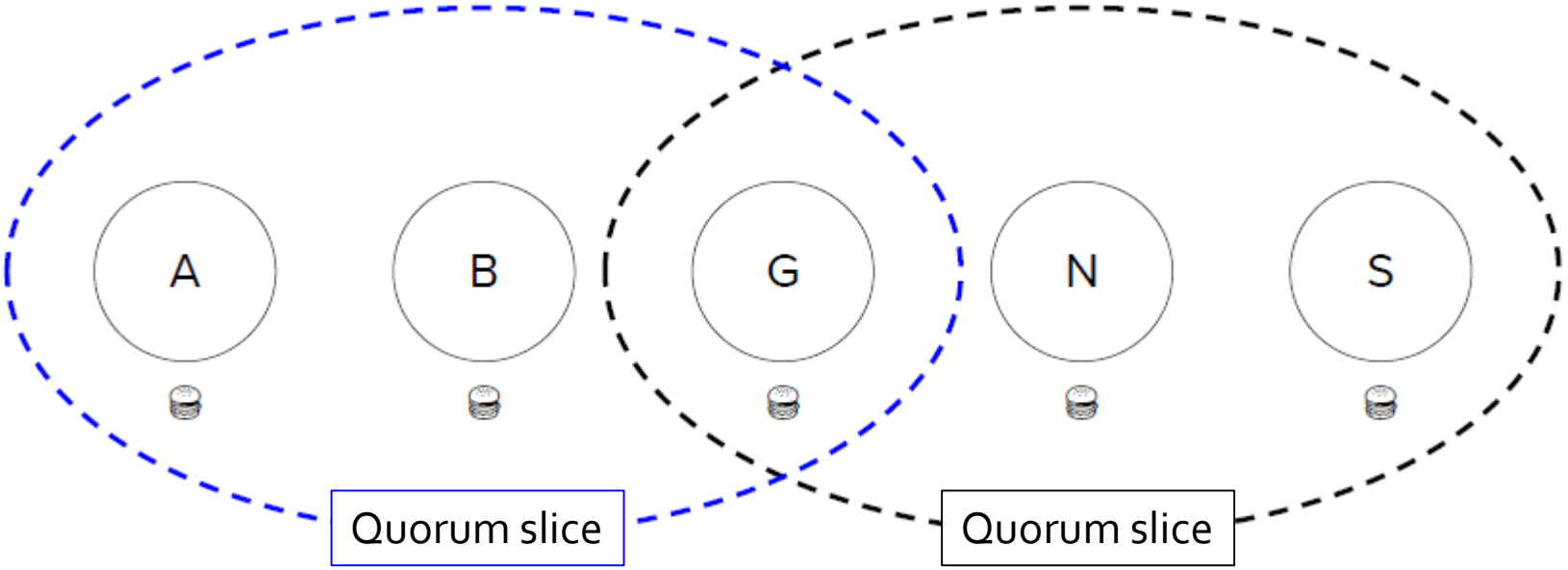
Federated Byzantine Agreement

- Lunchtime Consensus



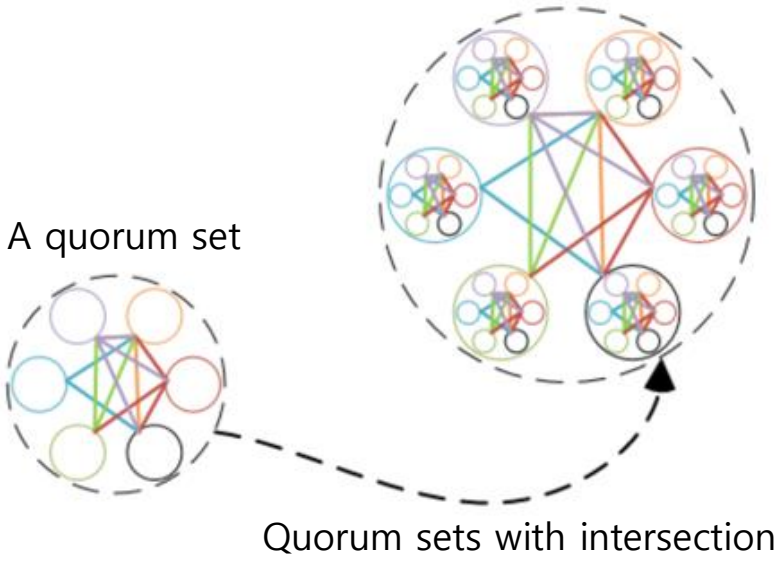
Federated Byzantine Agreement

- Lunchtime Consensus



Federated Byzantine Agreement

- Federated Byzantine Agreement (FBA) 의 가정
  - Quorum intersection은 원장 기록에서의 fork를 방지하기 위해 사용됨
  - Quorum intersection은 1개 이상의 정직하고 올바르게 작동하는 노드가 포함됨



"All else equal, bigger slices lead to **bigger quorums with greater overlap**, meaning fewer failed node sets [...] will undermine quorum intersection
...
On the other hand, **bigger slices are more likely to contain failed nodes**, endangering quorum availability"

From the white paper "Stellar Consensus Protocol"

CONTENTS

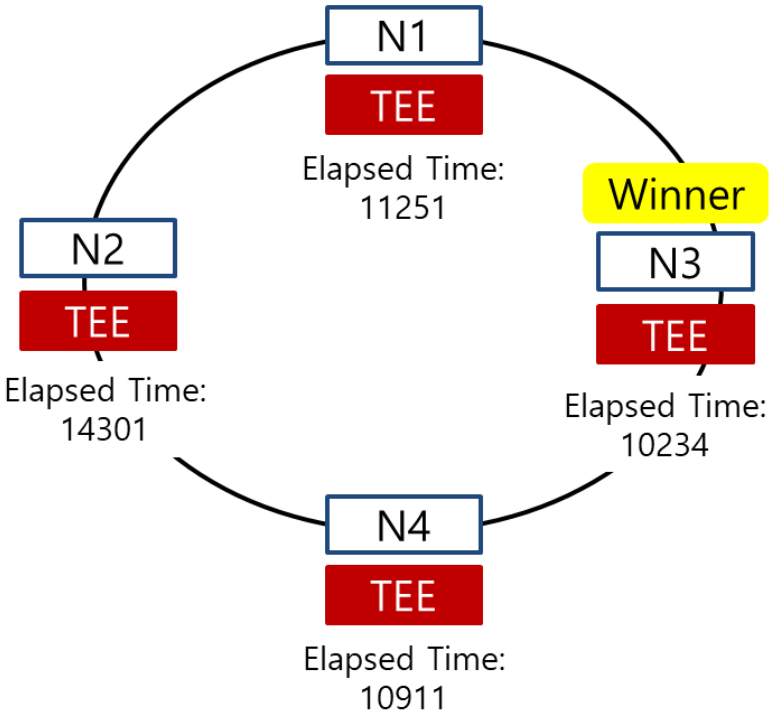
- Proof of Elapsed Time

Proof of Elapsed Time

- Proof of Elapsed Time (PoET) 의 가정
  - 모든 검증 또는 마이닝 노드들은 Intel SGX와 같은 TEE (**Trusted Execution Environments**)를 구동 시켜야함
  - PoET는 TEE로 보호되는 랜덤 리더 선출 모델 / 복권 기반 선거 모델 (Lottery based election model)을 사용
    - 각 유효성 검증자는 **TEE 내부에서 실행되는 코드로부터 대기 시간을 요청함**
    - **대기 시간이 가장 짧은 검증자가 복권에 당첨**되어 리더가 될 수 있음
    - **대기 시간 생성의 무작위성**은 리더 역할이 모든 검증 노드에 무작위로 분포되도록 함
  - 선택된 리더가 블록을 생성함

Proof of Elapsed Time

- 검증 노드가 리더라고 주장하고 블록을 생성하는 경우, 다른 노드가 쉽게 확인할 수 있는 TEE 내에서의 생성된 증거를 생성할 수 있음
  - 즉, **가장 짧은 대기 시간을 가졌다는 것을 증명**해야 함



Comparison of Consensus models

	PoW	PoS	BFT	BFA	PoET
Openness	Permissionless	Both	Permissioned	Permissionless	Both
Token needed?	Yes	Yes	No	No	No
Cost of participation	Yes	Yes	No	No	No
Scalability of peer network	High	High	Low	High	High
Trust model	Untrusted	Untrusted	Semi-trusted	Semi-trusted	Semi-trusted

(Source: Arati Baliga, "Understanding Blockchain Consensus Models ")

References

- Lecture slides from BLOCKCHAIN @ BERKELEY

- https://www.slideshare.net/vpnmentor/mining-pools-and-attacks?from_action=save

Q & A