LFT2

ICONLOOP

Abstract. 블록체인 플랫폼을 선도했던 비트코인과 이더리움은 작업 증명(Proof of work) 합의 알고리즘의 한계로 인해 낮은 처리량, 불필요한 에너지 소모, 합의 완결성 제공의 어려움 등의 문제를 가진다. 이를 극복하기 위해 PBFT 기반으로 설계된 블록체인 합의 알고리즘들이 제안되어 현실적인 블록체인 서비스를 도모하고 있다. 최근 블록체인을 활용한 실용적인 서비스들이 다수 제안되면서 보다 효율적인 블록체인 합의 알고리즘이 요구되고 있다. LFT는 PBFT를 블록체인 환경에 맞게 개선하여 설계된 경량 블록체인 합의 알고리즘이다. 블록체인의 특성을 활용하여 PBFT의 기존 3단계 합의 과정을 2단계로 단축하여 합의에 필요한 전체 메시지 전파 횟수를 감소시켰다. 그 결과 네트워크 부하가 줄어들고 신속한 블록 합의를 기대할 수 있다. 또한 합의 과정 경량화로 우려되는 보안 이슈는 Candidate/Commit Block을 도입하여 LFT는 부분 동기 네트워크에서 safety와 liveness를 보장한다.

1. 서론

2008년 11월 사토시 나카모토는 작업 증명 기반 블록체인을 도입하여 제3의 신뢰 기관(Trusted Third Party) 없이 분산 합의가 가능한 최초의 암호화폐인 비트코인(Bitcoin)[1]을 제안했다. 일반적으로 블록체인에 기록된 데이터는 위변조가 현실적으로 불가능하므로 금융 거래, 온라인 본인 인증 등 다양한 분야에서 활용될 것이라고 기대되었다. 하지만, 비트코인이 도입한 블록 합의 알고리즘인 작업증명은 노드가 막대한 컴퓨팅 자원이 필요한 문제에 대한 정답을 공개함으로써 네트워크가 유지되는데[9], 이를 위해 투자된 컴퓨팅 하드웨어와 전기 에너지 등이 낭비되었다[13,14]. 높은 에너지 낭비에도 불구하고 낮은 거래 처리 속도를 보일뿐 아니라 즉각적인 합의 완결성을 보장하지 않아 거래가 즉시 처리되어야 하는 금융 서비스 등에서 활용하기 어렵다는 한계가 있다.

이러한 작업 증명 알고리즘의 한계를 극복하기 위해, PBFT(Practical Byzantine Fault Tolerance)[2] 합의 알고리즘을 블록체인 기술에 응용한 합의 알고리즘들이 등장하기 시작했다. Tendermint[3,4,5]는 PBFT의 장애 리더 교체 알고리즘인 View Change를 제거하고 Lock을 도입하여 리더 교체를 위한 별도의 과정 없이 장애 리더를 교체하도록 설계한 특징을 가진 합의 알고리즘으로, 부분 동기 네트워크에서 safety와 liveness를 보장하며 합의 완결성을 제공하여 블록체인 플랫폼에서 동작할 수 있는 다양한 서비스의 가능성을 보였다. 그러나, PBFT 계열 알고리즘은 하나의 블록을 합의하기 위하여 많은 양의 메시지 교환이 필요하다는 한계점을 가지고 있다. 따라서, PBFT 계열 블록체인의 합의 효율을 증대시키기 위하여 합의에

필요한 메시지를 줄이는 Casper FFG[7,8], Pipelined BFT[12] 등의 합의 알고리즘이 연구되고 있다.

본 논문에서 소개하는 LFT는 블록체인의 특성을 활용하여 PBFT 합의 알고리즘에서 요구하는 메시지 전송량을 감소시킨 경량 블록체인 합의 알고리즘이다. 요약하자면, LFT는 2번의 합의 과정을 겹쳐 진행하도록 설계하여, PBFT에서 각 합의 과정마다 요구되는 2번의 메시지 전파 과정을 1번으로 줄였다. 아울러, 합의 알고리즘 경량화로 인한 보안성 감소를 방지하기 위해 Candidate/Commit Block을 도입하여 부분 동기 네트워크에서 safety와 liveness를 보장하였다.

본 논문의 2장에서는 LFT 합의 알고리즘의 규칙과 동작 프로세스에 대하여 설명하고, 3장에서는 LFT의 safety와 liveness를 증명한다.

2. LFT

LFT는 노드간 메시지 통신을 위해 gossip communication[6]을 이용하여 특정 시간 내에 전송된 메시지를 전달받을 수 있는 부분동기 네트워크를 가정한다. 또한, 블록체인 네트워크를 구성하는 노드는 블록 생성, 검증과 보유의 의무를 가지며, 자신의 메시지를 식별할 수 있는 전자 서명을 생성할 수 있다. LFT는 PBFT 합의 알고리즘을 계승하여 비잔틴 노드가 f개 존재하더라도 전체 노드가 3f + 1 이상이라면 safety와 liveness를 보장한다. 따라서, 합의에 참여하는 n개의 노드 중에 최대 비잔틴 노드 수 f를 제외한 n-f 개의 노드의 합의로 해당 블록에 정당성을 부여한다. 즉, 합의 과정에서 진행되는 투표에서 정족수 n-f 개의 투표를 받은 블록이 정당한 블록으로 인정된다.

2.1 용어

항목	정의
Round	블록이 합의되는 과정, Round의 일련번호는 진행되면서 1씩 증가하고 Node Set이 변경되면 0으로 초기화 된다.
Node Set	합의에 참여하는 모든 노드, Leader와 Validator를 포함한다.
Leader	블록을 생성하여 네트워크에 전파하는 노드를 의미한다.
Validator	Leader가 제안한 블록을 검증하는 노드를 의미한다.
Propose	Leader가 블록을 생성하여 전파하는 이벤트를 의미한다.
Vote	블록에 대한 투표를 생성하고 네트워크에 전파하는 이벤트를

	의미한다.
Candidate Block	Round가 성공적으로 완료 될 경우 해당 Round의 블록은 Candidate Block이 된다.
Commit Block	Candidate Block에 연결된 블록이 합의에 성공할 경우 Candidate Block은 Commit Block이 되어 블록체인에 저장된다.

2.2 가정 및 규칙

LFT 합의 알고리즘을 구성하는 시스템과 노드는 2.2.1에서 설명되는 가정 위에서 동작한다. 또한, 블록 합의에 참여하는 노드들은 네트워크 2.2.2의 규칙을 따른다. 만약, 규칙을 따르지 않는 노드는 비잔틴 노드로 분류되고, 비잔틴 노드는 최대 f개 존재하는 네트워크를 가정한다.

2.2.1 가정

LFT 합의 알고리즘에 참여하는 Node는 다음을 가정한다.

- 노드는 메시지를 받으면 해당 메시지를 근접한 노드들에게 특정 시간(Δ) 내로 전달한다. (gossip communication)
- Leader 선출 알고리즘은 별도로 존재하며 모든 노드는 Leader 순서를 알고 있다.
- 로컬 Timer를 가지고 있다.
- Crypto suite를 사전에 공유하고 있으며 다른 노드에 보내는 메시지에는 서명을 포함한다.

네트워크를 멈추거나 분기할 목적을 가지는 비잔틴 노드는 다음과 같은 행동을 할 수 있다.

- 비잔틴 노드는 메시지를 지연시켜 보내거나 보내지 않을 수 있다.
- 비잔틴 노드는 서로 다른 노드에 서로 다른 메시지를 보낼 수 있다.
- 비잔틴 노드는 다른 노드의 서명을 생성할 수 없다.

2.2.2 규칙

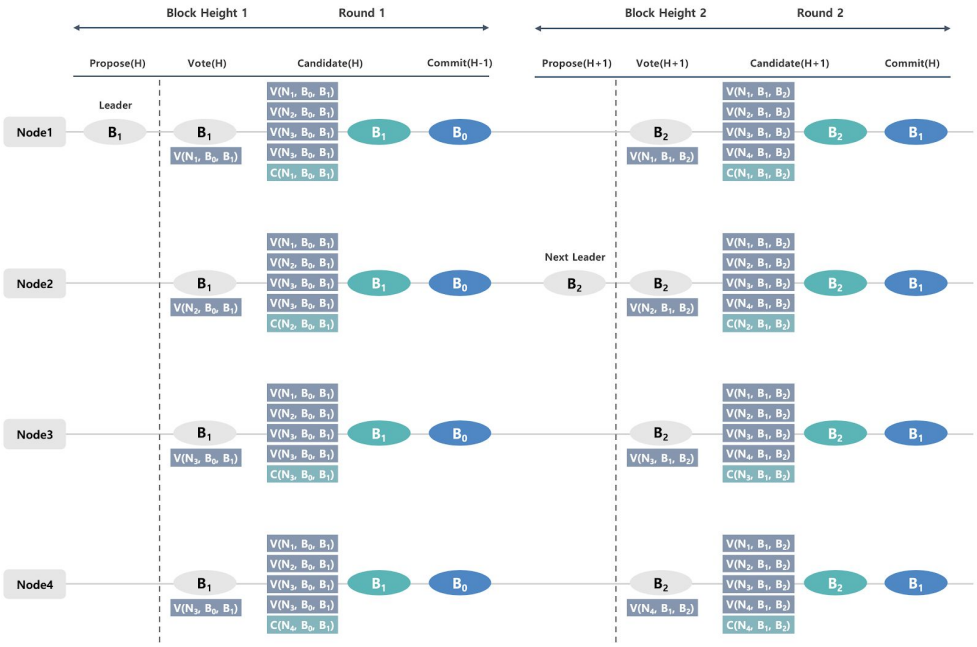
장애가 없는 Node는 다음 규칙에 따라 동작한다.

- Rule 1. Leader Election
    - Round에 따라 정해진 노드가 Leader가 된다.

    - Leader의 순서는 동일하게 알고 있다.
- Rule 2. Timer
    - ProposeTimer
        - Propose 단계가 시작되면 ProposeTimer가 동작한다.
        - Leader가 정해진 시간 내에 블록을 생성하여 전달하지 않을 경우 ProposeTimeout이 발생하여 실패 투표가 진행된다.
    - VoteTimer
        - Vote 단계 시 정족수 이상의 투표가 도착하였으나 합의되지 않았다면 VoteTimer가 시작된다.
        - 정해진 시간 내에 합의 되면 VoteTimer가 종료된다.
        - 정해진 시간 내에 합의에 도달하지 못하면 VoteTimeout이 발생한다.
- Rule 3. Propose
    - Leader는 하나의 Round에서 하나의 블록만 생성한다.
- Rule 4. Vote
    - Validator는 하나의 Round에서 Leader가 만든 하나의 블록에만 투표한다.
    - 자신의 Candidate Block에 연결된 블록에만 투표한다.
- Rule 5. Candidate
    - 자신의 Candidate Block보다 높은 Round의 블록에 대한 정족수 이상의 투표를 발견하면 해당 블록을 Candidate Block으로 변경한다.
    - 자신의 Candidate Block보다 높은 높이의 블록에 대한 정족수 이상의 투표를 발견하면 해당 블록을 Candidate Block으로 변경한다.
- Rule 6. Commit
    - 현재 Candidate Block에 연결된 블록을 새로운 Candidate Block으로 교체할 경우 이전 Candidate Block을 Commit 한다.

2.3 합의 알고리즘

본 절에서는 용이한 설명을 위해 4개의 노드로 구성된 예제를 통해 LFT 합의 알고리즘을 설명한다. <그림 1>은 LFT 합의 알고리즘을 설명하는 도식도이고, LFT의 이벤트 순서는 좌측에서 우측으로 발생한다. 그리고 수직으로 표시된 점선은 각 단계별 이벤트를 구분하며 각각의 노드는 다른 노드에게 이벤트 별 메시지를 전송한다. 약어로 표시된 내용은 다음과 같다.



<그림 1> LFT 합의 알고리즘 예시

- B : Block ID
- V(N, B', B) : Vote(Node Number, Candidate Block ID, Target Block ID)
- C(N, B', B) : Candidate(Node Number, Previous Candidate Block ID, Candidate Block ID)

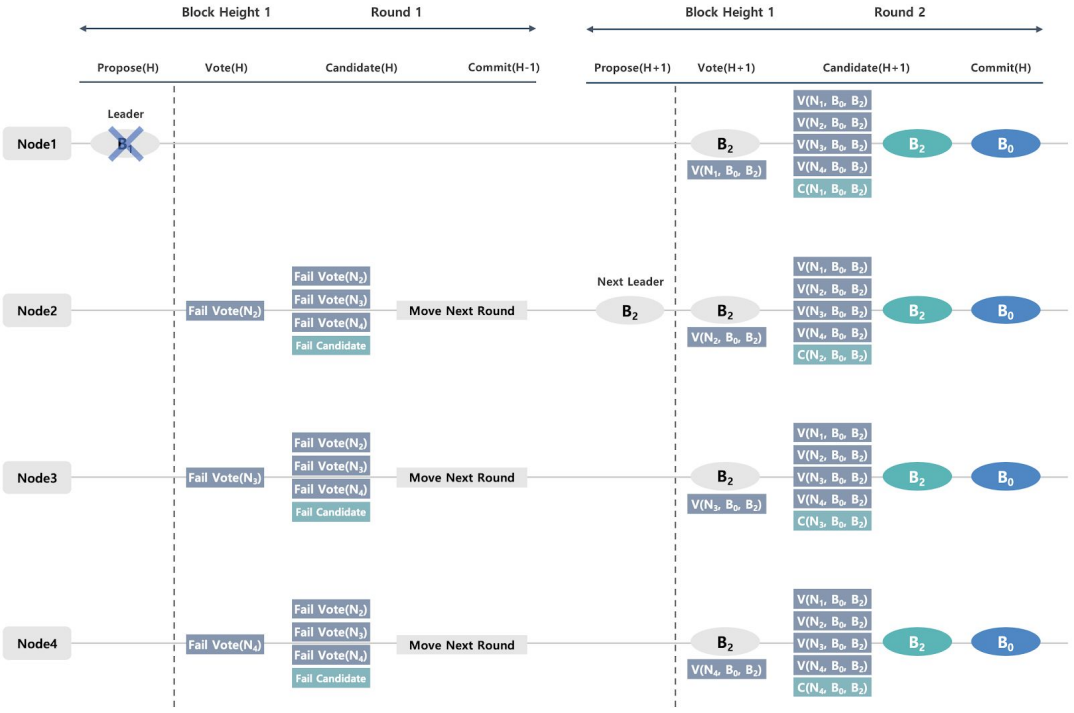
LFT가 동작하는 방식은 Round 또는 블록 높이에 따라 구분지을 수 있으며 각 단계별 발생하는 이벤트는 다음과 같다.

- Propose
  - 새로운 Round가 시작되면 ProposeTimer가 초기화되어 동작하고, 다음 블록을 생성하기 위해 정해진 Leader로 순서가 변경된다.
  - Leader는 수집한 트랜잭션과 Candidate Block의 투표를 포함하여 새로운 높이의 블록을 생성한다.
  - 생성한 블록과 전자 서명을 다른 Validator들에게 전파한다.
- Vote
  - Validator들은 Leader로부터 생성된 블록을 전달 받는다.
  - 전달 받은 블록이 정당한 Leader가 생성했는지 확인한다.
  - 블록의 높이와 이전 블록 해시가 올바른지 확인한다.
  - 전달받은 블록이 Candidate Block과 연결되어 있는지 확인한다
  - Validator들은 상기 검증 결과에 따라 투표 메시지를 생성하여 Node Set에게 전파한다.

- Candidate
  - Node Set은 자신을 제외한 Node Set으로부터 투표 메시지를 전달 받는다.
  - Node Set은 정족수 이상의 같은 투표 V(N, B', B)를 받으면 블록 B를 Candidate Block C(N, B', B)로 변경한다.
  - 만약, 정족수 이상의 투표 메시지가 도착하였으나 합의되지 않았다면 VoteTimer가 시작되고 <Rule 2. Timer>의 VoteTimer 조건에 따라 행동한다.
  - Candidate Block 정보는 다음 블록에 포함되지 않고 각 노드만 알고 있다.
- Commit
  - Node Set은 정족수 이상의 같은 투표 V(N, B', B)를 받으면 블록 B'을 Commit 한다.
  - 다음 Round의 Propose 단계로 진입한다.

2.4 Round Advance

Round Advance는 특정 Round에서 정상적으로 블록이 생성되지 않았거나, 생성된 블록에 대한 투표 내용이 상이하여 블록이 합의되지 않은 경우에도 다음 Round로 진행되는 과정을 의미한다. <그림 2>는 Round 1에서 Leader가 정상적으로 블록을 생성하지 못하여 Round 실패하더라도 다음 Round로 진행되어 결국 블록이 합의되는 과정을 나타내는 도식도다.



<그림 2> Round Advance (f = 1)

<그림 2>의 시나리오에서 각 노드별 상태 정보는 다음과 같다.

- <Round 1>
    - Node 1 (Leader) : 블록 생성 실패
    - Node 2, 3, 4 (Validator) : 실패 투표 메시지 생성 및 정상 작동
- <Round 2>
    - Node 2 (Leader) : 정상 작동
    - Node 1, 3, 4 (Validator) : 정상 작동

블록을 생성하기 위한 새로운 Round가 시작되면 ProposeTimer가 동작한다. 이때, 노드1은 Leader로써 수집한 트랜잭션을 새로운 높이의 블록 B_1을 생성하고 전자 서명과 함께 다른 Validator들에게 전송해야한다. 하지만, <그림 2>는 Leader가 장애가 발생하여 블록 B_1을 생성하지 못한 경우를 보여준다. 이 경우, ProposeTimer에 설정된 시간 내에 블록을 생성하여, 각 Validator들에게 전달하지 못하기 때문에, 각 Validator에는 ProposeTimeout이 발생한다. 따라서, Validator는 블록 B_1을 전달 받지 못했다는 실패 투표 메시지를 전파한다. 각 노드는 정족수 이상의 실패 투표 메시지를 전달 받았기 때문에 VoteTimer는 동작하지 않으며, 해당 Round에서는 블록 합의 과정이 실패라 판단하고 다음 Round로 넘어간다.

다음 Round가 시작되면 마찬가지로 ProposeTimer가 동작하고, 새로운 Round의 Leader 노드2는 정해진 시간 내에 블록 B_1을 생성하여 전자 서명과 함께 다른 Validator들에게 전한다. 노드1, 노드3, 노드4는 Leader로 부터 전달 받은 블록 B_1을 검증하고 문제가 없다고 판단되었기 때문에 정상 투표 메시지를 전파한다. 각 노드는 정족수 이상의 정상 투표 메시지를 전달 받았고 합의가 되었기 때문에 VoteTimer는 동작하지 않으며, 이전 Round의 Candidate Block인 B_0를 현재 Round의 블록 B_1으로 Candidate Block을 변경하고 블록 B_0는 Commit Block이 되어 블록체인에 저장된다.

결과적으로 Round 1에서 Leader가 블록 B_1을 정해진 시간 내에 전파하지 못하더라도 Round Advance가 발생하여 다음 Round에서 네트워크 장애 없이 블록 B_2를 합의할 수 있다.

3. 합의 알고리즘 증명

비잔틴 합의 알고리즘을 증명하기 위해서는 f개 이하의 비잔틴 노드에 의해 네트워크 분기가 발생하지 않고, 네트워크 내에서 반드시 합의가 이루어 진다는 것을 보장해야 한다[2,10,11]. 따라서, LFT가 부분 동기 네트워크에서 분기가 발생하지 않는다는 safety와 장애 없이 유지될 수 있는 liveness를 증명하고자 한다. 이를 바탕으로 LFT는 블록체인에서 한번 합의가 이루어지면 합의 결과가 변경되지 않는 완결성을 제공할 수 있다.

3.1 Safety 증명

Safety 합의 알고리즘은 블록 생성 과정에서 같은 높이의 서로 다른 블록이 합의되지 않아야 한다. LFT는 Candidate/Commit Block 규칙을 통해 safety를 보장한다.

<정리 1> B(prev_block, height)를 이전 블록이 prev_block인 높이 height의 블록이라 하자. LFT 합의 알고리즘 과정에서 같은 높이를 갖는 서로 다른 두 블록 B(B_0, H)와 B'(B_0, H)이 Commit되지 않는다.

proof) B_0에 연결된 서로 다른 두 블록 B(B_0, H)와 B'(B_0, H)가 Commit 되었다고 가정하면 <Rule 5. Candidate Block>에 의해서 B와 B'에 각각 이어진 상위 height의 두 블록 B''(B, H + 1)과 B'''(B', H + 1)이 Candidate Block이 된다.

한편, B''(B, H + 1)가 Candidate Block이 되었다면 B''의 prev_block인 B를 Candidate Block으로 설정한 노드 n - f 개 이상이었다. 마찬가지로 B'''(B', H + 1)도 Candidate Block이 되었다면 B'''의 prev_block인 B'이 Candidate Block으로 설정한 노드 수가 n - f 이상이었다.

따라서, 3f + 1개의 노드가 참여 중인 해당 네트워크에는 B를 Candidate Block으로 설정한 2f + 1개의 노드가 존재하고 B'를 Candidate Block으로 설정한 2f + 1개의 노드가 존재한다. 즉, 네트워크에는 투표를 할 수 있는 노드가 4f + 2개 존재한다. 만약 f개의 비잔틴 노드가 이중 투표를 하더라도 생성 가능한 투표 수는 3f + 1 + f = 4f + 1개이므로 B_0에 연결된 서로 다른 두 블록 B(B_0, H)와 B'(B_0, H)가 Commit 되었다는 가정에 모순이된다. 결과적으로 LFT 합의 알고리즘에서 같은 높이의 서로 다른 블록은 생성될 수 없다.

3.2 Liveness 증명

합의 알고리즘의 liveness란, 네트워크가 합의 불가능한 상태로 유지되지 않고 언젠간 합의가 계속 진행될 수 있는 성질을 의미한다. 본 절에서는 비잔틴 노드가 f개 이하인 네트워크에서 LFT가 liveness를 보장하는 것을 보여주기 위해 특정 Round가 무한 대기 상태에 빠지지 않고 완료된다는 것과 모든 Round에서 블록이 합의에 도달하지 못하는 상태로 유지되지 않음을 보인다.

<정리 2> 모든 노드가 같은 Round에 있으면 해당 Round에서 합의가 완료된다.

proof) 장애가 없는 노드는 새로운 Round에 진입하면 ProposeTimer가 동작하고 Leader의 블록 생성 여부에 상관 없이 정상 또는 실패 투표를 진행한다. 이때 정족수 이상의 투표 메시지가 전달 된다면 VoteTimer를 통하여 합의 도달 여부에 상관 없이 다음 Round로 진행 할 수 있기 때문에 f개 이하의 장애 노드가 합의에 참여하게 되더라도 해당 Round에서 합의 과정이 멈추지 않는다.

<정리 3> gossip communcation을 통한 네트워크 전파시간이 Δ라 하자. 특정 시각 t에서 장애가 없는 노드가 새로운 Round로 진입하였다면 모든 장애가 없는 노드는 t + Δ에 도달하기 전에 새로운 Round에 진입한다.

proof) 특정 시각에 t'에서 장애가 없는 노드 n이 가장 먼저 정족수 이상의 투표를 받았다고 하자. gossip communication에 의해서 t' + Δ전에 장애가 없는 모든 노드는 정족 수 이상의 투표를 받을 수 있다.

t에서 노드 n이 합의에 도달하여 다음 Round로 진행 하였다면 적어도 t + Δ에는 모든 장애가 없는 노드가 다음 Round로 진행 한다. VoteTimeout을 통하여 Round로 넘어간 시간 t가 t' + VoteTimeout이라면 장애가 없는 모든 노드는 t' + Δ에서 정족수 이상의 투표를 받았기 때문에 t + Δ 인 t' + Δ + VoteTimeout 내로 다음 Round에 진입한다.

<정리 4> gossip communcation을 통한 네트워크 전파시간이 Δ이고, 블록 생성 및 검증 시간을 0이라 가정할 때, 2·Δ < ProposeTimeout를 만족한다면, 장애가 없는 노드가 Leader인 Round에서 모든 노드는 ProposeTimeout전에 Leader가 만든 블록을 받을 수 있다.

proof) 특정 시각 t에서 가장 먼저 새로운 Round에 진입했다면, <정리 3>에 의해 t + Δ 전에는 모든 노드가 새로운 Round에 진입한다. 즉, 새로운 Round의 Leader도 t + Δ전에 새로운 Round에 진입하여 블록을 전파한다. 따라서 t + 2·Δ 전에는 모든 노드가 새로운 블록을 받을 수 있다. 2·Δ < ProposeTimeout이므로 가장 먼저 Round에 진입한 노드도 ProposeTimeout 전에 새로운 블록을 받을 수 있다.

결과적으로 <정리 4>에 의하여 장애가 없는 Leader가 블록을 생성할 경우 모든 노드가 ProposeTimeout 이벤트가 발생하기 전에 블록을 전송 받을 수 있다.

<정리 5> gossip communcation을 통한 네트워크 전파시간이 Δ이고, 블록 생성 및 검증 시간을 무시할 때, 2·Δ < ProposeTimeout, 2·Δ < VoteTimeout를 만족한다면, 장애가 없는 노드가 Leader인 Round는 합의에 도달 할 것이다.

proof) <정리 4>에서 보았듯이 모든 장애가 없는 노드는 ProposeTimeout 전에 새로운 블록을 받을 수 있다. 이때 비잔틴 노드가 VoteTimer 시간을 앞당겨서 해당 Round가 합의에 도달하지 못하도록 하려고 할 수 있다. Leader가 t에서 새로운 블록을 전파하였다면 t + Δ 전에는 모든 노드들이 새로운 블록을 받는다. 그리고 t + 2·Δ 전에는 모든 장애가 없는 노드의 투표가 장애가 없는 모든 노드에게 전파된다. f개의 비잔틴이 네트워크 딜레이가 거이 없는 노드들에게 t에 VoteTimeout을 발생 시키어도 t + VoteTimeout은 t + 2·Δ 보다 이후이기 때문에 해당 Round는 VoteTimeout이 발생하기 전에 합의에 도달한다.

<정리 5>를 통하여서 비잔틴이 아닌 노드가 Leader일 경우 해당 Round는 합의에 도달한다는 것을 보였고, <정리 2>을 통하여 Leader가 비잔틴 노드이더라도 해당 Round가 완료 되는

것을 보였다. 따라서 LFT는 비잔틴 노드가 Leader가 되어 네트워크가 잠시 멈추어도 결국엔 합의를 진행 할 수 있다.

4. 결론

블록체인은 제 3 신뢰 기관 없이 안전한 거래가 가능하게 한 기술로 공개된 네트워크에서 합의를 이루어 위변조가 불가능한 데이터 베이스를 제공한다. 블록체인의 안정성과 활용 가능성은 해당 블록체인이 어떤 합의 알고리즘을 사용하느냐에 따라 달라질 수 있다. 예를 들어 합의 완결성을 제공하는 합의 알고리즘은 선정된 대표자들만 합의에 참여하나 한 두 블록 데이터만을 이용하여 블록이 해당 블록체인에 실제로 존재하는 것을 증명할 수 있다.

LFT는 PBFT의 Commit msg 전파과정을 다음 Round의 블록 Propose와 통합하는 방향으로 설계하여 2단계 합의가 가능한 경량 합의 알고리즘이다. Validator들이 자신의 Candidate Block에 연결된 블록에만 투표하도록 설계하여 safety와 liveness를 보장한다. LFT는 합의 완결성을 제공하며, 기존의 PBFT 알고리즘의 메시지 전파 과정을 한 단계 줄였다. 메시지 전파 과정을 한 단계 줄임으로써 하나의 합의 과정에 필요한 네트워크 부하를 줄였으며 합의시간이 단축될 될 것으로 예상한다. 이러한 특성을 통하여 높은 트랜잭션 처리량을 제공할 것이다. 또한, 기존의 EOS Pipelined BFT등의 2단계 합의 알고리즘 들은 Local 시간을 정확하게 동기화 해야하는 단점이 있으나 LFT는 부분 동기 네트워크에서 동작하기 때문에 Local 시간을 정확하게 동기화 할 필요는 없다. 블록이 해당 합의 Round가 아니라 다음 Round에서 합의 되는 알고리즘 특성상 지연시간이 한 Round에서 블록이 합의 되는 알고리즘보다 늦어질 수 있다는 한계를 가지고 있다.

References

[1] S. Nakamoto, “Bitcoin: A peer-to-peer electronic cash system”, 2008.

[2] M. Castro, B. Liskov, et al., “Practical byzantine fault tolerance”, 1999.

[3] J. Kwon, “Tendermint: Consensus without mining”, 2014.

[4] E. Buchman, “Tendermint: Byzantine Fault Tolerance in the Age of Blockchains”, 2016.

[5] E. Buchman, J. Kwon, Z. Milosevic, “The latest gossip on BFT consensus”, 2018.

[6] A. Demers, D. Greene, C. Hauser, W. Irish, J. Larson, S. Shenker, H. Sturgis, D. Swinehart, and D. Terry, “Epidemic algorithms for replicated database maintenance” 1987.

[7] Vitalik Buterin and Virgil Griffith, “Casper the Friendly Finality Gadget”, 2019.

[8] Vitalik Buterin, Daniel Reijsbergen, Stefanos Leonardos, Georgios Piliouras, “Incentives in Ethereum’s Hybrid Casper Protocol”, 2019.

[9] BitFury Group, “Proof of Stake versus Proof of Work”, 2015.

[10] C. Dwork, N. Lynch, L. Stockmeyer, “Consensus in the Presence of Partial Synchrony”, 1988.

[11] C. Cachin, M. Vukolic, “Blockchain Consensus Protocols in the Wild”, 2017.

[12] Daniel Larimer, “DPOS BFT - Pipelined Byzantine Fault Tolerance”, https://medium.com/eosio/dpos-bft-pipelined-byzantine-fault-tolerance-8a0634a270ba, 2018. [13] Digiconomist, “Bitcoin energy consumption index.”, 2018.

[14] T. Swanson, “How much electricity is consumed by Bitcoin, Bitcoin Cash, Ethereum, Litecoin, and Monero?”, 2018.