

헤데라(Hedera)

해당 번역본은 이용자가 디지털자산 관련 정보를 편리하게 확인할 수 있도록
단순 참고용으로 번역된 것이며, 투자 권유를 목적으로 하지 않습니다.
그 내용이 정확하지 않을 수 있으므로 정확한 정보 습득을 위해서는
원문을 참고하시거나 원문 작성 주체 측에 문의하시기 바랍니다.
두나무는 해당 번역본 내용의 진실성 및 정확성을 보장하지 않고,
제공된 정보에 의한 투자 결과에 대하여 어떠한 책임도 지지 않으며,
투자 손실은 투자자에게 귀속됩니다.

스왈즈 해시그래프 합의 알고리즘: 공정성, 속도, 비잔틴 내결 합성

리몬 베어드(Leemon Baird)

2016년 5월 31일

수정일: 2018년 3월 18일

Swirls 기술 보고서 SWIRLDS-TR-2016-01

초록(Abstract)

본 논문에서는 비잔틴 내결합성(Byzantine fault tolerance)이 보장된 복제 상태 기계(replicated state machine)를 위한 새로운 시스템인 스왈즈 해시그래프 합의 알고리즘(Swirls hashgraph consensus algorithm)을 제안한다. 본 알고리즘은 두 거래 중 어느 쪽이 합의 순서에서 먼저 선택될지를 공격자가 조작하기 어렵다는 의미에서 공정성(fairness)을 달성한다. 이 알고리즘은 완전한 비동기성(asynchrony)을 가지며, 리더(leader)가 없고, 라운드 로빈(round robin)이나 작업증명(proof-of-work)을 사용하지 않는다. 또한, 확률 1로 수렴하는 최종적 합의(eventual consensus)와 결함이 없는 상황에서의 고속 처리를 제공한다.

이 알고리즘은 가십 프로토콜(gossip protocol)을 기반으로 하며, 참여자들이 단순히 거래에 대해 가십하는 것이 아니라, 가십에 대해 가십(gossip about gossip)한다. 이들은 모든 가십 이벤트를 반영하는 해시그래프(hashgraph)를 공동으로 구축하며, 이를 통해 가상 투표(virtual voting)를 통해 비잔틴 합의(Byzantine agreement)를 달성할 수 있다. 앨리스(Alice)는 인터넷을 통해 밥(Bob)에게 실제로 투표를 보내지 않는다. 대신 밥은 앨리스가 알고 있는 정보를 기반으로, 앨리스가 보냈을 법한 투표를 계산한다. 이를 통해 거래 자체 이외의 통신 오버헤드를 거의 추가하지 않고, 모든 거래에 대해 공정한 비잔틴 합의와 총합 순서(total order)를 달성할 수 있다.

키워드(Keywords): 비잔틴(Byzantine), 비잔틴 합의(Byzantine agreement), 비잔틴 내결합성(Byzantine fault tolerance), 복제 상태 기계(replicated state machine), 공정성(fairness), 해시그래프(hashgraph), 가십 어바웃 가십(gossip about gossip), 가상 투표(virtual voting), 스왈즈(Swirls)

1. 서론(Introduction)

분산 데이터베이스는 종종 비잔틴 내결합성(Byzantine fault tolerance)을 갖춘 복제 상태 기계로 구현되어야 한다. 일부 연구에서는 “비잔틴”이라는 용어를 약한 의미로 사용하기도 하는데, 예를 들어 공격자들이 답하지 않는다고 가정하거나 통신이 약한 비동기성을 가진다고 가정하는 경우이다 [1]. 본 논문에서는 “비잔틴”을 그 원래 정의 [2]에 따른 강한 의미로 사용한다. 즉, 전체 참여자의 3분의 1 미만까지 공격자가 될 수 있으며, 이들은 답할 수 있고, 정직한 참여자 간의 메시지를 임의로 삭제하거나 지연시킬 수 있으며, 메시지 지연에는 어떠한 상한도 존재하지 않는다. 공격자는 네트워크를 제어하여 임의의 메시지를 삭제하거나 지연시킬 수 있지만, 어느 시점에서든 정직한 참여자가 다른 참여자에게 반복적으로 메시지를 보내면, 결국 하나 이상의 메시지는 반드시 전달되어야 한다. 또한, 안전한 디지털 서명(digital signature)이 존재한다고 가정하며, 공격자가 메시지를 탐지 불가능한 방식으로 수정할 수 없다고 전제한다. 또한 충돌이 발견되지 않는 안전한 해시 함수(hash function)의 존재도 가정한다. 본 논문은 이러한 강한 정의하에서 스왈즈 해시그래프 합의 알고리즘을 제안하고, 그 비잔틴 내결합성을 증명한다.

FLP 정리(FLP theorem) [3]에 따르면, 결정론적(deterministic) 비잔틴 시스템은 메시지 지연이 무한할 수 있는 완전한 비동기 환경에서 합의를 보장할 수 없다. 그러나 비결정론적(nondeterministic) 시스템은 확률 1로 합의를 달성할 수 있다. 해시그래프 합의 알고리즘은 완전한 비동기성을 가지며, 비결정론적이고, 확률 1로 비잔틴 합의를 달성한다.

일부 시스템, 예를 들어 파코스(Paxos) [4]나 래프트(Raft) [5]는 리더(leader)를 사용한다. 그러나 이는 공격자가 현재 리더에 대해 서비스 거부 공격(denial of service attack)을 수행할 경우, 큰 지연이 발생할 수 있다는 취약점을 가진다 [6]. 많은 시스템은 단 한 명의 악의적 클라이언트로 인해 지연될 수도 있다 [7]. 실제로 후자의 논문에서는 이러한 취약점을 가진 시스템을 “비잔틴 내결합성(Byzantine fault tolerant)”이라기보다 “비잔틴 결함 생존성(Byzantine fault survivable)” 시스템으로 표현하는 것이 더 적절하다고 제안한다. 해시그래프 합의의 리더를 사용하지 않으며, 소수의 멤버에 대한 서비스 거부 공격에도 견고한(resilient) 구조를 가진다.

비트코인(Bitcoin)과 같은 다른 시스템은 작업증명(proof-of-work) 블록체인 [8]을 기반으로 한다. 이러한 방식은 앞서 언급한 문제들을 모두 회피할 수 있다. 그러나 이러한 시스템은 진정한 의미의 비잔틴 시스템이 될 수 없다. 그 이유는, 참여자가 언제 합의가 실제로 달성되었는지 확인할 수 없기 때문이다. 대신, 시간이 지남에 따라 점진적으로 상승하는 “신뢰 확률(probability of confidence)”만을 가질 뿐이다. 만약 두 개의 블록이 동시에 채굴된다면, 체인은 포크(fork)가 발생하며, 커뮤니티가 어느 가지(branch)를 확장할지 합의할 때까지 분기된 상태로 유지된다. 블록이 느리게 추가될 경우, 커뮤니티는 항상 더 긴 가지를 확장할 수 있으며, 결국 다른 가지는 생장이 멈추게 된다. 그러면 해당 가지는 “오래된(stale)” 블록으로 간주되어 잘려(prune) 제거된다. 이러한 과정은 일부 블록이 정상적으로 채굴되었음에도 불구하고 결국 폐기된다는 점에서 비효율성을 초래한다. 또한, 커뮤니티가 새로운 가지가 생겨나는 속도보다 더 빠르게 가지를 정리할 수 있도록, 블록 채굴 속도를 의도적으로 늦출 필요가 있다. 이것이 바로 작업증명의 목적이다. 작업증명에서는 채굴자들이 블록을 채굴하기 위해 어려운 연산 문제를 해결해야 하며, 이를 통해 네트워크 전체가 평균적으로 채굴 이벤트 사이에 충분히 긴 지연을 갖도록 보장한다. 해시그래프 합의 알고리즘은 일종의 블록체인과 유사하지만, “체인”이 끊임없이 분기(branching)하면서도 가지치기(pruning)가 전혀 없고, 블록이 “오래된(stale)” 상태가 되는 경우가 없으며, 채굴자가 초당 여러 개의 블록을 작업증명 없이 생성할 수 있고, 효율성은 100%에 달하는 구조라고 볼 수 있다.

작업증명 블록체인은 또한 추가적인 연산으로 인한 전력 낭비를 초래하며, 경우에 따라서는 고가의 채굴 장비를 구입해야 하는 부담도 있다. *만료된 시간 증명(proof-of-expired-time)* 시스템 [9]은 신뢰할 수 있는 하드웨어 칩을 사용하여 오랜 시간 동안 인위적으로 지연시키는 방식을 통해 이러한 전력 낭비(단, 채굴 장비 비용은 제외될 수 있음)를 피할 수 있다. 이는 마치 작업증명 연산을 수행하는 것처럼 동작한다. 그러나 이 방식은 모든 참여자가 해당 칩을 제작한 회사를 신뢰해야 한다는 전제가 필요하다. 이러한 칩 제조업체에 대한 신뢰는 일부 상황에서는 존재하지만, 다른 상황에서는 그렇지 않다. 예를 들어, FreeBSD가 안전한 난수 생성을 위해 하드웨어 RDRAND 명령어만을 의존하지 않도록 변경한 사례가 있다. 이는 “더 이상 그들을 신뢰할 수 없기 때문”이었다 [10].

비잔틴 합의(Byzantine agreement) 시스템은 위에서 언급한 문제들을 피하기 위해 별도의 합의 메커니즘을 발전시켜 왔다. 이러한 시스템들은 일반적으로 구성원들이 투표하기 위해 다수의 메시지를 교환해야 한다. n 명의 구성원이 단일 YES/NO 문제를 결정하기 위해서는, 일부 시스템에서는 $O(n)$ 개의 메시지를 네트워크를 통해 교환해야 한다. 다른 시스템에서는 이보다 더 많은 $O(n^2)$ 또는 심지어 $O(n^3)$ 개의 메시지가 필요할 수도 있다 [11]. 단일 YES/NO 결정을 위한 알고리즘은 이후 여러 거래 집합에 대한 총합 순서(total order)를 결정하는 알고리즘으로 확장될 수 있으며, 이 경우 투표 트래픽은 더욱 증가한다. 반면, 해시그래프는 네트워크를 통해 어떠한 실제 투표 메시지도 전송하지 않는다. 모든 투표가 *가상적으로(virtually)* 이루어지기 때문이다.

2. 핵심 개념(Core Concepts)

해시그래프 합의 알고리즘(hashgraph consensus algorithm)은 다음의 핵심 개념들에 기반한다.

- **거래(Transactions)** — 어떤 구성원(member)도 언제든지 서명된 거래를 생성할 수 있다. 모든 구성원은 해당 거래의 복사본을 수신하며, 커뮤니티는 이러한 거래들의 순서에 대해 비잔틴 합의(Byzantine agreement)에 도달한다.
- **공정성(Fairness)** — 소수의 공격자 그룹이 합의에서 거래의 순서에 부당한 영향을 미치기 어렵도록 설계되어야 한다.
- **가십(Gossip)** — 각 구성원이 반복적으로 무작위로 다른 구성원을 선택하여 자신이 알고 있는 모든 정보를 전달함으로써 정보가 확산된다.
- **해시그래프(Hashgraph)** — 누가 누구에게 어떤 순서로 가십했는지를 기록하는 데이터 구조이다.
- **가십 어바웃 가십(Gossip about gossip)** — 해시그래프는 가십 프로토콜을 통해 확산된다. 여기서 가십되는 정보는 가십의 이력(history of the gossip) 그 자체이므로, 이를 “가십에 대한 가십”이라 부른다. 이는 단순히 거래만을 가십하는 것에 비해 매우 적은 대역폭 오버헤드로 구현된다.
- **가상 투표(Virtual voting)** — 모든 구성원은 동일한 해시그래프의 복사본을 가지고 있으므로, 앨리스(Alice)는 밥(Bob)이 실제로 투표를 보냈다면 어떤 투표를 *보냈을 것인지를* 계산할 수 있다. 즉, 전통적인 비잔틴 합의 프로토콜처럼 실제 투표 메시지를 주고받을 필요가 없다. 각 구성원은 단 한 번의 실제 투표 전송 없이도 수많은 의사결정에 대해 비잔틴 합의에 도달할 수 있다. 해시그래프만으로 충분하며, 해시그래프 가십 이외에는 추가적인 대역폭이 전혀 사용되지 않는다.
- **유명한 증인(Famous witnesses)** — 커뮤니티가 n 개의 거래를 정렬하려면, “이벤트 x 가 이벤트 y 보다 먼저 발생했는가?”라는 형태의 $O(n \log n)$ 개의 YES/NO 질문에 대해 각각 별도의 비잔틴 합의 프로토콜을 수행할 수 있다. 하지만 훨씬 더 효율적인 접근법은 일부 이벤트(해시그래프의 정점(vertex))만을 선택하여 *증인(witness)*으로 정의하고, 해시그래프상에서 대부분의 구성원이 해당 이벤트를 생성 직후 빠르게 수신했음을 보여주는 경우 그 증인을 *유명(famous)*하다고 정의하는 것이다. 그런 다음 각 증인에 대해 “이 증인은 유명한가?”라는 단일 질문만으로 비잔틴 합의 프로토콜을 수행하면 된다. 정확한 “유명한 증인” 집합에 대한 합의가 달성되면, 해시그래프로부터 모든 이벤트에 대한 공정한 총합 순서(fair total order)를 쉽게 도출할 수 있다.
- **강한 시야(Strongly seeing)** — 해시그래프 내 임의의 두 정점 x 와 y 에 대해, x 가 y 를 강하게 본다(strongly see)는 것을 즉시 계산할 수 있다. 이는 충분한 수의 구성원을 통과하는 다중 방향 경로(directed paths)가 존재할 경우 참으로 정의된다. 이 개념은 핵심 보조정리(key lemma)를 증명하는 데 사용된다. 즉, 앨리스와 밥이 모두 동일한 질문에 대해 캐롤(Carol)의 가상 투표를 계산할 수 있다면, 두 사람은 반드시 동일한 결과를 얻는다는 것이다. 이 보조정리는 확률 1로 비잔틴 합의를 증명하는 수학적 기반을 이룬다.

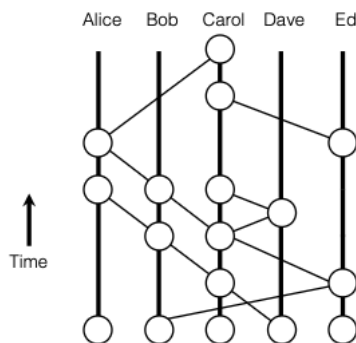


그림 1: 방향 그래프로 표현된 가십의 이력(Gossip history as a directed graph). 모든 가십 프로토콜의 이력은 그래프로 표현될 수 있으며, 각 구성원은 하나의 정점(column)으로 나타난다. 앨리스(Alice)가 밥(Bob)으로부터 가십을 수신하여 그가 알고 있는 모든 정보를 전달받는 경우, 그 가십 이벤트는 앨리스 열(column) 내의 하나의 정점(vertex)으로 표시된다. 이 정점은 앨리스와 밥의 직전 가십 이벤트로 각각 향하는 두 개의 하향 간선

3. 가십 어바웃 가십(Gossip about gossip): 해시그래프(Hashgraph)

해시그래프 합의(hashgraph consensus)는 가십 프로토콜(gossip protocol)을 기반으로 한다. 이는 앨리스(Alice)와 같은 구성원이 무작위로 다른 구성원(예: 밥(Bob))을 선택하여, 자신이 현재까지 알고 있는 모든 정보를 밥에게 전달하는 방식을 의미한다. 이후 앨리스는 또 다른 무작위 구성원을 선택해 동일한 과정을 반복한다. 밥 또한 같은 절차를 수행하며, 모든 구성원이 이 과정을 반복한다. 이처럼, 단 한 명의 구성원이 새로운 정보를 인식하면, 그 정보는 커뮤니티 전체로 지수적으로 빠르게 확산되어 결국 모든 구성원이 그 정보를 알게 된다.

모든 가십 프로토콜의 이력은 그림 1과 같은 방향 그래프(directed graph)로 표현할 수 있다. 앨리스 열의 각 정점(vertex)은 하나의 가십 이벤트를 나타낸다. 예를 들어, 앨리스 열의 가장 위쪽 정점은 밥이 앨리스에게 가십 동기화(gossip sync)를 수행하여 자신이 알고 있던 모든 정보를 전달한 사건을 나타낸다. 해당 정점은 두 개의 하향 간선을 가지며, 이는 각각 앨리스와 밥의 직전 가십 이벤트에 연결된다. 그래프에서 시간은 위쪽으로 흐르며, 아래쪽의 정점일수록 과거의 이벤트를 의미한다. 일반적인 가십 프로토콜에서는 이러한 다이어그램이 단순히 설명 목적으로 사용될 뿐, 실제로 메모리에 저장되는 그래프 구조는 존재하지 않는다.

그러나 해시그래프 합의에서는 이 그래프가 실제 데이터 구조로 존재한다. 그림 2는 이 데이터 구조를 시각화한 것이다. 각 이벤트(event), 즉 정점(vertex)은 생성자가 서명한 일련의 바이트(byte) 시퀀스로 메모리에 저장된다. 예를 들어, 앨리스(빨간색)의 한 이벤트는 밥이 가십 동기화를 수행하여 자신이 알고 있는 모든 정보를 전달했다는 사실을 기록한다. 이 이벤트는 앨리스에 의해 생성되고 그녀의 서명으로 인증되며, 두 개의 다른 이벤트(그녀의 직전 이벤트와 그 가십 직전에 존재했던 밥의 마지막 이벤트)의 해시(hash)를 포함한다. 이 빨간 이벤트는 또한 앨리스가 해당 시점에 생성하기로 선택한 거래들의 페이로드(payload)를 포함할 수 있으며, 생성 시점의 시간과 날짜를 나타내는 타임스탬프(timestamp)도 포함될 수 있다. 해당 이벤트의 다른 소상 이벤트들(회색)은 직접 포함되지 않지만, 암호학적 해시들의 집합을 통해 간접적으로 결정된다. 이러한 해시 기반 그래프 데이터 구조는 다른 목적에서도 사용될 바 있다. 예를 들어, Git에서는 정점이 파일 트리의 버전을 나타내고, 간선이 변경 사항을 나타낸다. 그러나 Git은 구성원 간 통신 이력을 기록하지 않는다. 반면, 해시그래프의 목적은 구성원들이 어떻게 통신했는지를 기록하는 것이다.

가십 프로토콜은 다양한 종류의 정보를 전송하는 데 널리 사용된다. 예를 들어, 사용자 신원(user identity)에 대한 가십, 거래(transaction)에 대한 가십, 블록체인 블록(blockchain blocks)에 대한 가십 등 다양한 정보가 있다. 그런데 만약 프로토콜이 가십에 대해 가십하도록 설계된다면 어떨까? 즉, 구성원들이 해시그래프 자체를 전송하기 위해 가십한다면 어떨까? 밥이 앨리스에게 가십을 수행할 때, 그는 자신이 알고 있고 앨리스가 아직 모르는 모든 이벤트를 그녀에게 전달하게 된다.

해시그래프를 가십함으로써 참여자들은 방대한 양의 정보를 얻게 된다. 새로운 거래가 어떤 이벤트의 페이로드에 추가되면, 그 거래는 모든 구성원에게 빠르게 확산되어 결국 전원이 이를 인식하게 된다. 앨리스는 해당 거래를 학습하게 되고, 밥이 그 거래를 언제 인식했는지도 정확히 알 수 있으며, 나아가 캐롤(Carol)이 “밥이 그 거래를 인식했다”는 사실을 언제 인식했는지도 정확히 알 수 있다. 이처럼 모든 구성원이 동일한 해시그래프 복사본을 보유하면, 이러한 깊은 추론 체인이 가능해진다. 해시그래프가 위로 확장됨에 따라, 각 구성원이 상단부의 최신 이벤트 일부에서는 약간 다른 부분집합을 가질 수 있지만, 하위 계층의 이벤트에서는 빠르게 동일한 내용으로 수렴한다. 더 나아가, 만약 앨리스와 밥이 동일한 이벤트를 모두 보유하고 있다면, 그들은 해당 이벤트의 모든 소상 이벤트 또한 반드시 동일하게 보유하고 있으며, 그 소상 하위 그래프(subgraph)의 모든 간선 구조에 대해서도 완전히 일치하는 인식을 가진다. 이러한 구조 덕분에 비잔틴 내결함성(Byzantine fault tolerance)을 포함한 강력한 알고리즘을 로컬 환경에서 수행할 수 있다.

이러한 강력한 기능은 매우 적은 통신 오버헤드로 구현된다. 커뮤니티가 단순히 서명된 거래를 가십하는 경우에도 일정한 대역폭이 필요하다. 그러나 해시그래프를 가십하는 경우, 이벤트 하나에 적어도 하나의 거래가 포함될 정도로 거래량이 충분히 많다면, 오버헤드는 극히 미미하다. 앨리스가 거래를 생성할 때 단순히 거래에 서명하는 대신, 해당 거래를 포함한 이벤트 전체에 서명하면 된다. 두 경우 모두 서명은 하나뿐이며, 거래 자체도 반드시 전송해야 한다. 추가 오버헤드는 두 개의 해시를 전송해야 한다는 점뿐이다. 그러나 이 또한 상당히 압축이 가능하다. 그림 2의 예시에서, 앨리스는 캐롤(Carol)이 이미 해당 이벤트의 모든 소상(이전 앨리스의 이벤트 또는 다른 사람과의 이전 동기화)을 가지고 있을 때만 빨간 이벤트를 전송한다. 따라서 두 개의 파란 부모 이벤트의 해시를 직접 보낼 필요가 없다. 대신, “이 이벤트는 앨리스의 다음 이벤트이며, 다른 부모는 밥의 세 번째 이벤트다”라고만 알려주면 충분하다. 적절한 압축 기법을 사용하면, 이는 매우 적은 바이트 수로 전송 가능하며, 전체 메시지 크기의 몇 퍼센트 수준만 증가시킨다.

4. 합의 알고리즘(Consensus algorithm)

모든 구성원이 모든 이벤트를 알고 있는 것만으로는 충분하지 않다. 각 이벤트의 선형적 순서(linear ordering), 즉 이벤트에 기록된 거래들의 순서에 대해서도 합의가 이루어져야 한다. 리더가 없는 대부분의 비잔틴 내결함성(Byzantine fault tolerance, BFT) 프로토콜은 구성원들이 서로에게 투표(vote)를 전송하는 방식에 의존한다. 따라서 n 명의 구성원이 단 하나의 YES/NO 질문에 합의하기 위해서는 각 구성원이 다른 모든 구성원에게 자신의 투표를 전송해야 하므로, 네트워크상에 $O(n^2)$ 개의 투표 메시지가 발생할 수 있다. 일부 프로토콜은 전송된 투표에 대한 영수증(receipt)까지 요구하므로, 이 경우 $O(n^3)$ 수준의 메시지가 필요하게 된다. 또한 이러한 프로토콜들은 여러 라운드의 투표를 수행해야 하므로, 전송되는 메시지의 수는 더욱 증가한다.

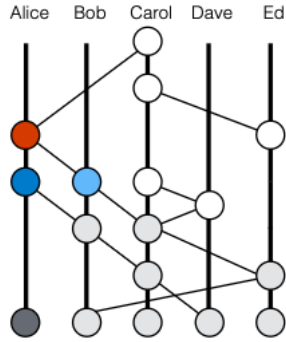


그림 2: 해시그래프 데이터 구조(Hashgraph data structure). 앨리스(Alice)는 밥(Bob)이 자신에게 가십 동기화(gossip sync)를 수행하여 모든 정보를 전달한 사실을 기록하는 이벤트(빨간색)를 생성한다. 이 이벤트는 두 개의 부모 이벤트(파란색)의 해시를 포함한다. 동일한 생성자인 앨리스의 자기 부모(self-parent, 짙은 파란색)와 밥의 다른 부모(other-parent, 연한 파란색)이다. 또한, 이 이벤트는 앨리스가 해당 시점에 생성하기로 선택한 새로운 거래(payload)와 앨리스의 디지털 서명(digital signature)을 포함한다. 나머지 조상 이벤트들(회색)은 빨간 이벤트 안에 저장되지 않지만, 해시들의 연결 관계를 통해 모두 결정된다. 자기 조상(self-ancestor, 짙은 회색)은 자기 부모 링크를 따라 도달할 수 있는 이벤트들이며, 그렇지 않은 조상들(연한 회색)은 그 외의 이벤트들이다.

해시그래프 합의(hashgraph consensus)에서는 어떠한 투표 메시지도 실제로 전송되지 않는다. 모든 구성원은 동일한 해시그래프의 복사본을 가지고 있다. 만약 앨리스와 밥이 동일한 해시그래프를 보유하고 있다면, 그들은 해당 해시그래프의 결정적 함수(deterministic function)에 따라 이벤트의 총합 순서(total order)를 계산할 수 있으며, 동일한 결과를 얻게 된다. 따라서 별도의 투표 메시지를 교환하지 않아도 합의(consensus)는 달성된다.

물론, 특정 시점에서 앨리스와 밥이 완전히 동일한 해시그래프를 가지고 있지 않을 수도 있다. 일반적으로 오래된 이벤트들에 대해서는 일치하지만, 최근의 이벤트들에서는 각자가 상대방이 아직 보지 못한 이벤트를 일부 보유하고 있을 수 있다. 또한, 간혹 커뮤니티에 새롭게 생성된 이벤트가 기존 해시그래프의 하위(더 과거) 위치에 삽입되어야 하는 경우도 발생한다. 해시그래프 합의 알고리즘은 이러한 문제를 가상 투표(virtual voting)라는 개념을 통해 해결한다.

가령, 앨리스가 해시그래프 A 를, 밥이 해시그래프 B 를 보유하고 있다고 하자. 이 두 그래프는 특정 시점에 약간 다를 수 있지만, 항상 일관성(consistency)을 유지한다. 일관성이란, A 와 B 가 동일한 이벤트 x 를 포함한다면, 두 그래프 모두 x 의 동일한 조상 집합(ancestors set)과 동일한 조상 간 간선(edges) 구조를 가진다는 것을 의미한다. 만약 앨리스는 x 를 알고 있고 밥은 모르는 상태이며, 두 사람 모두 정직하고(genuine) 적극적으로 참여 중이라면, 밥은 가십 프로토콜을 통해 곧 x 를 알게 될 것으로 예상된다. 합의 알고리즘은 이러한 전달이 언젠가는 일어날 것이라고 가정하지만, 그 속도에 대해서는 아무런 가정을 하지 않는다. 즉, 본 프로토콜은 완전히 비동기적(asynchronous)이며, 타임아웃(timeout)이나 가십 속도, 진행률 등에 대한 어떠한 전제도 두지 않는다.

앨리스는 해시그래프 A 내의 이벤트들에 대해 일련의 선거(election)를 계산함으로써 총합 순서(total order)를 도출한다. 각 선거에서는 A 내의 일부 이벤트가 투표를 발신하는 주체로 간주되고, 일부 이벤트가 투표를 수신하는 대상으로 간주된다. 앨리스는 여러 개의 선거를 계산하며, 특정 이벤트는 어떤 선거에는 참여하고 다른 선거에는 참여하지 않을 수 있다. 또한 동일한 이벤트가 서로 다른 선거에서 상이한 방식으로 투표할 수도 있다.

예를 들어, 해당 이벤트가 밥(Bob)에 의해 생성된 경우, 우리는 “밥이 특정 방식으로 투표했다”고 말할 수 있다. 그러나 실제 밥이라는 구성원은 이 과정에 전혀 관여하지 않는다. 이는 오직 앨리스가 로컬 환경에서 수행하는 계산에 불과하며, 실제 밥이 인터넷을 통해 투표 메시지를 전송하지 않아도 앨리스가 “밥이 보냈을 법한 투표”를 계산하는 것이다.

이러한 가상 투표(virtual voting)는 여러 가지 이점을 가진다. 우선, 네트워크 대역폭을 절약할 뿐 아니라, 모든 구성원이 규칙에 따라 일관되게 투표를 계산하도록 보장한다. 앨리스(Alice)가 정직한 구성원이라면, 그녀는 “가상의 밥(virtual Bob)”이 정직하게 투표하도록 계산한다. 실제 밥이 부정행위를 하더라도, 밥이 가상 밥을 이용해 앨리스를 공격하거나 잘못된 투표를 하도록 만들 수는 없다.

그러나 밥은 다른 방식으로 부정행위를 시도할 수 있다. 예를 들어, 밥이 자신의 이전 이벤트 z 를 자기 부모(self-parent)로 하는 해시를 포함한 이벤트 x 를 생성했다고 하자. 이후 그는 또 다른 이벤트 y 를 생성하지만, 이번에는 자기 부모 해시를 x 로 지정하지 않고, 다시 z 로 지정한다. 즉, y 는 x 이후에 생성되었음에도 불구하고 z 를 부모로 지정한 것이다. 이렇게 되면 해시그래프 내에서 밥의 이벤트들은 연속된 체인(chain)이 아닌 트리(tree) 구조를 이루게 되며, 밥은 포크(fork)를 생성한 셈이 된다. 만약 밥이 이벤트 x 를 앨리스에게, 이벤트 y 를 캐롤(Carol)에게 각각 가십한다면, 한동안 앨리스와 캐롤은 포크가 존재한다는 사실을 인식하지 못할 수 있다. 이때 앨리스는 x 에 대해 가상 투표를 계산하고, 캐롤은 y 에 대해 다른 방식의 가상 투표를 계산할 수 있다. 따라서 포크가 존재하더라도, 서로 일관된 해시그래프들 간에 그 포크가 일정 기간 동안 확산될 가능성이 존재한다. 즉, 앨리스의 해시그래프에는 x 가 있지만 y 는 없고, 캐롤의 해시그래프에는 y 는 있지만 x 는 없는 상황이 발생할 수 있으며, 이때 두 사람 모두 이것이 포크라는 사실을 아직 인식하지 못한다.

해시그래프 합의 알고리즘은 이러한 공격을 시야(seeing)와 강한 시야(strongly seeing) 개념을 사용하여 방지한다. 이 개념은 조상(ancestor)과 자기 조상(self-ancestor)의 정의에 기반하며, 모든 이벤트는 자기 자신을 조상이자 자기 조상으로 간주한다.

만약 밥이 두 이벤트 x 와 y 를 생성했는데, 둘 중 어느 것도 상대방의 자기 조상이 아니라면, 이는 밥이 포크를 생성한 부정행위가 된다. 어떤 이벤트 w 가 x 를 조상으로 가지고 있지만 y 를 조상으로 가지고 있지 않다면, w 는 이벤트 x 를 볼 수 있다(see)고 한다. 그러나 w 의 조상에 x 와 y 가 모두 포함되어 있다면, w 는 둘 중 어느 것도, 그리고 동일한 생성자에 의해 만들어진 다른 어떤 이벤트도 볼 수 없는 것으로 정의된다. 즉, w 는 x 를 알고 있으며, 해당 생성자에 의해 발생한 포크를 알지 못할 때만 x 를 “본다”고 할 수 있다.

n 명의 구성원($n > 1$)이 있을 때, 이벤트 w 가 이벤트 x 를 *강하게 본다(strongly see)* 고 하려면, w 가 x 를 볼 수 있는 서로 다른 구성원들의 이벤트 중 3분의 2를 초과하는(즉, $2n/3$ 보다 많은) 수의 이벤트를 볼 수 있어야 한다. 이 개념은 그림 3에 시각적으로 표현되어 있다. 동일한 해시그래프의 네 복사본이 표시되어 있으며, 각 그림에서 하단의 오렌지색 이벤트가 서로 다르다. (d)에서는 상단의 노란색 이벤트가 서로 다른 구성원들의 4개의 빨간색 이벤트를 볼 수 있으며, 이 빨간색 이벤트 각각은 하단의 오렌지색 이벤트를 볼 수 있다. 이는 (a), (b), (c)에서도 마찬가지이며, 특히 (a)에서는 5개의 빨간색 이벤트가 존재한다. 하지만 이 예시에서는 $n = 5$ 이므로, $2n/3$ 보다 큰 최소 정수는 4이기 때문에, 4개의 이벤트만으로도 *강한 시야* 조건을 만족한다.

이 개념은 실제 투표 없이, 로컬 환경에서의 가상 투표만으로도 비잔틴 내결함성(Byzantine fault tolerance)을 달성할 수 있게 한다.

가상 투표에서는, 이벤트 x 가 어떤 YES/NO 질문(예: 특정 이벤트가 “유명한(famous)”가 여부)에 대해 투표한다고 가정할 때, 그 투표는 오직 x 의 조상 집합(ancestors set)에 기반한 함수로 계산된다. 그리고 해당 투표는 오직 이벤트 w 가 x 를 *강하게 볼 수 있을 때만* x 에서 w 로 전송된 것으로 간주된다. 5장에서 증명되듯이, 만약 x 와 y 가 불법 포크(illegal fork)의 서로 다른 가지(branch)에 존재한다면, w 는 x 와 y 중 최대 하나만을 강하게 볼 수 있으며, 두 개를 동시에 강하게 볼 수는 없다. 또한, 해시그래프 A 와 B 가 일관적(consistency)을 유지한다면, A 에서는 어떤 이벤트가 x 를 강하게 보고, B 에서는 또 다른 이벤트가 y 를 강하게 보는 상황이 발생할 수 없다. 이 보조정리(lemma)는 비잔틴 합의 증명의 핵심이다. 즉, 공격자가 포크를 만들어 부정행위를 시도하더라도, 서로 다른 구성원들이 상이한 합의 순서에 도달하도록 만들 수 없음을 보장한다. 과거의 일부 비잔틴 합의 알고리즘은, 앨리스(Alice)가 밥(Bob)과 캐롤(Carol)에게 상충된 투표를 보낼 가능성에 대비하여, 구성원들이 수신한 각 투표에 대해 모든 다른 구성원에게 “영수증(receipt)”을 전송하도록 요구했다. 이러한 공격과 방어 메커니즘은 해시그래프에서의 포킹 공격(forking attack)과 유사한 측면이 있으며, “영수증”의 개념은 해시그래프에서의 “강한 시야(strongly seeing)” 개념과 유사한 역할을 수행한다.

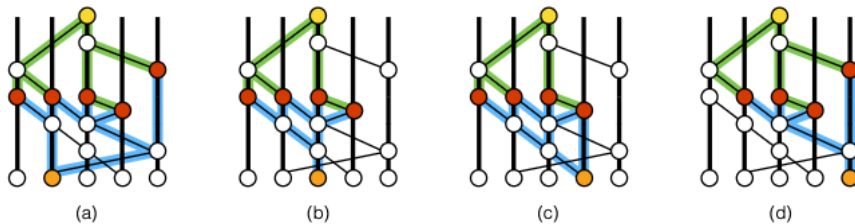


그림 3: 강한 시야(Strongly seeing)의 개념. 각 해시그래프에서 상단의 노란색 이벤트는 하단 행의 주황색 이벤트 중 하나를 강하게 볼(strongly see) 수 있다. 구성원 수는 $n = 5$ 이므로, $2n/3$ 보다 큰 최소 정수는 4이다. (d)에서 상단의 노란색 이벤트는 서로 다른 생성자에 의해 만들어진 4개의 중간 빨간색 이벤트 각각의 조상에 하나의 주황색 이벤트가 존재한다. 따라서 노란색 이벤트는 해당 주황색 이벤트를 강하게 본다. 다른 해시그래프들 역시 동일한 구조를 가지며, 각각 다른 주황색 이벤트를 대상으로, 노란색 이벤트가 최소 4개의 빨간색 이벤트를 통해 해당 이벤트를 본다. 모든 주황색 이벤트와 노란색 이벤트의 두 부모 이벤트가 생성 라운드 r 에 속한다면, 노란색 이벤트는 서로 다른 구성원들에 의해 라운드 r 에서 생성된 $2n/3$ 개 이상의 증인(witness)을 강하게 볼 수 있으므로, 라운드 $r + 1$ 에서 생성된 것으로 간주된다. 모든 이벤트는 자기 자신을 조상이자 자기 조상(self-ancestor)으로 정의한다는 점에 유의한다.

이러한 정의에 기반하여, 전체 해시그래프 합의 프로토콜은 그림 4, 5, 6, 7의 알고리즘으로 구성된다.

그림 4의 주요 알고리즘은 통신 방식이 매우 단순함을 보여준다. 앨리스(Alice)는 무작위로 다른 구성원 밥(Bob)을 선택하고, 자신이 알고 있는 모든 이벤트를 밥에게 가십(gossip)한다. 밥은 그 가십이 발생한 사실을 기록하기 위해 새로운 이벤트를 생성한다.

이 단순한 가십 프로토콜만으로도 비잔틴 내결함성(Byzantine Fault Tolerance, BFT)과 정합성(correctness)이 보장된다. 그러나 효율성을 높이기 위해 여러 확장이 가능하다. 예를 들어, 앨리스와 밥이 서로 어떤 이벤트를 이미 알고 있는지 먼저 교환한 뒤, 앨리스가 밥이 아직 모르는 이벤트들만 전송할 수도 있다. 또한, 프로토콜에서는 앨리스가 이벤트를 전송할 때 항상 위상 순서(topological order)를 따르도록 요구할 수 있다. 즉, 밥이 이벤트를 수신하기 전에 그 이벤트의 부모를 이미 알고 있도록 보장하는 것이다. 심지어 앨리스가 밥에게 동기화(sync)를 완료한 직후, 밥이 곧바로 앨리스에게 다시 동기화를 수행하도록 정의할 수도 있다. 여러 개의 동기화가 동시에 일어날 수도 있으므로, 앨리스가 여러 구성원에게 동기화하는 동시에 여러 구성원이 앨리스에게 동기화할 수 있다. 이러한 다양한 최적화 방안들이 존재하지만, 기본적인 이 단순 프로토콜만으로도 충분하다.

각 동기화가 완료된 후, 각 구성원은 가능한 많은 이벤트에 대해 합의 순서를 결정하기 위해 세 가지 절차를 호출한다. 이 과정에는 어떠한 통신도 필요하지 않으며, 오직 로컬 계산만으로 충분하다. 각 절차의 for 루프는 이벤트를 위상 순서로 탐색한다. 즉, 이벤트의 부모가 항상 먼저 탐색된다. 알고리즘의 첫 번째 for 루프에서, 만약 x 가 역사상 첫 번째 이벤트라면 부모나 이전 라운드가 없으므로 $x.\text{round} = 1$ 로 설정되고 $x.\text{witness} = \text{TRUE}$ 로 설정되어야 한다. 알고리즘은 또한 전체 참여자 수를 나타내는 상수 n ($n > 1$)과, 2보다 큰 작은 정수 상수 c (예: $c = 10$)를 사용한다. 이 알고리즘에서 비잔틴 합의는 확률 1로 보장된다.

각 이벤트에 대해 조상(ancestors)의 함수로서 라운드 번호(round number)를 정의하는 것은 매우 유용하다. **divideRounds** (그림 5)에서는 모든 알려진 이벤트에 대해 그 조상들의 라운드 번호를 기반으로 한 정수형 *라운드 번호*를 할당한다(정의 5.2). 그림 3의 해시그래프는 이 과정이 어떻게 수행되는지를 보여준다. 만약 하단 행의 모든 이벤트가 라운드 r 이라면, 노란색 이벤트를 제외한 모든 이벤트는 라운드 r 에 속하게 된다. 노란색 이벤트는 라운드 $r + 1$ 로 진전되는데, 이는 라운드 r 의 $2n/3$ 개 이상의 이벤트를 강하게 볼 수 있기 때문이다. 역사상 첫 번째 이벤트는 라운드 1로 정의되며, 이후의 모든 라운드는 이로부터 결정된다. 각 이벤트는 결국 *생성 라운드(round created)*와 *수신 라운드(round received)*를 모두 갖게 된다. 생성 라운드는 흔히 *라운드* 또는 *라운드 번호*라고도 불린다.

각 구성원에게 있어, 매 라운드에서 생성하는 첫 번째 이벤트를 *증인(witness)*이라 한다. 오직 이러한 증인 이벤트만이 가상 투표를 송수신하며, 이는 그림 6에 제시된 **decideFame** 절차에서 수행된다. 이 절차가 바로 비잔틴 합의가 이루어지는 단계이다. 각 증인에 대해, 해당 증인이 *유명(famous)*한지를 결정한다. 한 증인은 다음 라운드의 다수 증인들이 그 볼 수 있다면 유명한 것이고, 그렇지 않다면 유명하지 않은 것으로 간주된다. 비잔틴 합의 프로토콜은 각 증인마다 하나의 선거(election)를 실행하여 그 증인이 유명하지 여부를 결정한다. 라운드 r 의 증인 x 에 대해, 라운드 $r + 1$ 의 각 증인은 x 를 볼 수 있다면 “유명하다”에 투표한다. 만약 $2n/3$ 을 초과하는 다수의 증인들이 동일한 의견을 보인다면, 커뮤니티는 합의에 도달한 것이며, 해당 선거는 종료된다. 반대로 투표가 균형을 이루면, 필요할 만큼 여러 라운드 동안 투표가 계속된다. 이때 각 증인은 이전 라운드에서 자신이 강하게 볼 수 있는 다수의 증인들의 의견을 따라 투표한다. 공격자가 인터넷을 제어하는 상황에 대비하기 위해, 프로토콜은 주기적으로 *코인 라운드(coin round)*를 도입하여 증인들이 의사난수(pseudorandom) 방식으로 투표할 수 있게 한다. 즉, 공격자가 네트워크 내 모든 메시지를 조작하여 투표가 교묘히 받

로 갈리도록 하더라도, 커뮤니티가 우연히 $2n/3$ 임계치를 넘길 확률이 존재한다. 따라서 결국 확률 1로 합의가 이루어진다.

그림 6에서, 알고리즘의 “if $d = 1$ ” 구문을 “if $d = 2$ ”로 바꾸더라도 알고리즘은 여전히 정상적으로 작동한다. 이렇게 수정된 알고리즘에서는 각 선거(election)가 한 라운드 늦게 시작된다. 나아가, 두 접근법을 결합한 하이브리드 알고리즘도 유효하다. 즉, 매 라운드마다 먼저 “ $d = 1$ ” 검사를 적용해 모든 선거를 수행한다. 만약 해당 라운드의 모든 증인(witness)에 대한 유명 여부(fame)가 결정되었고, 그 라운드에서 유명한 증인을 생성한 구성원이 $2n/3$ 이하라면, 그 라운드의 선거들만 다시 수행하되 이번에는 “ $d = 2$ ” 검사를 적용한다. 이 하이브리드 알고리즘에서도 본 논문의 모든 정리(theorem)는 여전히 유효하며, 비잔틴 내결함성(Byzantine Fault Tolerance, BFT) 증명 또한 그대로 유지된다. 이 새로운 선거가 트리거되는 라운드의 경우, 합의에 도달하기까지의 시간이 다소 증가할 수 있으며(약 20% 정도), 이는 이론적으로 성능에 약간의 영향을 줄 수 있다. 그러나 실제로는 이러한 상황이 매우 드물게 발생하며, 오히려 발생 시에는 공정성을 보장하기 위해 유명한 증인의 수를 증가시킬 가능성이 있다.

특정 라운드의 각 증인이 유명하지 여부에 대한 합의가 이루어지면, 이를 활용하여 이전 이벤트들에 대한 합의 타임스탬프(consensus timestamp)와 합의 총합 순서(consensus total order)를 쉽게 결정할 수 있다. 이 과정은 그림 7의 **findOrder** 절차에서 수행된다.

먼저, 수신 라운드(received round)가 계산된다. 이벤트 x 가 라운드 r 의 수신 라운드를 가진다는 것은, 라운드 r 이 모든 고유 유명 증인(unique famous witnesses)이 x 의 자손(descendant)이 된 최초의 라운드이며, r 이하의 모든 라운드에서 모든 증인의 유명 여부가 이미 결정되었음을 의미한다.

(라운드 내의 고유 유명 증인 집합(unique famous witnesses)은 “유명 증인(famous witnesses)” 집합에서 한 구성원이 동일 라운드에서 두 개 이상의 유명 증인을 생성한 경우, 그 구성원의 모든 유명 증인을 제외한 것으로 정의된다.)

그 다음, 수신 시각(received time)이 계산된다. 예를 들어, 이벤트 x 의 수신 라운드가 r 이고, 앨리스(Alice)가 라운드 r 에서 고유 유명 증인 y 를 생성했다고 하자. 알고리즘은 x 를 인식한 y 의 가장 이른 자기 조상(self-ancestor) 이벤트 z 를 찾는다. 그리고 z 를 생성할 때 앨리스가 내부에 기록한 타임스탬프를 t 라 하면, t 는 “앨리스가 x 를 처음 인식했다고 주장하는 시각”으로 간주된다. 이벤트 x 의 수신 시각은 라운드 r 에서 고유 유명 증인을 생성한 모든 구성원의 이러한 타임스탬프들의 중앙값(median)으로 정의된다.

이후, 합의 순서(consensus order)가 계산된다. 모든 이벤트는 먼저 수신 라운드(received round) 기준으로 정렬된다. 두 이벤트가 동일한 수신 라운드를 가진다면, 수신 시각(received time)을 기준으로 정렬된다. 그럼에도 불구하고 순서가 완전히 결정되지 않는 경우(동일한 시각 등), 각 이벤트의 서명(signature)을 기준으로 정렬하되, 해당 서명은 동일 수신 라운드의 모든 고유 유명 증인들의 서명과 XOR 연산을 수행하여 ‘화이트닝(whitening)’된 형태로 비교된다.

5. 비잔틴 내결함성 증명(Proof of Byzantine fault tolerance)

이 절에서는 여러 유용한 정의를 제시하고, 이후 일련의 증명을 통해 강한 시야 보조정리(Strongly Seeing Lemma, 정리 5.12)에서 시작하여 비잔틴 내결함성 정리(Byzantine Fault Tolerance Theorem, 정리 5.19)에 이르는 과정을 단계적으로 구성한다. 이 증명에서는 구성원 수가 n 명($n > 1$)이라고 가정하며, 그중 $2n/3$ 초과는 정직한(honest) 구성원이고, $n/3$ 미만은 비정직한 구성원이라고 가정한다. 또한 디지털 서명(digital signature)과 암호학적 해시(cryptographic hash)가 안전하다고 가정한다. 즉, 서명을 위조할 수 없으며, 서명된 메시지를 탐지되지 않은 채 변경할 수 없고, 해시 충돌이 발생할 가능성도 없다. 가십 동기화(syncing gossip) 프로토콜에 대해서는 다음을 가정한다. 앨리스(Alice)가 밥(Bob)에게 자신이 알고 있는 모든 이벤트를 전송할 때, 밥은 서명이 유효하고, 그가 이미 보유한 부모 이벤트에 해당하는 유효한 해시를 포함한 이벤트만 수락한다. 시스템은 완전히 비동기적(asynchronous)이다. 모든 정직한 구성원 쌍(예: 앨리스와 밥)에 대해, 앨리스는 언젠가는 밥과 동기화를 시도하며, 반복적으로 메시지 전송을 시도할 경우 결국 메시지 하나는 반드시 전달된다고 가정한다. 네트워크의 신뢰성, 속도, 또는 타임아웃(timeout)에 대한 다른 가정은 전혀 하지 않는다. 공격자는 네트워크를 완전히 제어할 수 있으며, 메시지를 임의로 삭제하거나 지연시킬 수 있다. 다만, 정직한 구성원 간에 반복적으로 전송된 메시지는 결국 한 번 이상 반드시 전달되어야 한다는 제약만 존재한다.

정의 5.1(Definition 5.1). 이벤트 x 가 이벤트 y 의 조상(ancestor)으로 정의되는 경우는 다음과 같다: x 가 y 자신이거나, y 의 부모(parent), 혹은 y 의 부모의 부모, 그 이상의 상위 부모인 경우. 또한 x 가 y 자신이거나, y 의 자기 부모(self-parent), 혹은 자기 부모의 자기 부모인 경우, x 는 y 의 자기 조상(self-ancestor)으로 정의된다.

정의 5.2(Definition 5.2). 이벤트 x 의 생성 라운드 번호(round created number) 또는 라운드(round)는 다음과 같이 정의된다. x 의 부모 이벤트들 중 최대 라운드 번호를 r 이라 하고 (부모가 없으면 $r = 1$ 이라 한다), x 가 라운드 r 에 속한 $2n/3$ 초과의 증인(witness)을 강하게 볼(strongly see) 수 있다면 $i = 1$, 그렇지 않으면 $i = 0$ 으로 정의한다. 따라서 x 의 라운드 번호는 $r + i$ 가 된다.

```
run two loops in parallel:
  loop
    sync all known events to a random member
  end loop
  loop
    receive a sync
    create a new event
    call divideRounds
    call decideFame
    call findOrder
  end loop
```

그림 4: 스왈즈(Swirls) 해시그래프 합의 알고리즘. 각 구성원은 무작위로 선택된 다른 구성원들과 반복적으로 통신(sync)한다. 송신 동기화와 수신 동기화는 병렬로 이루어진다. 앨리스가 밥에게 동기화를 수행할 때, 앨리스는 밥이 아직 보지 못한 모든 이벤트를 전송한다. 밥은 이 이벤트들을 해시그래프에 추가하되, 서명이 유효하고 부모 이벤트의 해시가 자신의 그래프에 존재하는 이벤트만 수락한다. 이후, 모든 알려진 이벤트들은 라운드로 구분된다. 각 라운드에서 각 구성원의 첫 번째 이벤트(즉, “증인”)에 대해, 가상 투표(virtual voting)를 통한 순수 로컬 비잔틴 합의로 해당 증인이 “유명(famous)”한지 여부가 결정된다. 그런 다음, 충분한 정보가 확보된 이벤트들에 대해 총합 순서(total order)가 결정된다. 만약 두 구성원이 독립적으로 동일한 이벤트의 역사적 위치를 계산했다면, 그들은 항상 동일한 위치를 할당하며, 이후 더 많은 정보가 추가되더라도 그 위치는 변경되지 않는다. 또한, 모든 이벤트는 결국 확률 1로 이러한 위치를 부여받게 된다.

```

procedure divideRounds
  for each event  $x$ 
     $r \leftarrow \text{max round of parents of } x \text{ (or 1 if none exist)}$ 
    if  $x$  can strongly see more than  $2n/3$  round  $r$  witnesses
       $x.\text{round} \leftarrow r + 1$ 
    else
       $x.\text{round} \leftarrow r$ 
     $x.\text{witness} \leftarrow (x \text{ has no self parent})$ 
      or  $(x.\text{round} > x.\text{selfParent.round})$ 

```

그림 5: divideRounds 절차.

이벤트 x 가 새로 알려지면, 즉시 라운드 번호 $x.\text{round}$ 이 할당되고, 불리언 값 $x.\text{witness}$ 가 계산된다. 이는 해당 이벤트가 “증인(witness)”인지, 즉 그 구성원이 해당 라운드에서 생성한 첫 번째 이벤트인지를 나타낸다.

```

procedure decideFame
  for each event  $x$  in order from earlier rounds to later
     $x.\text{famous} \leftarrow \text{UNDECIDED}$ 
  for each event  $y$  in order from earlier rounds to later
    if  $x.\text{witness}$  and  $y.\text{witness}$  and  $y.\text{round} > x.\text{round}$ 
       $d \leftarrow y.\text{round} - x.\text{round}$ 
       $s \leftarrow \text{the set of witness events in round } y.\text{round} - 1 \text{ that } y \text{ can strongly see}$ 
       $v \leftarrow \text{majority vote in } s \text{ (is TRUE for a tie)}$ 
       $t \leftarrow \text{number of events in } s \text{ with a vote of } v$ 
      if  $d = 1$  // first round of the election
         $y.\text{vote} \leftarrow \text{can } y \text{ see } x?$ 
      else
        if  $d \bmod c > 0$  // this is a normal round
          if  $t > 2 * n/3$  // if supermajority, then decide
             $x.\text{famous} \leftarrow v$ 
             $y.\text{vote} \leftarrow v$ 
            break out of the  $y$  loop
          else // else, just vote
             $y.\text{vote} \leftarrow v$ 
        else // this is a coin round
          if  $t > 2 * n/3$  // if supermajority, then vote
             $y.\text{vote} \leftarrow v$ 
          else // else flip a coin
             $y.\text{vote} \leftarrow \text{middle bit of } y.\text{signature}$ 

```

그림 6: decideFame 절차. 각 증인 이벤트(즉, $x.\text{witness} = \text{TRUE}$ 인 이벤트)에 대해, 해당 이벤트가 유명(famous) 한지를 결정한다. 이 결정은 가상 투표(virtual voting)에 기반한 비잔틴 합의 프로토콜을 통해 수행된다. 각 구성원은 자신이 보유한 해시그래프 복사본을 사용하여 로컬에서 이 절차를 실행하며, 추가적인 통신은 전혀 필요하지 않다. 알고리즘은 해시그래프 내의 이벤트들이 서로 투표를 주고받는 것처럼 계산하지만, 실제로는 모든 계산이 개별 구성원의 컴퓨터 내에서만 수행된다. 각 구성원은 여러 라운드에 걸쳐 각 증인에게 투표를 할당하며, 인구의 3분의 2(즉, $> 2n/3$) 이상이 동일한 결론에 도달할 때까지 반복된다. 이벤트 x 의 유명 여부를 결정하기 위해, 해시그래프가 확장될 때마다 이 절차가 반복적으로 실행되어, $x.\text{famous}$ 에 최종적인 값이 할당된다.

정의 5.3(Definition 5.3). 이벤트 x 의 수신 라운드 번호(round received number) 또는 수신 라운드(round received)는, 모든 고유 유명 증인(unique famous witnesses)이 x 의 자손(descendant)이 되는 최초의 라운드로 정의된다.

```

procedure findOrder

```

```

for each event  $x$ 
  if there is a round  $r$  such that there is no event  $y$ 
    in or before round  $r$  that has  $y.$  witness =  $TRUE$ 
    and  $y.$  famous =  $UNDECIDED$ 
    and  $x$  is an ancestor of every round  $r$  unique famous witness
    and this is not true of any round earlier than  $r$ 
  then
     $x.$  roundReceived  $\leftarrow r$ 
     $s \leftarrow$  set of each event  $z$  such that  $z$  is
      a self-ancestor of a round  $r$  unique famous witness,
      and  $x$  is an ancestor of  $z$  but not of the self-parent of  $z$ 
     $x.$  consensusTimestamp  $\leftarrow$  median of the timestamps of all the events in  $s$ 

return all events that have roundReceived  $\neq UNDECIDED$ ,
  sorted by roundReceived, then ties sorted by consensusTimestamp,
  then by whitened signature

```

그림 7: findOrder 절차. 라운드 r 의 모든 증인들에 대한 유명 여부가 결정되면, 해당 라운드의 유명 증인 집합을 구한다. 이때 동일한 생성자가 여러 개의 유명 증인을 생성했다면, 해당 구성원의 유명 증인들은 모두 제거한다. 남은 증인들이 고유 유명 증인(unique famous witnesses)이다. 이 고유 유명 증인들은 판사(judges)의 역할을 하여, 이전 이벤트들에 대해 수신 라운드(received round)와 합의 타임스탬프(consensus timestamp)를 부여한다. 이벤트 x 는, 모든 고유 유명 증인들이 x 를 수신한 최초의 라운드에서 “수신(received)”된 것으로 간주된다. 단, 이전의 모든 라운드에서 모든 증인의 유명 여부가 이미 결정되어 있어야 한다. x 의 타임스탬프는 각 구성원이 x 를 처음 수신했다고 주장한 시간들의 중앙값(median)으로 정의된다. 이후 모든 이벤트는 수신 라운드 순으로 정렬되며, 동일한 라운드 내에서는 합의 타임스탬프 순으로 정렬된다. 여전히 동물이 남는 경우, 이벤트의 서명을 기준으로 추가 정렬을 수행하되, 서명은 해당 수신 라운드의 모든 고유 유명 증인 서명과 XOR 연산을 수행하여 화이트닝(whitening) 처리된 형태로 비교한다.

정의 5.4(Definition 5.4). 이벤트 쌍 (x, y) 가 동일한 생성자(creator)를 가지며, 둘 중 어느 것도 서로의 자기 조상(self-ancestor)이 아닐 경우, (x, y) 를 포크(fork)라고 한다.

정의 5.5(Definition 5.5). 정직한 구성원(honest member)은 모든 다른 구성원과 무한히 자주 동기화(sync)를 시도하며, 각 동기화 후에 최신 자기 부모(self-parent) 및 다른 부모(other-parent)의 해시를 포함하는 유효한 이벤트를 생성하고, 서로 포크 관계에 있는 두 이벤트를 결코 생성하지 않는다.

정의 5.6(Definition 5.6). 이벤트 x 가 이벤트 y 를 볼 수 있다(see)고 하는 것은, y 가 x 의 조상(ancestor)이며, x 의 조상 집합에 y 의 생성자에 의한 포크가 포함되어 있지 않은 경우를 의미한다.

정의 5.7(Definition 5.7). 이벤트 x 가 이벤트 y 를 강하게 본다(strongly see)고 하는 것은, x 가 y 를 볼 수 있고, $2n/3$ 을 초과하는 구성원들의 이벤트 집합 S 가 존재하여, x 가 S 의 모든 이벤트를 볼 수 있으며, S 의 모든 이벤트가 y 를 볼 수 있는 경우를 의미한다.

정의 5.8(Definition 5.8). 증인(witness)은 구성원이 특정 라운드에서 생성한 첫 번째 이벤트를 의미한다.

정의 5.9(Definition 5.9). 유명한 증인(famous witness)은 커뮤니티가 본 알고리즘을 사용하여 “유명하다(famous)”고 결정한 증인이다. 비공식적으로, 커뮤니티는 대부분의 구성원이 다음 라운드 시작 시점까지 해당 증인을 인식하는 경우, 그 증인을 유명하다고 판단한다. 고유한 유명 증인(unique famous witness)은 동일한 라운드에서 동일한 생성자가 만든 다른 유명 증인이 존재하지 않는 유명 증인을 의미한다. 포크가 존재하지 않는 한, 모든 유명 증인은 동시에 고유한 유명 증인이기도 하다.

정의 5.10(Definition 5.10). 두 해시그래프 A 와 B 가 일관적(consistent)이라고 하는 것은, 두 그래프 모두에 포함된 임의의 이벤트 x 에 대해 동일한 조상 집합을 포함하고, 그 조상들 사이의 부모(parent) 및 자기 부모(self-parent) 간선이 동일한 경우를 말한다.

보조정리 5.11(Lemma 5.11). 모든 구성원은 일관된 해시그래프를 가진다.

증명: 만약 두 구성원이 이벤트 x 를 포함한 해시그래프를 보유하고 있다면, 두 그래프 모두 x 에 포함된 동일한 두 개의 해시를 포함하고 있다. 구성원은 해당 이벤트의 부모를 모두 보유하지 않은 상태에서는 해당 이벤트를 수락하지 않으므로, 두 해시그래프는 반드시 x 의 두 부모 이벤트를 모두 포함해야 한다. 암호학적 해시가 안전하다고 가정하므로, 두 부모 이벤트는 동일해야 한다. 귀납적으로, x 의 모든 조상도 동일해야 한다. 따라서 두 해시그래프는 일관성을 가진다. ■

강한 시야(strongly seeing) 개념의 목적은 다음 보조정리를 성립시키기 위함이다. 이 보조정리는 전체 증명의 기초가 된다. 이 정리가 있어야만 투표 결과의 일관성이 보장되며, 실제 투표 메시지를 교환하지 않아도 모든 구성원이 동일한 결과를 계산할 수 있음을 증명할 수 있다.

보조정리 5.12 (Strongly Seeing Lemma). 만약 이벤트 쌍 (x, y) 가 포크(fork)이고, 해시그래프 A 에서 이벤트 z 가 x 를 강하게 본다면, A 와 일관적인 어떤 해시그래프 B 에서도 y 를 강하게 보는 이벤트는 존재하지 않는다.

증명: 모순에 의한 증명을 사용한다. 만약 B 내의 이벤트 w 가 y 를 강하게 본다고 가정하자. 강한 시야의 정의에 따라, A 내에는 z 가 볼 수 있는 이벤트들의 집합 S_A 가 존재하며, S_A 의 모든 이벤트는 x 를 볼 수 있다. 마찬가지로, B 내에는 w 가 볼 수 있는 이벤트들의 집합 S_B 가 존재하며, S_B 의

모든 이벤트는 y 를 본다. S_A 는 $2n/3$ 을 초과하는 구성원들의 이벤트로 구성되어야 하며, S_B 도 마찬가지이다. 따라서 두 집합은 $n/3$ 초과와 구성원들이 생성한 이벤트를 공유해야 한다. $n/3$ 미만의 구성원만이 비정적하다고 가정하므로, 두 집합 모두에 이벤트를 생성한 정직한 구성원 m 이 적어도 한 명 존재해야 한다. 이 구성원이 S_A 내에서 생성한 이벤트를 q_A , S_B 내에서 생성한 이벤트를 q_B 라 하자. m 이 정직하므로, q_A 와 q_B 는 포크 관계일 수 없으며, 둘 중 하나는 다른 하나의 자기 조상(self-ancestor)이어야 한다. 일반성을 잃지 않고, q_A 가 q_B 의 자기 조상이라고 하자. 해시그래프 A 와 B 는 일관적이므로, q_B 가 B 에 포함되어 있다면, q_B 의 조상인 q_A 역시 B 에 포함되어야 한다. 따라서 B 내에서 x 는 q_A 의 조상이며, q_A 는 q_B 의 조상이므로, x 는 q_B 의 조상이다. 또한 y 역시 q_B 의 조상이다. 즉, x 와 y 는 서로 포크 관계이면서 q_B 의 조상이 된다. 그렇다면 q_B 는 x 나 y 중 어느 것도 볼 수 없으며, 동일한 생성자의 포크를 인식한 상태가 된다. 그러나 이는 q_B 가 y 를 볼 수 있다는 가정에 모순된다. 따라서 이 모순을 통해 정리가 증명된다. ■

모든 시점에서 모든 구성원은 일관된 해시그래프를 가진다. 두 해시그래프가 일관적이라면, 두 그래프에 모두 포함된 이벤트 x 는 동일한 조상 집합을 가진다. 따라서 x 의 조상에 의해 결정되는 모든 속성에 대해 두 그래프는 동일한 결과를 갖게 된다. 이러한 속성에는 생성 라운드(round created), 증인 여부(witness), 어떤 이벤트를 볼 수 있는지(seeing), 강하게 볼 수 있는지(strongly seeing), 그리고 (증인인 경우) 각 선거에서 어떤 방식으로 투표할지를 포함한다. 대부분의 이러한 속성에 대해서는 정의상 즉시 성립하지만, 다음 보조정리는 생성 라운드에 대해서도 동일함을 증명한다.

보조정리 5.13(Lemma 5.13). 만약 해시그래프 A 와 B 가 일관적이며, 둘 다 이벤트 x 를 포함한다면, 두 그래프는 x 에 동일한 생성 라운드 번호(round created number)를 할당한다.

증명: 두 일관된 해시그래프가 모두 이벤트 x 를 포함한다면, 두 그래프는 x 의 모든 조상 집합(역사상 첫 번째 이벤트 포함)을 동일하게 가진다. 귀납법을 사용하여 증명할 수 있다. 두 그래프는 첫 번째 이벤트의 라운드 번호가 1이라는 점에 대해 일치한다. 그리고 임의의 이벤트 y 에 대해, 그 조상들의 라운드 번호가 동일하다면, 두 그래프는 y 의 부모 이벤트들의 최대 라운드 번호 r 에 대해 동일한 값을 갖는다. 또한 y 가 서로 다른 구성원들에 의해 라운드 r 에서 생성된 $2n/3$ 초과와 증인을 강하게 볼 수 있는지 여부에 대해서도 두 그래프는 동일하게 판단한다. 따라서 y 의 라운드 번호 역시 동일하게 계산된다. 그러므로 x 를 포함한 두 그래프가 공유하는 모든 이벤트의 라운드 번호는 일치한다. ■

라운드 r 의 증인 x 가 라운드 $r - 1$ 의 증인 y 를 강하게 볼 수 있다면, 임의의 선거(election)에 대해 y 가 x 에게 투표를 “보낸다(send)”고 하고, x 가 y 로부터 투표를 “받는다(collect)”고 표현한다. 이는 순전히 가상 투표(virtual voting)이며, 해시그래프 자체를 통해 계산된다. 서로 다른 구성원들은 약간씩 다른 해시그래프를 보유할 수 있으므로, 개별 선거 과정도 다소 차이가 있을 수 있다. 그러나 모든 투표는 일관성을 유지한다. 즉, 한 해시그래프가 “앨리스(Alice)가 밥(Bob)에게 특정 라운드의 특정 선거에서 특정한 방식의 투표를 보냈다”고 나타낸다면, 일관적인 다른 해시그래프는 동일한 투표를 보여주거나, 혹은 해당 투표 자체가 존재하지 않을 뿐이다. 두 일관적인 해시그래프가 동일한 라운드에서 앨리스의 서로 다른 투표 결과를 보여주는 것은 불가능하다. 이는 다음 보조정리에서 증명된다.

보조정리 5.14(Lemma 5.14). 만약 해시그래프 A 와 B 가 일관적이고, A 에서 실행 중인 알고리즘이 특정 선거에 대해 구성원 m_0 의 라운드 r 중 인이 구성원 m_1 의 라운드 $r + 1$ 증인에게 투표 v_A 를 보냈다고 나타내며, 동시에 B 의 알고리즘이 같은 선거에서 구성원 m_0 의 라운드 r 증인이 구성원 m_1 의 라운드 $r + 1$ 증인에게 투표 v_B 를 보냈다고 나타낸다면, $v_A = v_B$ 이다.

증명: 알고리즘은 이벤트 x 가 이벤트 y 를 강하게 볼 수 있는 경우에만 x 에서 y 로 투표를 전송한 것으로 간주한다. 강한 시야 보조정리(Strongly Seeing Lemma, 보조정리 5.12)에 따르면, 일관적인 두 해시그래프가 동시에 포크(fork) 관계에 있는 두 이벤트를 모두 강하게 보는 것은 불가능하다. 따라서 두 투표는 동일한 이벤트 x 로부터 발생한 것이다. 이벤트의 투표는 그 조상 집합의 함수로서만 계산되므로, 두 해시그래프는 동일한 투표 결과를 산출하게 되며, $v_A = v_B$ 가 성립한다. ■

비잔틴 합의(Byzantine agreement)는 특정 YES/NO 질문에 대해 여러 라운드의 가상 투표를 통해 달성된다. 구성원은 일반 라운드(coin 라운드)가 아닌 라운드)에서, 라운드 $r + 1$ 의 이벤트가 라운드 r 에서 동일한 방향으로 투표한 구성원의 수가 $2n/3$ 을 초과하는 것을 강하게 볼 수 있다면, 라운드 r 에서 선거 계산을 종료한다. 이 조건이 충족되면, 다른 모든 구성원들도 라운드 r , $r + 1$, 혹은 (coin 라운드의 경우) $r + 2$ 안에서 동일한 결론에 도달한다. 즉, 다음 보조정리는 만약 한 구성원이 특정 YES/NO 질문에 대해 결정을 내렸다면, 나머지 모든 구성원들도 거의 즉시 동일한 방향으로 비잔틴 합의에 도달함을 증명한다.

보조정리 5.15(Lemma 5.15). 만약 해시그래프 A 와 B 가 일관적이며, A 가 라운드 r 에서 결과 v 로 비잔틴 합의 선거(Byzantine agreement election)를 결정하고, B 가 r 이전에는 결정을 내리지 않은 경우, B 는 라운드 $r + 2$ 이내에 동일한 결과 v 로 결정을 내린다.

증명: 결정은 코인 라운드(coin round)에서는 발생할 수 없으므로, r 은 일반 라운드(regular round)여야 한다. A 가 투표 결과 v 를 결정했다는 것은, 라운드 r 의 어떤 증인이 $2n/3$ 을 초과하는 구성원 집합 S 로부터 동일한 투표 v 를 받았음을 의미한다. 이전 보조정리에 의해, 투표는 일관성이 유지되므로, A 와 B 의 다른 모든 라운드 r 이벤트들도 $2n/3$ 을 초과하는 구성원들로부터 투표를 받게 되며, 이 구성원들의 다수는 S 에 속한다. 즉, 크기가 $2n/3$ 을 초과하는 두 부분집합은 반드시 서로 절반 이상 겹치는 구성원을 공유하게 되므로, A 와 B 의 모든 라운드 r 증인은 동일한 방향으로 v 에 투표하게 된다(일부는 이 시점에서 이미 결정을 내릴 수 있다). 만약 라운드 $r + 1$ 이 일반 라운드라면, A 와 B 의 모든 이벤트는 일치된 투표 v 를 수신하고, 모두 v 로 결정을 내린다.

반면, 라운드 $r + 1$ 이 코인 라운드라면, 모든 이벤트는 동일하게 v 투표를 수신하므로 코인을 던지지 않고 모두 v 로 투표하며, 이 경우 라운드 $r + 2$ 에서 모두 v 로 결정을 내리게 된다. ■

다음 정리는 단일 YES/NO 질문에 대해 비잔틴 내결합성이 달성됨을 보여준다.

정리 5.16(Theorem 5.16). 어떤 단일 YES/NO 질문에 대해서도, 합의는 결국 확률 1로 달성된다.

증명: 어떤 구성원이 해당 질문에 대한 결정을 내렸다면, 앞선 보조정리에 의해 모든 구성원은 2라운드 이내에 동일한 방향으로 결정을 내린다. 따라서 합의가 실패할 유일한 경우는, 어느 구성원도 결정을 내리지 못하는 경우, 즉 어떤 증인도 $2n/3$ 을 초과하는 일치된 투표를 받지 못하는 경우이다. 그러나 코인 라운드에서는, 이러한 초다수(supermajority)가 아직 형성되지 않았을 때, 모든 정직한 구성원이 무작위로 투표를 선택하므로, 이들이 모두 동일한 투표를 선택할 비영(非0) 확률이 존재한다. 코인 라운드는 주기적으로 무한히 반복되므로, 결국 정직한 구성원들이 동일한 투표에 도달할 확률은 1이며, 그 이후 2라운드 이내에 합의가 이루어진다. ■

해시그래프 합의 알고리즘에서는, 비잔틴 합의가 각 라운드의 증인(witness)이 유명(famous) 한지를 결정하는 데 사용된다. 다음 보조정리에 의해, 모든 라운드에는 반드시 하나 이상의 유명한 증인이 존재함이 보장된다.

보조정리 5.17(Lemma 5.17). 임의의 라운드 번호 r 에 대해, 라운드 $r + 3$ 에 적어도 하나의 이벤트가 존재하는 해시그래프에서는, 라운드 r 내에 적어도 하나의 증인이 합의 알고리즘에 의해 유명한 것으로 결정되며, 이 결정은 모든 라운드 $r + 3$ 의 증인에 의해(또는 그 이전에) 이루어진다.

증명

라운드 $r + 3$ 에 적어도 하나의 증인이 존재한다고 하자. 이때, 그 중 하나의 증인을 포함하는 집합을 S_{r+3} 라 정의한다. 모든 $i < r + 3$ 에 대해, S_i 를 라운드 $i + 1$ 의 적어도 한 증인에 의해 강하게 보이는(strongly seen) 모든 라운드 i 의 증인 집합으로 정의한다. 라운드 $i + 1$ 의 이벤트가 존재한다는 것은, 라운드 i 에서 $2n/3$ 을 초과하는 증인들이 강하게 보였음을 의미하므로, 모든 $i < r + 2$ 에 대해 $2n/3 < |S_i| \leq n$ 이 성립한다. 또한, 강한 시야 보조정리(Strongly Seeing Lemma, 보조정리 5.12)에 의해, 각 구성원은 동일한 라운드에서 둘 이상의 증인을 생성할 수 없으므로, 이러한 조건은 유지된다. 강하게 본다는 것은 곧 본다는(seeing) 것을 의미하므로, 라운드 $r + 1$ 의 각 이벤트는 라운드 r 의 이벤트 중 $2/3$ 초과를 본다. 따라서 평균적으로, 라운드 r 의 각 이벤트는 라운드 $r + 1$ 의 이벤트 중 $2/3$ 초과에 의해 관찰된다. 모든 이벤트가 평균 이하일 수는 없으므로, 라운드 r 의 이벤트 중 하나(이를 x 라 하자)는 라운드 $r + 1$ 의 $2/3$ 초과 이벤트들에 의해 관찰된다. 그 결과, 라운드 $r + 1$ 의 $2/3$ 초과 이벤트들이 “ x 가 유명하다”는 YES 투표를 하게 된다. 따라서 라운드 $r + 2$ 의 모든 이벤트는 x 의 유명 여부에 대해 YES가 다수임을 확인하고, 이에 따라 x 가 유명하다고 투표한다(이 시점에서 결정을 내릴 수도 있고 아닐 수도 있다). 결국 라운드 $r + 3$ 의 이벤트는 x 의 유명 여부에 대한 전원 일치(unanimous) 투표를 수신하게 되며, 이로 인해 x 를 유명하다고 결정한다. 따라서 라운드 $r + 3$ 에 이벤트를 가진 모든 구성원은 라운드 $r + 2$ 또는 $r + 3$ 에서 최초로 x 가 유명하다고 결정한다. ■

보조정리 5.18(Lemma 5.18). 해시그래프 A 가 이벤트 x 를 포함하지 않지만 x 의 모든 부모를 포함하고, B 가 A 에 x 를 추가하여 얻어진 결과라고 하자. 또한 x 가 라운드 r 에서 생성된 증인(witness)이고, A 가 라운드 r 내의 증인 중 하나 이상의 유명 여부(fame)가 이미 결정된 상태(유명 혹은 비유명)라면, B 에서는 x 가 “유명하지 않음(not famous)”으로 결정된다.

증명 A 에서 라운드 r 의 증인 중 하나의 유명 여부를 이미 결정한 증인을 w 라 하자. A 에는 x 가 존재하지 않으므로, w 의 어떠한 조상도 x 를 볼 수 없다. 또한 일관성(consistency)에 의해 B 에서도 동일한 조상 구조를 가지므로, w 의 조상들은 B 에서도 x 를 볼 수 없다. 따라서 w 의 조상 중 라운드 $r + 1$ 의 증인들은 모두 B 에서 x 의 유명 여부에 대해 NO로 투표하게 된다. 결과적으로, 라운드 $r + 2$ 의 w 의 조상 이벤트는 B 에서 x 가 유명하지 않다고 결정한다. ■

앞선 세 가지 보조정리와 정리로부터, 모든 라운드는 결국 모든 증인들이 “유명” 또는 “유명하지 않음”으로 분류되며, 이 과정은 전체 커뮤니티의 합의에 의해 결정됨을 알 수 있다. 또한, 각 라운드에는 최소 하나 이상의 유명한 증인이 존재한다. 그 이후에는, 해당 라운드의 유명한 증인 집합이 새로운 이벤트가 추가되더라도 절대 변경되지 않는다. 따라서 이 유명 증인 집합은 판사(judge)의 역할을 하며, 자신들에게 도달한 모든 이벤트에 대해 총합 순서(total order)와 합의 타임스탬프(consensus timestamp)를 정의할 수 있다.

정리 5.19(Theorem 5.19) (비잔틴 내결함성 정리, Byzantine Fault Tolerance Theorem). 정직한 구성원(honest member)이 생성한 모든 이벤트 x 는 결국 확률 1로 전체 이벤트의 총합 순서(total order) 내에서 합의 위치(consensus position)를 부여받는다.

증명 정직한 구성원은 정의상 모든 다른 정직한 구성원과 결국 통신하게 되며, 네트워크를 제어하는 공격자도 정직한 구성원 간의 통신을 무한히 차단할 수는 없다고 가정한다. 따라서 모든 정직한 구성원은 결국 이벤트 x 를 알게 된다. 그 결과, 모든 고유 유명 증인(unique famous witnesses)이 x 의 자손(descendant)이 되는 라운드가 결국 존재하게 된다. 이때 혹은 그보다 이른 시점의 라운드 r 에서, 모든 유명 증인들이 x 의 자손이 된다. 따라서 x 는 라운드 r 의 수신 라운드(received round)로 지정되며, 라운드 r 의 구성원들이 x 를 처음 인식한 시각들의 중앙값(median)을 기반으로 합의 타임스탬프(consensus timestamp)를 부여받는다. 이 시점에서 x 의 역사적 위치(consensus place in history)는 고정된다. 또한 이후에 x 보다 앞서야 할 새로운 이벤트 y 가 발견될 가능성은 없다. 왜냐하면, 합의 순서에서 더 앞서려면 y 의 수신 라운드가 r 이하이어야 하는데, 이는 라운드 r 의 모든 유명 증인들이 y 를 수신해야 함을 의미한다. 그러나 한 라운드의 유명 증인 집합이 결정되면, 그들의 모든 조상 또한 결정되므로, 이후 해시그래프가 성장하더라도 새로운 조상이 추가될 수 없다. 더 나아가, 한 라운드의 모든 증인에 대한 유명 여부가 결정된 후에는, 새로운 유명 증인이 추가되는 일도 없다. 새롭게 발견된 라운드 r 의 증인은 이미 알려진 라운드 $r + 1$ 의 증인들(그 수가 $2n/3$ 초과임)의 조상이 아니므로, 합의 알고리즘은 즉시 해당 증인을 “유명하지 않음”으로 분류한다. 따라서 한 번 이벤트가 총합 순서(total order) 내의 위치를 부여받으면, 그 위치는 절대로 바뀌지 않는다. 새로운 이벤트가 추가되어도 기존 이벤트의 순서가 교체되거나, 새 이벤트가 기존보다 앞서 삽입되는 일은 결코 발생하지 않는다. ■

6. 공정성(Fairness)

기존의 대부분의 분산 합의(distributed consensus) 시스템은 거래 순서 결정(consensus ordering of transactions)에서 “공정성(fairness)”을 보장하지 못할 수 있다. 이를 이해하기 위해 먼저 단일 서버에서 운영되는 주식시장을 생각해 보자. 앨리스(Alice)와 밥(Bob)이 각각 해당 서버에 매수 주문을 제출한다고 가정하자. 앨리스가 밥보다 조금 먼저 주문을 보냈다면, 서버가 공정하다면 앨리스의 거래가 밥의 거래보다 먼저 처리되어야 한다. 일부 애플리케이션에서는 정확한 순서가 중요하지 않지만, 주식시장과 같은 환경에서는 이 결정이 공정하게 이루어지는 것이 매우 중요하다.

이제 중앙 서버가 없는 분산형 피어투피어(peer-to-peer) 시스템을 생각해 보자. 여기서는 커뮤니티 전체가 합의(consensus)를 통해 어떤 거래가 먼저였는지를 결정해야 한다. 이 경우에도 합의 결정이 공정해야 한다는 점은 여전히 매우 중요하다. 그러나 “공정”의 정의는 무엇이어야 할까?

한 가지 단순한 정의는 “먼저 생성된 거래를 우선시한다”일 수 있다. 그러나 이것은 나쁜 정의이다. 예를 들어, 앨리스가 숲속의 오두막에서 인터넷이 연결되지 않은 상태로 밥보다 1초 먼저 거래를 생성했다고 가정하자. 커뮤니티는 밥의 거래만 듣게 되고, 따라서 밥이 먼저 거래했다고 판단할 것이다. 1년 후, 앨리스가 숲에서 나와 인터넷에 다시 연결되면, 커뮤니티는 “공정”을 위해 과거의 역사를 수정해야 할 것이다. 이는 심각한 문제를 초래한다. 따라서 “공정성”을 그렇게 정의하는 것은 적절하지 않다. 거래가 실제로 커뮤니티에 전송되어야만 “먼저 제출된 것”으로 인정받을 수 있다는 요건이 필요하다.

또 다른 정의는 “현재 리더(leader)”에게 먼저 도달한 거래를 먼저로 간주하는 것이다. 그러나 이 역시 문제가 있다. 리더는 팍소스(Paxos) 알고리즘에 의해 선택된 구성원일 수도 있고, 라운드로빈(round-robin) 방식에서 현재 차례가 된 구성원일 수도 있으며, 작업증명(proof-of-work) 시스템에서는 퍼즐을 먼저 푼 채굴자일 수도 있다. 그러나 이러한 리더는 자신의 의도에 따라 임의로 앨리스의 거래나 밥의 거래 중 하나를 일정 시간 무시하거나 지연시켜, 순서를 인위적으로 조작할 수 있다. 신뢰가 분산되어야 하는 시스템에서, 단일 구성원에게 전적인 신뢰를 맡기는 것은 바람직하지 않다.

또 다른 접근으로는, 거래가 커뮤니티 전체의 일정 비율(fraction)에 먼저 도달한 순서를 반영하도록 “공정성”을 정의할 수도 있다. 이 경우 커뮤니티는 거래가 “가상 서버(virtual server)”에 먼저 도달한 시점을 기준으로 순서를 결정한다. 여기서 “서버에 도달했다”는 것은 커뮤니티의 일정 비율 이상이 해당 거래를 수신했다는 의미이다. 하지만 이 정의에도 문제가 있다. 예를 들어, “거래가 커뮤니티의 절반 이상에게 먼저 도달한 경우”를 기준으로 삼는다면, 캐롤(Carol)은 앨리스를 먼저 보고, 데이브(Dave)는 밥을 먼저 보고, 나머지는 정확히 반반으로 나뉘는 경우가 생길 수 있다. 이때 캐롤과 데이브가 공격자일 경우, 그들은 자신들이 본 결과를 커뮤니티에 전달하지 않은 채 영구적으로 오프라인이 될 수 있다. 이런 경우 커뮤니티는 캐롤과 데이브의 투표를 무기한 기다리게 되어, 결코 “공정한 합의”에 도달하지 못하게 된다.

더 나은 정의는 다음과 같을 수 있다. 앨리스의 거래가 밥의 거래보다 먼저 커뮤니티의 상당 비율에 도달하고, 그 거래를 먼저 본 구성원들이 곧이어 다른 대부분의 구성원들과 빠르게 통신했을 경우, “앨리스의 거래가 먼저”라고 판단하는 것이 공정하다고 보는 것이다. 이 정의에 따르면, 앨리스와 밥이 거의 동시에 거래를 네트워크에 퍼뜨리고, 두 거래가 비슷한 속도로 전파된다면, 어느 쪽이 먼저로 결정되더라도 공정하다고 볼 수 있다. 그러나 앨리스가 밥보다 약간 더 빨리(단 하나의 가십 동기화 정도의 시간 차이로) 거래를 퍼뜨렸다면, 거래가 가십 프로토콜을 통해 지수적으로 확산되면서 각 동기화마다 수신자가 두 배씩 늘어나기 때문에, 결과적으로 전체 인구의 $2/3$ 이상이 밥보다 앨리스를 먼저 인식하고, $1/3$ 이하만이 밥을 먼저 인식하게 될 것이다. 이 경우 합의에서 앨리스를 밥보다 우선하는 것이 공정하다. 해시그래프 합의 알고리즘은 바로 이러한 의미에서 공정하다. 온라인 상태이며 정기적으로 참여하는 구성원들은 *유명한 증인(famous witnesses)*으로 불리는 이벤트 집합을 생성하며, 합의 결정은 “대다수의 유명 증인에게 먼저 도달한 거래”를 “먼저 발생한 거래”로 간주하는 방식으로 이루어진다. 소수의 구성원들이 오프라인이거나, 네트워크 분할(partition)로 인해 나머지와 통신하지 못한다면, 그들은 유명 증인을 가질 수 없으며, 따라서 해당 구성원들에게 거래가 도달하더라도 커뮤니티 전체에 도달한 것으로 간주되지 않는다. 반면, 그들이 커뮤니티의 나머지 구성원들과 정상적으로 통신한다면, 그들은 유명 증인으로 간주되며, 커뮤니티 내에서 “누가 먼저 거래를 전파했는가”를 결정하는 데 참여하게 된다.

이 시스템에 대한 일부 공격은 합의 시스템의 실패로 간주되지 않는다. 왜냐하면 동일한 공격이 단일 서버 기반 시스템에도 똑같이 효과적이기 때문이다. 예를 들어, 비잔틴 증명(Byzantine proofs)에서는 공격자가 인터넷을 제어하고 임의의 메시지를 지연시킬 수 있다고 가정한다. 만약 공격자가 실제로 그러한 능력을 가진다면, 단순히 앨리스를 인터넷에서 차단하여 밥이 거래를 전송하고 기록할 때까지 기다리게 만들 수 있다. 이는 실제 인터넷 환경에서도 서비스 거부(DoS) 공격을 통해 가능하며, 밥이 수많은 패킷을 전송하여 네트워크를 포화시킴으로써 앨리스의 통신을 방해하는 식이다. 이러한 공격은 중앙 서버에 연결된 경우에도 동일하게 효과적이므로, 이는 합의 시스템의 결함이라기보다 인터넷 인프라 자체의 한계로 볼 수 있다.

비슷하게, 밥이 더 많은 대역폭(bandwidth)을 구매함으로써 앨리스보다 유리한 위치를 점할 수도 있다.

예를 들어, 밥이 앨리스보다 8배 빠른 네트워크를 가지고 있어, 앨리스가 거래를 한 명에게 전송하는 동안 밥은 8명에게 동시에 전송할 수 있다면, 그는 약 3회 정도의 가십 동기화(gossip sync)에 해당하는 시간 이점을 얻는다. 그러나 이것은 시스템의 실패로 간주되지 않는다. 밥의 메시지가 실제로 앨리스의 메시지보다 먼저 세상에 도달했다면, 그가 먼저 기록되는 것이 당연하다. 이는 오늘날의 주식시장과 유사한데, 일부 기업들이 거래소 서버에 더 빨리 접근하기 위해 더 빠른 네트워크 연결에 막대한 비용을 지불하는 것과 같다. 따라서 이러한 경우 합의 알고리즘이 “불공정하다”고 볼 수는 없으며, 오히려 중앙 서버 기반의 시스템과 동일한 수준의 공정성을 유지하고 있다고 평가할 수 있다.

7. 일반화 및 확장(Generalizations and Enhancements)

7.1. 지분증명(Proof-of-stake)

지금까지는 모든 구성원이 동등하다고 가정했다. 앞서 제시된 알고리즘에서는 “ $2n/3$ 이상의 구성원” 또는 “유명한 증인의 절반 이상”과 같은 표현이 등장하며, “중앙값(median)” 개념도 사용된다. 또한, $2n/3$ 을 초과하는 정직한 구성원이 존재할 때 비잔틴 합의가 수렴함을 증명하였다.

이 알고리즘은 구성원이 불평등한 상황으로 쉽게 확장할 수 있다. 각 구성원에게 양의 정수 값을 부여하고, 이를 해당 구성원의 *지분(stake)*이라 정의한다. 그렇다면 기존의 단순 투표는 가중 투표(weighted voting)로 대체될 수 있으며, 중앙값은 지분에 따라 가중된 중앙값(weighted median)으로 계산될 수 있다. 즉, 각 투표는 구성원의 지분에 비례하여 가중된다. 이때 모든 정의, 알고리즘, 증명에서 “ $2n/3$ 이상의 구성원”이라는 표현을 “전체 지분 합이 n 일 때, 총 지분의 $2n/3$ 을 초과하는 구성원 집합”으로 바꾸면 된다. 또한 “이벤트 집합 S 의 타임스탬프 중앙값”은 “이벤트 생성자의 지분을 가중치로 한 가중 중앙값(weighted median)”으로 바뀐다. 이 개념을 직관적으로 설명하면, S 의 각 이벤트 y 에 대해 해당 생성자의 지분만큼 y 의 타임스탬프 복사본을 가상의 주머니(bag)에 넣고, 그 전체 타임스탬프들의 중앙값을 취하는 방식이다.

비잔틴 내결함성 증명은 공격자가 전체 인구의 $1/3$ 미만일 때 성립했다. 새로운 정의를 적용하면, 이제 공격자들의 총 *지분 합계*가 전체 지분의 $1/3$ 미만일 때 시스템의 안전성이 보장된다.

이 지분증명(Proof-of-stake, PoS) 기반 시스템은 기존의 비가중 시스템보다 더 일반화된 형태이다.

각 구성원의 지분을 1로 설정하면 기존의 균등 시스템을 그대로 구현할 수 있다. 하지만 이 개념을 활용하면 더 나은 형태의 시스템 설계도 가능하다. 예를 들어, 구성원의 신뢰 수준(trust level)에 따라 지분을 다르게 설정할 수 있다. 어떤 구성원은 신원 검증이나 심사 과정을 거쳐 더 높은 신뢰도를 얻었다면, 그에 비례하여 더 높은 지분을 부여받을 수 있다. 또는 시스템의 안정적 운영에 더 큰 이해관계를 가진 구성원에게 더 큰 투표 가중치를 부여할 수도 있다. 암호화폐의 경우, 각 구성원의 코인 수량을 지분으로 정의할 수 있다. 이는 코인을 많이 보유한 사람이 시스템의 정상 작동에 더 큰 이해관계를 가진다는 가정에 기반한다. 또한, 상호 신뢰하는 구성원 집단이 동등한 지분을 가진 상태에서 커뮤니티를 시작할 수도 있다. 이후, 각 기존 구성원이 새로운 구성원을 초대할 수 있도록 허용하되, 초대된 사람은 자신의 지분 일부를 피조대자와 나누도록 규정할 수 있다. 이 방식은 한 구성원이 다수의 허위 계정(sybil account)을 생성해 투표를 조작하는 *시빌 공격(Sybil attack)*을 억제하는 효과가 있다. 즉, 계정을 늘릴수록 자신의 지분이 희석되므로, 투표권을 인위적으로 확대할 유인이 사라진다.

*지분 기록(stake record)*이란 각 구성원과 그가 보유한 지분의 양을 기록한 목록을 의미한다. 지금까지는 지분 기록이 모든 구성원에게 알려져 있으며, 변하지 않는다고 가정해왔다. 그러나 이 가정을 완화하는 것은 어렵지 않다.

특정 유형의 거래(transaction)가 지분 기록을 변경한다고 가정하자.

커뮤니티는 초기 단계에서 이러한 거래의 유효성을 판단하는 규칙을 정할 수 있다. 예를 들어, 각 구성원은 최대 10명의 신규 구성원만 초대할 수 있도록 하거나, 새로운 구성원을 초대하는 사람은 자신의 지분 일부를 피초대자에게 반드시 양도해야 한다는 규칙을 둘 수 있다. 이러한 거래의 유효성은 합의 순서(consensus order)에서 거래가 처리되는 정확한 순서에 따라 달라질 수도 있다. 예를 들어 “단 한 명만 신규 구성원을 초대할 수 있다”는 규칙이 있을 때, 앨리스가 캐롤(Carol)을 초대하고 밥이 동시에 데이브(Dave)를 초대할 경우, 합의 순서에서 먼저 처리된 초대만 성공하며 나머지는 실패한다.

이 모든 상황은 해시그래프 합의 알고리즘으로 처리 가능하다. 라운드 r 의 “첫 번째(famous)” 증인들이 결정된 시점에, 정확히 어떤 이벤트들이 라운드 r 의 수신 라운드(received round)를 갖는지, 그리고 그 이벤트들이 합의 순서 내에서 어떤 위치를 차지하는지를 계산할 수 있다. 그 시점에 해당 이벤트에 포함된 각 거래를 처리하면서 규칙에 따라 유효성을 검증하고, 유효한 거래만 실제로 적용할 수 있다. 이 과정에서 지분 기록이 변경될 수 있다.

만약 지분 기록이 변경된다면, 알고리즘은 라운드 r 이후의 모든 이벤트에 대해 다시 실행되어야 한다. 이로 인해 “강하게 봄(strongly seeing)”의 계산 결과, 이벤트의 라운드 번호, 어떤 이벤트가 증인(witness)인지, 그리고 어떤 이벤트가 유명 증인(famous witness)인지의 계산 결과가 변경될 수 있다.

중요한 점은, 라운드 r 의 증인들이 유명한지를 판단할 때는 *이전 지분 기록(old stake record)*을 사용한다는 것이다. 라운드 r 의 투표 과정은 여러 라운드에 걸쳐 계속될 수 있지만, 그동안은 계속 이전 지분 기록을 사용한다. 라운드 r 의 합의가 완전히 끝난 후, 이후 라운드들의 계산은 새 지분 기록(new stake record)을 사용하여 다시 수행된다.

이 접근법을 사용하면, 모든 구성원이 동일한 시점의 동일한 지분 기록을 기준으로 계산을 수행하게 되어, 계산 결과에 대한 완전한 합의가 가능해진다. 따라서 모든 구성원이 동일한 결과를 도출할 수 있으며, 비잔틴 합의는 여전히 확률 1로 보장된다.

7.2. 서명된 상태(Signed state)

시스템의 또 다른 확장 방식은 *서명된 상태(signed state)*를 도입하는 것이다. 라운드 r 의 각 증인이 유명하지 여부에 대한 합의가 이루어지면, 수신 라운드(received round)가 r 이하인 모든 이벤트에 대해 총합 순서(total order)를 계산할 수 있다. 또한 그 시점 이후의 모든 이벤트(아직 알려지지 않은 이벤트 포함)는 반드시 수신 라운드가 r 보다 크다. 즉, 이 시점에서 라운드 r 이하의 모든 이벤트에 대한 “역사(history)”는 동결(frozen)되어 불변(immutable) 상태가 된다. 따라서 구성원은 해당 이벤트들의 모든 거래를 합의 순서대로 데이터베이스에 적용하여, 그 결과로 도달하는 상태(state)를 계산할 수 있다. 모든 구성원이 동일한 합의 순서를 사용하므로, 결과적으로 계산된 상태 역시 동일하게 된다. 이것이 바로 *합의 상태(consensus state)*이다. 각 구성원은 이 합의 상태의 해시(hash)를 계산하고 디지털 서명을 생성하여, 이를 새로운 거래로 포함시킬 수 있다. 이후 가십(gossip)을 통해 각 구성원은 합의 상태에 대한 다수의 서명을 수신하게 된다. 전체 구성원의 1/3 이상으로부터 서명이 수집되면, 그 합의 상태와 해당 서명 집합이 결합되어 시스템의 공식 합의 상태(official consensus state), 즉 *서명된 상태(signed state)*로 인정된다. 이 서명된 상태는 라운드 r 시작 시점의 공식 시스템 상태로 간주된다. 이 서명된 상태는 커뮤니티 외부의 제3자에게도 전달할 수 있으며, 누구든 서명을 검증함으로써 상태의 진위를 확인할 수 있다. 이 시점 이후, 구성원은 해당 상태를 생성하기 위해 사용된 거래나 이벤트들을 삭제해도 된다. 오직 최종 상태(state)만 보존하면 충분하다. 이러한 정리 작업은 몇 분마다 수행될 수 있어, 시스템은 결코 과도한 양의 거래나 이벤트를 저장하지 않아도 된다. 즉, 오직 “합의 상태(consensus state)”만 유지하면 된다. 물론, 구성원이 과거의 이벤트나 거래, 상태를 기록이나 감사를 위해 별도로 보존할 수도 있다. 그러나 모든 참여자가 그 데이터를 삭제하더라도, 시스템은 여전히 불변하고(immutable) 안전하게 유지된다.

인구의 3분의 1 미만만 비정직하다는 가정하에, *서명된 상태(signed state)*에는 반드시 하나 이상의 정직한 서명이 포함되므로, 이는 합의 알고리즘이 도출한 커뮤니티의 합의 결과를 신뢰할 수 있는 형태로 표현한다. 만약 구성원 집합(또는 그들의 지분)이 시간이 지나면서 변화할 수 있다면, 그 지분 기록(stake record)과 그 이력(history) 역시 상태(state)의 일부로 포함된다. 또한, 신뢰 임계치인 1/3은 다른 값(예: 2/3 초과 등)으로 대체하더라도 시스템의 정상 동작에는 문제가 없다.

7.3. 효율적인 가십(Efficient gossip)

가십 프로토콜(gossip protocol)은 대역폭을 매우 효율적으로 사용한다. 만약 충분히 많은 거래(transaction)가 생성되어, 모든 이벤트가 최소 한 개 이상의 거래를 포함한다고 가정하자. 어떤 복제 상태 기계(replicated state machine)에서도, 인터넷과 같은 점대점(point-to-point) 네트워크를 사용하는 한, 각 구성원은 모든 서명된 거래를 한 번씩 수신하고, 평균적으로 한 번씩 전송해야 한다. 해시그래프 기반 가십에서도 동일하다. 단지 서명이 개별 거래가 아니라 거래를 포함하는 이벤트(event)에 부여된다는 점만 다르다. 추가적인 오버헤드는 단 두 개의 해시(hash)와 하나의 타임스탬프(timestamp), 그리고 동기화(sync) 시작 시 전송되는 카운트 배열(array of counts)뿐이다. 하지만 이 해시 값들은 실제로 인터넷을 통해 전송할 필요조차 없다. 이벤트의 생성자(creator)와 그 이벤트의 다른 부모(other-parent)의 순번(sequence number) 정보만 전송하면 충분하다.

예를 들어, 앨리스(Alice)가 생성한 100번째 이벤트의 다른 부모(other-parent)가 에드(Ed)의 50번째 이벤트라고 하자. 밥(Bob)이 동기화 중에 이 앨리스의 이벤트를 캐롤(Carol)에게 전송할 때, 밥은 이벤트 내의 두 개 해시를 보낼 필요가 없다. 대신 “이 이벤트는 앨리스가 만든 것이고, 다른 부모는 에드의 50번째 이벤트다”라고만 알리면 된다. 밥은 초기 카운트를 통해 캐롤이 어떤 이벤트들을 이미 보유하고 있는지 알고 있으므로, 캐롤은 자신이 알고 있는 앨리스의 마지막 이벤트가 99번째임을 알게 된다. 따라서 밥이 100번째 이벤트의 자기 부모(self-parent) 해시나 순번을 전송할 필요가 없다. 단지 “이 이벤트는 앨리스의 것이다”라고만 알리면 된다. 마찬가지로, 밥은 “다른 부모는 에드의 50번째 이벤트다”라는 정보만 추가로 전송하면 된다. 즉, 두 개의 대형 해시를 전송하는 대신, 밥은 단순히 (Alice, Ed, 50)이라는 3개 항목(triplet)을 전송하는 셈이다. 식별자(ID)와 순번(sequence number)은 각각 1~2바이트로 압축할 수 있으므로, 이 트리플렛은 전체 3~6바이트만 필요하다. 이는 512비트(64바이트) 서명(signature)나 이벤트 내의 거래 데이터(평균 100바이트 이상)에 비해 매우 작은 오버헤드이다. 따라서 각 이벤트가 최소 한 개 이상의 거래를 포함한다면, 거래만을 단순히 가십하는 경우와 비교해 해시그래프를 가십하는 데 추가되는 오버헤드는 사실상 무시할 만한 수준이다.

또한, 투표(voting)가 가상이기 때문에 합의를 달성하기 위한 추가 대역폭 비용은 전혀 발생하지 않는다. 이러한 점에서, 해시그래프 합의의 대역폭 요구량은 이론적 한계(theoretical limit)에 매우 근접한다. 즉, 단순히 서명된 거래를 타임스탬프와 함께 전송하는 데 필요한 최소 대역폭 수준이다.

만약 어떤 시스템이 거래를 전송할 때 타임스탬프를 첨부하지 않아도 되는 애플리케이션이라면, 대역폭을 절약할 수 있다. 해시그래프 합의도 동일한 최적화를 적용할 수 있다. 이 경우, 이벤트 내의 “타임스탬프”는 단순히 자기 부모의 타임스탬프에 1을 더한 정수(integer)로 대체될 수 있다. 밥이 이벤트를 캐롤에게 전송할 때, 캐롤은 이 순번을 직접 계산할 수 있으므로, 밥이 실제로 이를 인터넷을 통해 전송할 필요가 없다.

또한, 거래만 전송하는 시스템은 동일한 생성자(creator)의 여러 거래를 묶어서 하나의 서명만 부여함으로써 대역폭을 절약할 수 있다. 해시그래프도 동일한 방식으로 작동할 수 있다. 여러 거래를 하나의 이벤트에 포함시키고, 해당 이벤트에 하나의 서명만 추가하면 된다.

결론적으로, 해시그래프 합의의 대역폭 요구량은 모든 경우에 있어 이론적으로 가능한 최소 수준에 매우 근접한다.

7.4. 빠른 선거(Fast elections)

알고리즘의 두 번째 부분은 증인의 유명 여부(fame)를 결정하기 위한 비잔틴 합의(Byzantine agreement) 단계이다. 이 단계에는 흥미로운 특성이 있다. 모든 구성원이 온라인 상태이며 정기적으로 참여하고 있을 때, 비잔틴 합의는 거의 모든 투표자들이 동일한 초기 투표값을 가지는 일련의 선거에 적용된다. 그 이유는 라운드 $r + 1$ 의 증인(witness)이 다수의 라운드 r 증인들을 강하게 보기(strongly see) 때문이다. 따라서 하나의 라운드는 대략 두 “가십 주기(gossip period)” 정도 지속된다고 예상할 수 있다. 가십 주기란 하나의 메시지가 커뮤니티 전체에 전파되는 데 걸리는 시간이다. 참여자 수가 n 일 때, 이는 약 $\log_2(n)$ 번의 동기화(sync)에 해당하는 시간이다. 라운드 $r + 1$ 의 증인 x 가 라운드 r 의 증인 y 의 유명 여부에 대해 YES로 투표하기 위해 반드시 y 를 강하게 볼 필요는 없다. 단순히 y 를 볼 수 있거나 하면 된다. y 는 단일 가십 주기 안에 온라인 상태의 대부분의 구성원들에게 전파될 것으로 예상되며, 실제로는 두 번의 가십 주기 내에 거의 모든 구성원에게 전달될 확률이 매우 높다. 따라서 실제 운영 환경에서는 모든 참여자가 정상적으로 온라인 상태일 경우, 증인의 유명 여부는 거의 항상 즉시 결정되며, 다수의 투표 라운드가 필요하지 않다.

유사하게, y 가 라운드 r 의 증인이라 하더라도, 해당 증인이 잠자고 있던 구성원에 의해 라운드 r 의 거의 끝부분에서 생성된 경우, 대부분의 라운드 $r + 1$ 증인들이 y 에 대해 NO로 투표할 가능성이 높으며, 이 경우에도 선거는 즉시 종료된다. 단, 구성원이 라운드 종료 직전 깨어나 y 를 생성하는 매우 짧은 시간(한 번의 동기화 정도의 시간) 동안에는 투표가 근소한 차이로 양분될 수 있다. 그러나 모든 온라인 구성원들이 서로 무작위로 자주 동기화한다면, 이러한 경우에도 약 3라운드 내에 합의가 수렴되며, 3라운드를 초과할 확률은 수% 이하, 6라운드를 초과할 확률은 0.1% 미만이다. 만약 공격자가 인터넷을 완전히 제한한다면, 이러한 선거 과정을 지속적으로 많은 라운드로 끌 수 있다. 이를 방지하기 위해 “서명의 중간 비트(middle bit of the signature)” 대신, 암호학적 공유 코인(shared coin) 프로토콜을 사용할 수 있다. 서명의 중간 비트는 각 구성원이 독립적으로 임의의 동전 던지기 결과를 얻는 것과 유사한 방식으로 동작하며, 공격자가 미리 결과를 예측할 수 없도록 한다. 공유 코인 프로토콜은 이와 유사하지만, 모든 구성원이 동일한 “임의의 결과(random result)”를 공유하도록 보장한다. 이 방식을 사용하면 이론적 최악의 경우에도 기대 합의 시간이 상수(constant) 수준으로 줄어든다. 그러나 실제 환경에서는 이러한 개선이 큰 효용을 제공하지는 않는다. 공격자가 인터넷을 충분히 통제하여 정직한 구성원들이 장기간 무작위 동기화를 하지 못하게 만든다면, 이미 공격자는 정직한 사용자들의 인터넷 접근 자체를 차단할 수 있을 정도의 권한을 가진 것이다. 따라서 공유 코인(shared coin)은 이론적 연구 관점에서는 흥미롭지만, 실제 시스템에서는 반드시 필요하지 않다. 그럼에도 불구하고, 필요 시 언제든지 도입할 수 있는 선택지로 남아 있다.

7.5. 효율적인 계산(Efficient calculations)

알고리즘의 첫 번째 단계는 이벤트가 충분한 수의 라운드 r 이벤트를 강하게 볼 수 있는지 여부에 따라 해당 이벤트에 라운드 r 또는 $r + 1$ 을 할당하는 것이다. 따라서 라운드 r 의 증인 이벤트 x 가 임의의 이벤트 y 에 의해 강하게 보일 수 있는지를 계산할 필요가 있다. 다음은 그 답을 계산하는 한 가지 방법이다.

각 이벤트에 자기 부모(self-parent)의 시퀀스 번호보다 1 큰 시퀀스 번호를 부여한다. y 를 위한 배열과 x 를 위한 배열을 각각 저장한다. y 배열은 y 의 조상(ancestor)인 각 구성원이 생성한 마지막 이벤트의 시퀀스 번호를 기억한다. x 배열은 x 의 자손(descendant)인 각 구성원이 생성한 가장 이른 이벤트의 시퀀스 번호를 기억한다. 이 두 배열을 비교하여, y 배열의 원소 중에서 x 배열의 해당 원소보다 크거나 같은 것이 몇 개인지를 찾는다. 그러한 일치 항목이 $2n/3$ 보다 많으면, y 는 x 를 강하게 본다고 판단할 수 있다. x 와 y 배열의 비교 연산은 멀티스레딩(multithreading, 더 많은 코어 사용), 여러 요소를 하나의 정수에 묶는 데이터 패킹(packing, ALU 효율 향상), 어셈블리 언어 사용(assembly language, CPU 벡터 명령어 접근), 또는 GPU 사용(더 많은 벡터 병렬성 확보)을 통해 속도를 높일 수 있다.

8. 결론(Conclusions)

본 논문에서는 스월즈(Swirls) 해시그래프 데이터 구조(hashgraph data structure)와 스월즈 해시그래프 합의 알고리즘(hashgraph consensus algorithm)을 기반으로 한 새로운 시스템을 제시하였다. 이 시스템은 공정성(fairness), 고속성(fast performance), 비잔틴 내결함성(Byzantine fault tolerance)을 모두 갖추고 있으며, 가상 투표(virtual voting)를 통해 대역폭 효율이 매우 높다. 본 논문에서 제시된 알고리즘은 도식화된 의사코드(pseudocode)로 설명되어 있으며, 명령형 언어(imperative language) 기반의 형식을 따른다. 그러나 동시에, 이 알고리즘은 함수형 언어(functional form)로도 매우 자연스럽게 표현될 수 있다. 부록(Appendix)에서는 해당 알고리즘을 간결한 함수형 형태로 제시하며, 이는 기술적 관심이 있는 독자들에게 흥미로울 수 있다.

참고 문헌(References)

[1] Miguel Castro and Barbara Liskov. Practical byzantine fault tolerance. In *Proceedings of the Third Symposium on Operating Systems Design and Implementation, OSDI '99*, pages 173–186, Berkeley, CA, USA, 1999. USENIX Association.

[2] Leslie Lamport, Robert Shostak, and Marshall Pease. The byzantine generals problem. *ACM Trans. Program. Lang. Syst.*, 4(3):382–401, July 1982.

[3] Michael J. Fischer, Nancy A. Lynch, and Michael S. Paterson. Impossibility of distributed consensus with one faulty process. *J. ACM*, 32(2):374–382, April 1985.

[4] Leslie Lamport. The part-time parliament. *ACM Trans. Comput. Syst.*, 16(2):133–169, May 1998.

- [5] Diego Ongaro and John Ousterhout. In search of an understandable consensus algorithm. In *2014 USENIX Annual Technical Conference (USENIX ATC 14)*, pages 305–319, Philadelphia, PA, June 2014. USENIX Association.
- [6] Andrew Miller, Yu Xia, Kyle Croman, Elaine Shi, and Dawn Song. The honey badger of bft protocols. *Cryptology ePrint Archive*, Report 2016/199, 2016. <http://eprint.iacr.org/>.
- [7] Allen Clement, Edmund Wong, Lorenzo Alvisi, Mike Dahlin, and Mirco Marchetti. Making byzantine fault tolerant systems tolerate byzantine faults. In *Proceedings of the 6th USENIX Symposium on Networked Systems Design and Implementation*, NSDI'09, pages 153–168, Berkeley, CA, USA, 2009. USENIX Association.
- [8] Satoshi Nakamoto. Bitcoin: A peer-to-peer electronic cash system. Posted to the internet November, 2008. 2008. <http://bitcoin.org/bitcoin.pdf>.
- [9] Giulio Prisco. Intel develops ‘Sawtooth Lake’ distributed ledger technology for the Hyperledger project. *Bitcoin Magazine*, April 2016.
- [10] Dag-Erling Smorgrav. FreeBSD quarterly status report. Posted on FreeBSD.org, 2013. <http://www.freebsd.org/new/status/report-2013-09-devsummit.html#Security>.
- [11] Miguel Miguel Correia, Giuliana Santos Veronese, Nuno Ferreira Neves, and Paulo Veríssimo. Byzantine consensus in asynchronous message-passing systems: a survey. *International Journal of Critical Computer-Based Systems*, 2(2):141–161, 2011.

9. 부록 A: 함수형 형태의 합의 알고리즘(Appendix A: Consensus algorithm in functional form)

이벤트(event)는 튜플 $e = \{p, h, t, i, s\}$ 로 정의된다.

$p = \text{payload}(e)$ = “페이로드(payload)” 데이터 (예: 거래 목록)
 $h = \text{hashes}(e)$ = 이벤트의 부모 해시 목록 (self-parent가 첫 번째)
 $t = \text{time}(e)$ = 이벤트 생성자가 주장하는 생성 일시(timestamp)
 $i = \text{creator}(e)$ = 이벤트 생성자의 ID 번호
 $s = \text{sig}(e)$ = $\{p, h, t, i\}$ 에 대한 생성자의 디지털 서명

$\text{parents}(x)$ = 이벤트 x 의 부모 집합
 $\text{selfParent}(x)$ = 이벤트 x 의 자기 부모(self-parent), 없으면 $\emptyset$
 n = 전체 구성원 수
 c = 코인 라운드 주기(예: $c = 10$)
 d = 선거 시작 전 지연 라운드 수(예: $d = 1$)
 E = 해시그래프 내의 모든 이벤트 집합
 E_0 = $E \cup \{\emptyset\}$
 $\mathbb{T}$ = 모든 가능한 (시간, 날짜) 쌍의 집합
 $\mathbb{B} = \{\text{true}, \text{false}\}$
 $\mathbb{N} = \{1, 2, 3, \dots\}$

$\text{parents} : E \rightarrow 2^E$
 $\text{selfParent} : E \rightarrow E_0$
 $\text{ancestor} : E \times E \rightarrow \mathbb{B}$
 $\text{selfAncestor} : E \times E \rightarrow \mathbb{B}$
 $\text{manyCreators} : 2^E \rightarrow \mathbb{B}$
 $\text{see} : E \times E \rightarrow \mathbb{B}$
 $\text{stronglySee} : E \times E \rightarrow \mathbb{B}$
 $\text{selfParentRound} : E \rightarrow \mathbb{N}$
 $\text{round} : E \rightarrow \mathbb{N}$
 $\text{witness} : E \rightarrow \mathbb{B}$
 $\text{diff} : E \times E \rightarrow \mathbb{I}$
 $\text{votes} : E \times E \times \mathbb{B} \rightarrow \mathbb{N}$
 $\text{fractTrue} : E \times E \rightarrow \mathbb{R}$
 $\text{decide} : E \times E \rightarrow \mathbb{B}$
 $\text{copyVote} : E \times E \rightarrow \mathbb{B}$
 $\text{vote} : E \times E \rightarrow \mathbb{B}$
 $\text{famous} : E \rightarrow \mathbb{B}$
 $\text{uniqueFamous} : E \rightarrow \mathbb{B}$
 $\text{roundsDecided} : \mathbb{N} \rightarrow \mathbb{B}$

$roundReceived : E \rightarrow \mathbb{N}$
 $timeReceived : E \rightarrow \mathbb{T}$

$$ancestor(x, y) = x = y \vee \exists z \in parents(x), ancestor(z, y)$$

$$selfAncestor(x, y) = x = y \vee (selfParent(x) \neq \emptyset \wedge selfAncestor(selfParent(x), y))$$

$$manyCreators(S) = |S| > 2n/3 \wedge \forall x, y \in S, (x \neq y \implies creator(x) \neq creator(y))$$

$$\begin{aligned}
see(x, y) &= \\
ancestor(x, y) &\wedge \neg(\exists a, b \in E, creator(y) = creator(a) = creator(b) \wedge ancestor(x, a) \wedge ancestor(x, b) \wedge \neg selfAncestor(a, b)) \\
stronglySee(x, y) &= see(x, y) \wedge (\exists S \subseteq E, manyCreators(S) \wedge (\forall z \in S \implies (see(x, z) \wedge see(z, y))))
\end{aligned}$$

$$selfParentRound(x) = \begin{cases} 1 & \text{if } selfParent(x) = \emptyset \\ round(selfParent(x)) & \text{otherwise} \end{cases}$$

$$\begin{aligned}
round(x) &= \\
\max(\{selfParentRound(x)\} \cup \{r + 1 \mid \exists S \subseteq E, manyCreators(S) \wedge (\forall y \in S, round(y) = r \wedge stronglySee(x, y))\})
\end{aligned}$$

$$witness(x) = (selfParent(x) = \emptyset) \vee (round(x) > round(selfParent(x)))$$

$$diff(x, y) = round(x) - round(y)$$

$$votes(x, y, v) = |\{z \in E \mid diff(x, z) = 1 \wedge witness(z) \wedge stronglySee(x, z) \wedge vote(z, y) = v\}|$$

$$fractTrue(x, y) = \frac{votes(x, y, true)}{\max(1, votes(x, y, true) + votes(x, y, false))}$$

$$\begin{aligned}
decide(x, y) &= \\
(selfParent(x) \neq \emptyset \wedge decide(selfParent(x), y)) \vee (witness(x) \wedge witness(y) \wedge diff(x, y) > d \wedge ((diff(x, y)) \bmod c = 0))
\end{aligned}$$

$$copyVote(x, y) = (\neg witness(x)) \vee (selfParent(x) \neq \emptyset \wedge decide(selfParent(x), y))$$

$$vote(x, y) = \begin{cases} vote(selfParent(x), y) & \text{if } copyVote(x, y) \\ see(x, y) & \text{if } \neg copyVote(x, y) \wedge diff(x, y) = d \\ 1 = middleBit(signature(x)) & \text{if } \neg copyVote(x, y) \wedge diff(x, y) \neq d \\ & \wedge (diff(x, y) \bmod c = 0) \\ & \wedge (\frac{1}{3} \leq fractTrue(x, y) \leq \frac{2}{3}) \\ fractTrue(x, y) \geq \frac{1}{2} & \text{otherwise} \end{cases}$$

$$famous(x) = \exists y \in E, decide(y, x) \wedge vote(y, x)$$

$$uniqueFamous(x) =$$

$$famous(x) \wedge \neg \exists y \in E, y \neq x \wedge famous(y) \wedge round(x) = round(y) \wedge creator(x) = creator(y)$$

$$roundsDecided(r) = \forall x \in E, ((round(x) \leq r \wedge witness(x)) \implies \exists y \in E, decide(y, x))$$

$$roundReceived(x) =$$

$$\min(\{r \in \mathbb{N} \mid roundsDecided(r) \wedge (\forall y \in E, round(y) = r \wedge uniqueFamous(y)) \implies ancestor(y, x)\})$$

$$timeReceived(x) =$$

$$\text{median}(\{time(y) \mid y \in E \wedge ancestor(y, x) \wedge (\exists z \in E, round(z) = roundReceived(x) \wedge uniqueFamous(z) \wedge selfAncestor(z, x))\})$$