

엑스알피(XRP)

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.

리플 프로토콜 합의 알고리즘

David Schwartz
david@ripple.com

Noah Youngs
nyoungs@nyu.edu

Arthur Britto
arthur@ripple.com

주의

본 논문은 원장 합의 프로토콜의 현재 상태나 그 분석을 반영하지 않습니다. 역사적 참고를 위해 이 초안을 계속 공개하지만, 이를 참조 자료로 사용해서는 **안 됩니다**. 합의 프로토콜에 대한 최신 분석 및 설명은 2018년 2월 20일 공개된 [arXiv:1802.07242](https://arxiv.org/abs/1802.07242)를 참조하십시오.

초록

비잔틴 장군 문제(Byzantine Generals Problem), 특히 분산 결제 시스템과 관련된 문제에 대해 여러 합의 알고리즘이 존재하지만, 많은 경우 네트워크 내 모든 노드가 동기적으로 통신해야 한다는 요구로 인해 높은 지연(latency)이 발생하는 한계를 지닙니다. 본 연구에서는 이러한 요구를 피하기 위해 더 큰 네트워크 내에서 공동으로 신뢰할 수 있는 서브네트워크(subnetworks)를 활용하는 새로운 합의 알고리즘을 제시합니다. 우리는 이러한 서브네트워크가 요구하는 “신뢰”가 사실상 최소이며, 구성 노드를 원칙적으로 선택함

으로써 더 줄일 수 있음을 보입니다. 또한 네트워크 전체에서 합의를 유지하기 위해 필요한 연결성 역시 최소화될 수 있음을 입증합니다. 그 결과, 비잔틴 오류에 직면하더라도 견고성을 유지하면서 낮은 지연을 보장하는 합의 알고리즘을 도출합니다. 우리는 이 알고리즘이 리플 프로토콜(Ripple Protocol)에 구현된 형태를 제시합니다.

1. 서론

분산 합의 시스템에 대한 관심과 연구는 최근 몇 년 동안 크게 증가했으며, 그 중심에는 분산 결제 네트워크가 있습니다. 이러한 네트워크는 중앙화된 주체에 의해 통제되지 않는 빠르고 저비용의 거래를 가능하게 합니다. 이러한 시스템의 경제적 이점과 단점만으로도 충분히 연구할 가치가 있지만, 본 연구에서는 모든 분산 결제 시스템이 직면해야 하는 몇 가지 기술적 과제에 초점을 맞춥니다. 이러한 문제는 다양하지만, 우리는 이를 크게 정확성(correctness), 합의(agreement), 유용성(utility)이라는 세 가지 범주로 구분합니다.

정확성이란 분산 시스템이 올바른 거래와 사기 거래를 구분할 수 있어야 함을 의미합니다. 전통적인 금융 환경에서는 기관 간 신뢰와, 거래가 실제로 주장된 기관에서 비롯되었음을 보장하는 암호학적 서명을 통해 이를 달성합니다. 그러나 분산 시스템에서는 네트워크 내 모든 참여자의 신원이 알려지지 않을 수 있으므로 이러한 신뢰가 존재하지 않습니다. 따라서 정확성을 확보하기 위해서는 대체적인 방법들이 필요합니다.

합의는 분산된 회계 시스템 내에서 단일한 글로벌 진실을 유지하는 문제를 의미합니다. 이는 정확성 문제와 유사하지만, 차이점은 네트워크의 악의적인 사용자가 사기 거래를 생성할 수는 없더라도(정확성 위배), 서로 인지하지 못하는 여러 개의 올바른 거래를 만들어낼 수 있다는 점입니다. 이렇게 되면 결과적으로 사기 행위가 발생할 수 있습니다. 예를 들어, 악의적인 사용자가 계정에 개별 구매를 각각 총당할 만큼의 자금만 있는 상태에서 동시에 두 건의 구매를 진행한다고 가정해 봅시다. 각각의 거래는 독립적으로는 올바르지만, 분산 네트워크 전체가 두 거래를 동시에 인식하지 못하면 명백한 문제가 발생합니다. 이는 흔히 “이중 지불 문제(Double-Spend Problem)” [1]라고 불립니다. 따라서 합의 문제는 네트워크 내에서 전역적으로 인정되는 거래 집합이 단 하나만 존재해야 한다는 요구로 요약될 수 있습니다.

유용성은 다소 추상적인 문제인데, 일반적으로는 분산 결제 시스템의 “유용성”으로 정의되며, 실제로는 대부분 시스템의 지연(latency) 문제로 단순화됩니다. 예를 들어, 정확성과 합의를 모두 충족하지만 거래 처리에 1년이 걸리는 분산 시스템은 명백히 실현 불가능한 결제 시스템입니다. 유용성의 추가적인 측면에는 정확성과 합의 과정에 참여하기 위해 요구되는 컴퓨팅 파워의 수준이나, 네트워크 내에서 사기를 당하지 않기 위해 최종 사용자가 갖추어야 하는 기술적 숙련도가 포함될 수 있습니다.

이러한 문제들 중 많은 부분은 현대 분산 컴퓨터 시스템이 등장하기 오래 전부터 “비잔틴 장군 문제(Byzantine Generals Problem)” [2]라는 문제를 통해 연구되어 왔습니다. 이 문제에서 여러 장군들은 각각 군대의 일부를 지휘하며, 전령을 보내 서로 협력하여 공격을 조율해야 합니다. 장군들이 낯선 적지에 있기 때문에 전령이 목적지에 도달하지 못할 수도 있고(분산 네트워크의 노드가 실패하거나 의도한 메시지 대신 손상된 데이터를 전송하는 것과 유사), 일부 장군은 배신자일 수 있으며 개별적으로 또는 집단적으로 음모를 꾸밀 수도 있습니다. 그 결과 충성스러운 장군들을 실패로 몰아가기 위한 거짓 작전 메시지가 도착할 수 있습니다(분산 시스템의 악의적 참여자가 사기 거래나 동일한 올바른 거래의 여러 버전을 네트워크에 수용시

키려 하는 것과 유사). 따라서 분산 결제 시스템은 일반적인 실패뿐만 아니라 네트워크의 여러 출처에서 동시에 발생할 수 있는 소위 “비잔틴 실패”에도 견고해야 합니다.

본 연구에서는 분산 결제 시스템의 한 구체적 구현인 리플 프로토콜(Ripple Protocol)을 분석합니다. 우리는 위에서 언급한 정확성, 합의, 유용성의 목표를 달성하기 위해 사용되는 알고리즘에 초점을 맞추며, 이 모든 목표가 (잘 알려진 사전 정의된 허용 오차 한계 내에서) 충족됨을 보입니다. 또한 네트워크 크기, 악의적 사용자 수, 메시지 전송 지연(latency)을 매개변수화할 수 있는 합의 과정의 시뮬레이션 코드를 제공합니다.

2. 정의, 형식화 및 기존 연구

우리는 리플 프로토콜의 구성 요소를 정의하는 것부터 시작합니다. 정확성, 합의, 유용성 속성을 증명하기 위해 먼저 이러한 속성을 공리(axioms)로 형식화합니다. 이러한 속성들이 모이면 합의(consensus)라는 개념, 즉 네트워크 내 노드들이 올바른 합의에 도달하는 상태가 형성됩니다. 이후 합의 알고리즘과 관련된 기존의 연구 결과를 강조하고, 마지막으로 우리 형식화 틀 안에서 리플 프로토콜의 합의 목표를 제시합니다.

2.1 리플 프로토콜 구성 요소

리플 네트워크를 설명하기 위해 다음 용어들을 정의합니다:

- **서버(Server):** 서버는 리플 서버 소프트웨어를 실행하는 모든 엔티티를 의미하며(단순히 자금을 송수신만 하는 리플 클라이언트 소프트웨어와는 대조됨), 합의 과정에 참여합니다.
- **원장(Ledger):** 원장은 각 사용자의 계좌에 있는 화폐의 양을 기록한 장부이며 네트워크의 “근본적 진실(ground truth)”을 나타냅니다. 원장은 합의 과정을 성공적으로 통과한 거래들로 반복적으로 갱신됩니다.
- **최종 마감 원장(Last-Closed Ledger):** 최종 마감 원장은 합의 과정에 의해 인준된 가장 최신 원장이며, 따라서 네트워크의 현재 상태를 나타냅니다.
- **열린 원장(Open Ledger):** 열린 원장은 노드의 현재 운영 상태를 나타내며(각 노드는 자체 열린 원장을 유지함), 특정 서버의 최종 사용자에게 의해 시작된 거래는 해당 서버의 열린 원장에 적용됩니다. 그러나 거래는 합의 과정을 통과하기 전까지는 확정되지 않으며, 합의 과정을 거쳐야 열린 원장이 최종 마감 원장이 됩니다.
- **고유 노드 목록(Unique Node List, UNL):** 각 서버 s 는 고유 노드 목록을 유지하며, 이는 합의를 결정할 때 s 가 질의하는 다른 서버들의 집합입니다. 합의를 결정할 때는 네트워크의 모든 노드가 아니라 s 의 UNL에 속한 서버들의 투표만 고려됩니다. 따라서 UNL은 집합적으로 보았을 때 s 가 네트워크를 속이려는 담합을 하지 않을 것이라고 “신뢰”하는 네트워크의 부분집합을 나타냅니다. 이때 “신뢰”의 정의가 UNL의 각 개별 구성원을 신뢰해야 함을 의미하지는 않는다는 점에 주의해야 합니다 (3.2절 참조).

- **제안자(Proposer)**: 어떤 서버든 합의 과정에 포함될 거래를 브로드캐스트할 수 있으며, 모든 서버는 새로운 합의 라운드가 시작될 때 모든 유효한 거래를 포함하려고 시도합니다. 그러나 합의 과정 동안 서버 s 는 자신의 UNL에 속한 서버들로부터의 제안만 고려합니다.
-

2.2 형식화

우리는 네트워크에서 정직하게 오류 없이 동작하는 노드를 **비결함(nonfaulty)** 이라고 부릅니다. 반대로 **결함 노드(faulty node)**란 오류를 경험하는 노드로, 이는 데이터 손상이나 구현 오류 등으로 인한 정직한 오류일 수도 있고, 비잔틴 오류와 같은 악의적 오류일 수도 있습니다. 거래를 검증하는 개념은 단순한 이진 결정 문제로 환원됩니다. 즉, 각 노드는 자신이 받은 정보에 기반해 0 또는 1의 값을 결정해야 합니다.

Attiya, Dolev, Gill (1984) [3]에 따라, 합의는 다음 세 가지 공리에 의해 정의됩니다:

1. (C1): 모든 비결함 노드는 유한 시간 내에 결정을 내린다.
 2. (C2): 모든 비결함 노드는 동일한 결정값에 도달한다.
 3. (C3): 모든 비결함 노드에게 0과 1은 모두 가능한 값이다. (이는 주어진 정보와 상관없이 모든 노드가 무조건 0 또는 1을 선택하는 사소한 해결책을 배제한다).
-

2.3 기존 합의 알고리즘

비잔틴 오류에도 불구하고 합의를 달성하는 알고리즘에 대한 연구는 상당히 많이 이루어져 왔습니다. 기존 연구에는 네트워크의 모든 참여자가 사전에 알려지지 않은 경우, 메시지가 비동기적으로 전달되는 경우(개별 노드가 결정을 내리는 데 걸리는 시간에 상한이 없는 경우), 강한 합의(strong consensus)와 약한 합의(weak consensus)의 구분이 존재하는 경우 등이 포함됩니다.

합의 알고리즘에 대한 중요한 기존 결과 중 하나는 Fischer, Lynch, Patterson (1985) [4]의 연구로, 비동기적 경우에는 단 한 개의 결함 프로세스만 존재하더라도 합의 알고리즘이 항상 종료하지 못할 가능성이 있음을 증명했습니다. 이는 수렴을 보장하기 위해(또는 적어도 비수렴이 반복되도록 하기 위해) 시간 기반의 휴리스틱이 필요함을 보여줍니다. 리플 프로토콜에서 이러한 휴리스틱은 3장에서 설명합니다.

합의 알고리즘의 강도는 일반적으로 허용 가능한 결함 프로세스의 비율로 측정됩니다. 비잔틴 장군 문제(동기성 및 참여자 사전 인지를 가정)에서는 네트워크의 최대 $(n - 1)/3$, 즉 33%가 악의적으로 행동하는 경우 이상은 허용할 수 없음을 증명할 수 있습니다 [2]. 그러나 이 해결책은 노드 간에 전달되는 메시지의 검증 가능한 진위(디지털 서명)를 요구하지 않습니다. 메시지의 위변조 불가능성이 보장된다면, 동기적 경우 훨씬 더 높은 결함 허용도를 가진 알고리즘이 존재합니다.

비동기적 경우 비잔틴 합의를 위해 더 복잡한 알고리즘들이 제안되었습니다. FaB Paxos [5]는 n 개의 노드로 구성된 네트워크에서 $(n - 1)/5$ 개의 비잔틴 오류를 허용하며, 이는 최대 20%의 노드가 악의적으로 담합하더라도 견딜 수 있음을 의미합니다. Attiya, Dovey, Gill [3]은 비동기적 상황에서 $(n - 1)/4$ 의 오류, 즉 네트워크의 최대 25%를 허용할 수 있는 단계적 알고리즘을 소개했습니다. 마지막으로 Alchieri 외 (2008) [6]는 BFT-CUP을 제시했는데, 이는 참여자가 사전에 알려지지 않은 비동기적 상황에서도 비

잔틴 합의를 달성하며, 최대 $(n - 1)/3$ 오류 허용 한계를 유지하지만 기반 네트워크의 연결성에 추가적인 제약을 둡니다.

2.4 형식적 합의 목표

본 연구의 목표는 리플 프로토콜에서 사용되는 합의 알고리즘이 각 원장 마감 시점마다 합의를 달성함을 보이는 것입니다(모든 거래가 거부되는 사소한 합의(trivial consensus)일지라도). 또한 비잔틴 오류가 존재 하더라도, 이러한 사소한 합의는 알려진 확률로만 발생함을 보입니다.

네트워크 내 각 노드는 신뢰하는 노드 집합(즉, 자신의 UNL에 속한 다른 노드들)의 제안에 대해서만 투표하며, 각 노드는 서로 다른 UNL을 가질 수 있습니다. 우리는 UNL의 구성과 관계없이 모든 노드 간에 단 하나의 합의만이 도달됨을 또한 보입니다. 이는 네트워크 내 “포크(fork)”를 방지하는 목표라고도 불리며, 포크란 두 개의 분리된 노드 집합이 각각 독립적으로 합의에 도달하여, 각 집합의 노드들이 서로 다른 최종 마감 원장을 관찰하는 상황을 의미합니다.

마지막으로 우리는 리플 프로토콜이 $(n - 1)/5$ 의 오류에도 이러한 목표를 달성할 수 있음을 보일 것입니다. 이는 문헌에서 제시된 가장 강력한 결과는 아니지만, 리플 프로토콜이 유용성을 크게 향상시키는 몇 가지 다른 바람직한 특징들을 보유하고 있음을 함께 제시합니다.

3. 리플 합의 알고리즘

리플 프로토콜 합의 알고리즘(Ripple Protocol Consensus Algorithm, RPCA)은 네트워크의 정확성과 합의를 유지하기 위해 모든 노드에 의해 몇 초마다 실행됩니다. 일단 합의가 달성되면, 현재 원장은 “마감(closed)”된 것으로 간주되어 최종 마감 원장이 됩니다. 합의 알고리즘이 성공적으로 작동하고 네트워크에 포크가 존재하지 않는다고 가정하면, 네트워크 내 모든 노드가 유지하는 최종 마감 원장은 동일합니다.

3.1 정의

RPCA는 라운드(round) 단위로 진행됩니다. 각 라운드에서:

- 초기 단계에서 각 서버는 합의 라운드 시작 이전에 보았으나 아직 적용되지 않은 모든 유효 거래(서버의 최종 사용자가 새로 시작한 거래, 이전 합의 과정에서 보류된 거래 등)를 모아 “후보 집합(candidate set)”이라 불리는 목록 형태로 공개합니다.
 - 각 서버는 자신의 UNL에 속한 모든 서버들의 후보 집합을 통합하고, 모든 거래의 진위 여부에 대해 투표합니다.
 - 일정 비율 이상의 “예” 투표를 받은 거래는(라운드가 더 있다면) 다음 라운드로 넘어가며, 충분한 표를 받지 못한 거래는 폐기되거나, 다음 원장에 대한 합의 과정 시작 시 후보 집합에 포함됩니다.
 - 최종 합의 라운드에서는 서버의 UNL 중 최소 80%가 해당 거래에 동의해야 합니다. 이 요건을 충족한 모든 거래는 원장에 적용되며, 그 원장은 마감되어 새로운 최종 마감 원장이 됩니다.
-

3.2 정확성

최대 허용 가능한 비잔틴 오류가 주어졌을 때, 합의 과정에서 결함 노드의 수가 그 한계를 초과하지 않는 한 사기 거래가 확정될 수 없음을 보여야 정확성이 달성됩니다. RPCA의 정확성 증명은 여기서 직접적으로 도출됩니다. 즉, 거래는 서버의 UNL 중 80%가 동의할 때만 승인되므로, UNL의 80%가 정직하다면 어떤 사기 거래도 승인되지 않습니다. 따라서 네트워크에서 n 개의 노드로 이루어진 UNL의 경우, 합의 프로토콜은 다음 조건이 충족되는 한 정확성을 유지합니다:

$$f \leq (n - 1)/5$$

여기서 f 는 비잔틴 오류의 수를 의미합니다. 사실 $(n - 1)/5 + 1$ 개의 비잔틴 오류가 존재하더라도 정확성은 여전히 기술적으로 유지됩니다. 이 경우 합의 과정은 실패하지만, 사기 거래가 확정되는 것은 불가능합니다. 실제로 잘못된 거래가 승인되려면 $(4n + 1)/5$ 개의 비잔틴 오류가 필요합니다. 우리는 이 두 번째 한계를 **약한 정확성(weak correctness)**의 한계로, 첫 번째 한계를 **강한 정확성(strong correctness)**의 한계로 정의합니다.

또한 모든 “사기” 거래가 위협이 되는 것은 아님을 주목할 필요가 있습니다. 예를 들어 사용자가 두 거래에서 동일한 자금을 이중 지출하려고 시도한다 해도, 두 거래 모두 합의 과정에서 확정될 수 있습니다. 그러나 첫 번째 거래가 적용된 이후에는 자금이 더 이상 존재하지 않으므로 두 번째 거래는 실패합니다. 이러한 견고성은 거래가 결정론적으로 적용되며, 합의 과정이 네트워크의 모든 노드가 동일한 거래 집합에 대해 동일한 결정론적 규칙을 적용하도록 보장하기 때문에 가능합니다.

다른 분석을 위해, 임의의 노드가 담합하여 악의적 카르텔에 참여할 확률을 p_c 라고 가정해 봅시다. 이 경우 정확성의 확률 p^* 는 다음과 같이 주어집니다:

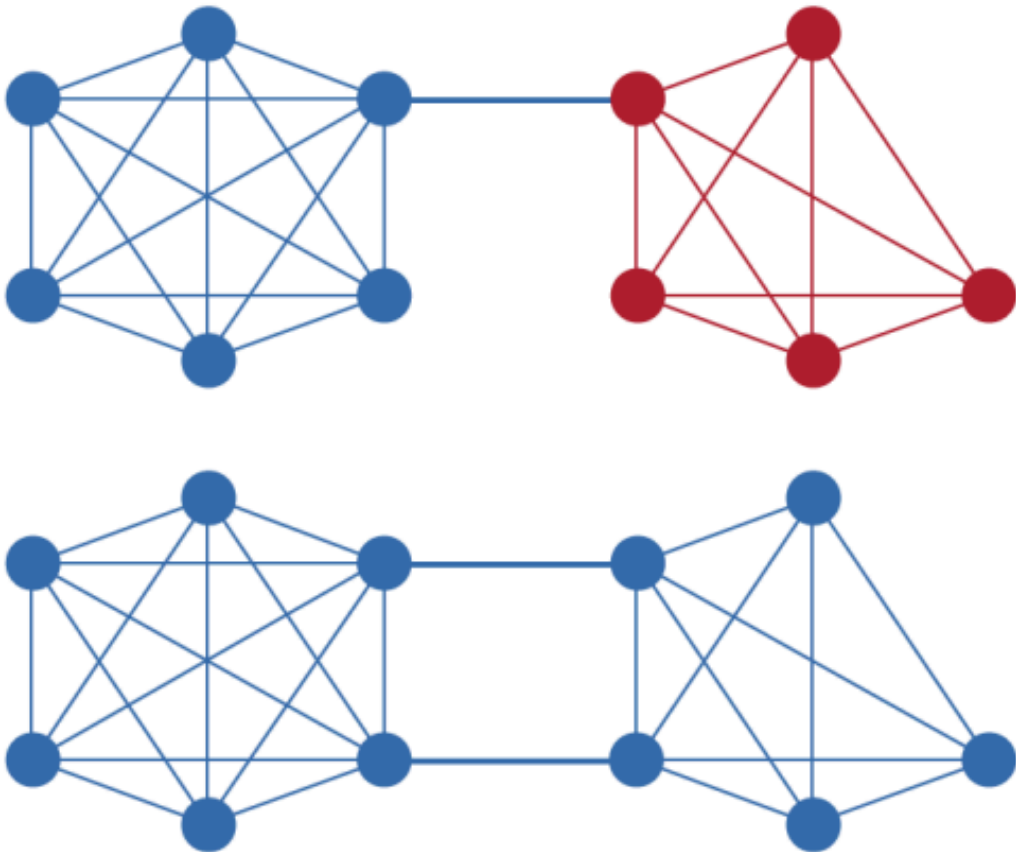
$$p^* = \sum_{i=0}^{\lfloor \frac{(n-1)}{5} \rfloor} \binom{n}{i} p_c^i (1 - p_c)^{n-i}$$

이 확률은 p_c 가 주어졌을 때 악의적 카르텔의 크기가 비잔틴 오류 허용 한계 이하로 유지될 가능성을 나타냅니다. 이 확률은 이항 분포이므로 p_c 가 20%를 초과하면 카르텔의 예상 크기가 네트워크의 20% 이상이 되어 합의 과정이 좌절됩니다. 실제로 UNL은 무작위로 선택되지 않고, p_c 를 최소화하기 위한 의도를 가지고 선택됩니다. 노드는 익명적이지 않고 암호학적으로 식별 가능하기 때문에, 대륙, 국가, 산업, 이념 등이 혼합된 노드들로 UNL을 구성하면 p_c 값은 20%보다 훨씬 낮아집니다. 예를 들어, 반명예훼손연맹(Anti-Defamation League)과 웨스트보로 침례교회(Westboro Baptist Church)가 담합하여 네트워크를 속일 확률은 확실히 20%보다 훨씬, 훨씬 작습니다. 심지어 UNL의 p_c 가 비교적 크더라도, 예를 들어 15%라 하더라도, UNL의 노드 수가 200개일 때 정확성 확률은 97.8%로 매우 높습니다.

UNL 크기에 따라 p_c 값이 달라질 때, 부정확성 확률이 어떻게 스케일링되는지를 보여주는 그래프적 표현은 그림 1에 나타나 있습니다. 여기서 세로축은 합의를 방해할 수 있는 악의적 카르텔의 확률을 나타내며, 따라서 값이 낮을수록 합의 성공 가능성이 높음을 의미합니다. 그림에서 볼 수 있듯이, p_c 가 10%에 달하더라도 UNL이 100개 노드를 넘어설 경우 합의가 좌절될 확률은 급격히 무시할 만한 수준으로 줄어듭니다.

3.3 합의성

합의성 요건을 만족하려면, UNL이 서로 다르더라도 모든 비결함 노드가 동일한 거래 집합에 대해 합의에 도달함을 보여야 합니다. 각 서버의 UNL은 다를 수 있기 때문에, 합의성은 정확성 증명만으로는 본질적으로 보장되지 않습니다. 예를 들어, UNL 구성에 제한이 없고 UNL 크기가 전체 네트워크 노드 수 n_{total} 의 $0.2 \times n_{total}$ 보다 크지 않다면 포크가 발생할 수 있습니다. 이는 단순한 예시(그림 2에 묘사됨)로 설명할 수 있습니다. UNL 그래프 내에 각각 $0.2 \times n_{total}$ 보다 큰 두 개의 클리크(cliques)가 있다고 가정합니다. 클리크란 각 노드의 UNL이 동일한 노드 집합으로 구성된 노드 집합을 의미합니다. 이 두 클리크가 구성원을 공유하지 않는다면, 각 클리크는 독립적으로 올바른 합의에 도달할 수 있으며, 이는 합의성을 위반합니다. 그러나 두 클리크의 연결성이 $0.2 \times n_{total}$ 를 초과한다면, 합의에 필요한 80% 동의 임계값에 도달할 수 없게 되어 포크는 더 이상 발생할 수 없습니다.



[그림 2. 두 UNL 클리크 간의 포크를 방지하기 위해 필요한 연결성의 예시.]

합의를 증명하기 위해 요구되는 연결성의 상한은 다음과 같이 주어집니다:

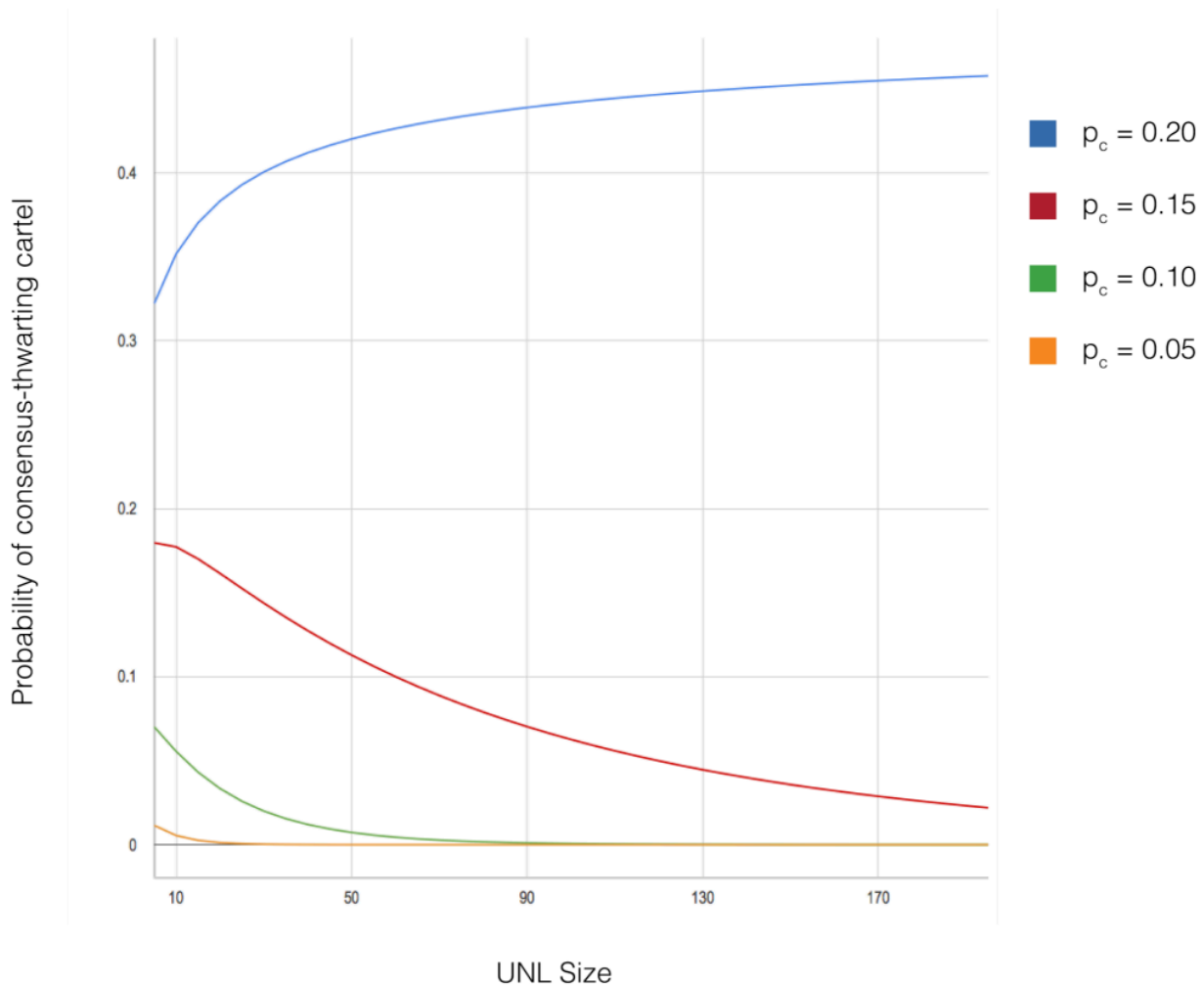
$$|UNL_i \cap UNL_j| \geq \frac{1}{5} \max(|UNL_i|, |UNL_j|) \quad \forall i, j$$

이 상한은 UNL이 클리크와 같은 구조를 가진다고 가정합니다. 즉, 노드들이 자신이 속한 집합 내 다른 노드들을 UNL에 포함시키는 경우입니다. 이 상한은 어떤 두 클리크도 상충하는 거래에 대해 합의에 도달할 수 없음을 보장합니다. 왜냐하면 합의에 필요한 80% 임계값에 도달하는 것이 불가능하기 때문입니다. 간접적인 UNL 간 연결까지 고려하면 더 엄격한 한계치를 도출할 수 있습니다. 예를 들어 네트워크 구조가 클리크형이 아닐 경우, 모든 노드의 UNL이 더욱 얹혀 있어 포크를 달성하는 것이 훨씬 더 어려워집니다.

흥미로운 점은 교차하는 노드의 성격에 대해 아무런 가정도 하지 않는다는 것입니다. 두 UNL의 교집합에는 결함 노드가 포함될 수 있지만, 교집합 크기가 합의를 보장하는 데 필요한 한계보다 크고 전체 결함 노드 수가 강한 정확성을 만족하는 데 필요한 한계보다 작다면, 정확성과 합의성은 모두 달성됩니다. 즉, 합의성은 비결함 노드의 교집합 크기가 아니라 단순히 노드 교집합 크기에만 의존합니다.

3.4 유용성

유용성의 많은 구성 요소는 주관적이지만, 실제로 증명 가능한 한 가지 요소는 수렴성입니다. 즉, 합의 과정이 유한 시간 내에 종료됨을 의미합니다.



[그림 1. 악의적 카르텔이 합의를 저지할 수 있는 확률을, UNL 크기의 함수와 p_c 값(UNL의 어떤 구성원이 다른 노드와 담합할 확률)의 관점에서 나타낸 것. 여기서 세로축 값이 낮을수록 합의 성공 확률이 더 높음을 의미한다.]

3.4.1 수렴성

우리는 수렴성을 RPCA가 원장에 대해 강한 정확성(strong correctness)을 가진 합의에 도달하고, 그 원장이 최종 마감 원장으로 되는 시점으로 정의합니다. 기술적으로 약한 정확성(weak correctness) 또한 알고리즘의 수렴을 의미하지만, 이는 단지 사소한 경우의 수렴일 뿐입니다. 왜냐하면 공리 C3가 위배되며, 어떤 거래도 확정되지 않기 때문입니다. 위의 결과에서 알 수 있듯이, $(n - 1)/5$ 개의 비잔틴 오류까지는 강한 정확성이 항상 달성되며, UNL 연결성 조건(식 3)이 충족되는 한 네트워크 전체에서 단 하나의 합의만이 달성됩니다. 따라서 남은 것은 이 두 조건이 모두 충족될 때 합의가 유한 시간 내에 도달함을 보이는 것입니다.

합의 알고리즘 자체는 결정론적이며, 합의가 종료되기 전까지 사전에 정해진 라운드 수 t 를 가집니다. 그리고 이 시점에서 현재 거래 집합은 승인 혹은 비승인으로 선언됩니다(비록 이때 어떤 거래도 요구되는 80% 동의에 도달하지 못해 합의가 단지 사소한 합의에 불과할 수 있더라도). 따라서 알고리즘 종료의 제한 요소는 노드 간의 통신 지연(latency)입니다. 이 값을 제한하기 위해 노드의 응답 시간이 모니터링되며, 지연이 사전 설정된 한계 b 를 초과하는 노드는 모든 UNL에서 제거됩니다. 이는 합의가 tb 의 상한으로 종료됨을 보장하지만, 중요한 점은 위에서 설명한 정확성과 합의성의 조건이 최종 UNL, 즉 모든 제거된 노드가 제외된 이후의 UNL에 대해 충족되어야 한다는 것입니다. 초기 UNL이 조건을 만족하더라도 지연 때문에 일부 노드가 네트워크에서 제거되면, 정확성과 합의성 보장은 자동으로 유지되지 않으며, 새로운 UNL 집합에 대해서 다시 충족되어야 합니다.

3.4.2 휴리스틱 및 절차

앞서 언급했듯이, 리플 네트워크의 모든 노드에는 합의 알고리즘이 수렴함을 보장하기 위해 지연 시간 한계 휴리스틱이 적용됩니다. 이 외에도 RPCA의 유용성을 제공하는 몇 가지 추가적인 휴리스틱 및 절차가 존재합니다.

- 각 합의 라운드에서 모든 노드가 초기 후보 집합을 제안할 수 있도록 의무적인 2초의 윈도우가 존재합니다. 이는 각 합의 라운드에 최소 2초의 지연을 도입하지만, 합리적인 지연을 가진 모든 노드가 합의 과정에 참여할 수 있음을 보장합니다.
- 각 합의 라운드에서 투표는 원장에 기록되며, 이를 통해 일부 흔히 발견되는 악의적 행위들을 하는 노드들을 식별하고 네트워크에서 제거할 수 있습니다. 예를 들어 모든 거래에 “아니오”로 투표하는 노드, 지속적으로 합의에 의해 검증되지 않는 거래를 제안하는 노드 등이 해당됩니다.
- 모든 사용자에게는 p_c 를 최소화하도록 선택된 기본 UNL이 제공됩니다(3.2절 참조). 사용자는 자신만의 UNL을 선택할 수 있고 또 그래야 하지만, 이 기본 노드 목록은 경험이 없는 사용자조차도 매우 높은 확률로 정확성과 합의성을 달성하는 합의 과정에 참여할 수 있도록 보장합니다.

- 네트워크 포크를 방지하기 위해 네트워크 분할 감지 알고리즘이 사용됩니다. 합의 알고리즘은 최종 마감 원장의 거래가 올바름을 보장하지만, 연결성이 낮은 네트워크 하위 구간에서 둘 이상의 최종 마감 원장이 존재할 가능성을 배제하지는 않습니다. 이러한 분할이 발생했는지 식별하기 위해 각 노드는 자신의 UNL 활성 멤버 수를 모니터링합니다. 이 수가 갑자기 사전 설정된 임계값 아래로 떨어지면, 분할이 발생했을 가능성이 있습니다. UNL의 큰 부분이 일시적인 지연을 겪는 경우 발생할 수 있는 오탐을 방지하기 위해, 노드는 거래를 처리하거나 투표하지 않더라도 여전히 합의 과정에 참여하고 있음을 선언하는 “부분 검증(partial validation)”을 게시할 수 있습니다. 이는 단절된 서브네트워크에서 별도의 합의 과정에 참여하는 것이 아님을 나타냅니다.
- RPCA를 단 한 번의 합의 라운드로 적용할 수도 있지만, 유용성은 여러 라운드를 통해 얻을 수 있습니다. 각 라운드는 요구되는 최소 동의 비율을 점차 증가시키며, 마지막 라운드에서 80% 요건이 적용됩니다. 이러한 라운드는 일부 지연 노드가 네트워크의 거래 속도를 병목시키는 경우 이를 탐지할 수 있게 합니다. 이러한 노드들은 낮은 요건의 초기 라운드에서는 따라잡을 수 있지만, 임계값이 증가함에 따라 뒤처지고 식별될 수 있습니다. 반면 단 한 번의 합의 라운드만 있는 경우, 80% 임계값을 충족하는 거래가 너무 적어 느린 노드조차 따라잡을 수 있어, 네트워크 전체의 거래 속도를 낮출 수 있습니다.

4. 시뮬레이션 코드

제공된 시뮬레이션 코드는 RPCA의 한 라운드를 보여주며, 네트워크 내 노드 수, 악의적 노드 수, 메시지 지연 시간 등 매개변수화 가능한 기능들을 포함합니다. 시뮬레이터는 완전한 불일치 상태(네트워크의 절반은 처음에 “예”를, 나머지 절반은 “아니오”를 제안)에서 시작한 후 합의 과정을 진행하며, 각 단계마다 노드들이 자신의 UNL 멤버들의 제안에 따라 제안을 조정하면서 네트워크 내 “예/아니오” 투표 수를 보여줍니다. 일단 80% 임계값이 도달되면 합의가 달성됩니다. 독자들은 “Sim.cpp”의 시작 부분에 정의된 상수들의 다양한 값을 실험해 보면서, 서로 다른 조건에서 합의 과정이 어떻게 작동하는지 익숙해지기를 권장합니다.

5. 논의

우리는 정확성, 합의성, 유용성의 조건을 충족하는 RPCA를 설명했습니다. 그 결과, 리플 프로토콜은 한 번의 합의 라운드가 완료되는 데 걸리는 몇 초 만에 안전하고 신뢰할 수 있는 거래를 처리할 수 있습니다. 이러한 거래들은 3장에서 제시한 한계 내에서 보안이 증명되며, 이는 비동기 비잔틴 합의에 대해 문헌에서 제시된 가장 강력한 결과는 아니지만, 빠른 수렴성과 네트워크 멤버십의 유연성을 가능하게 합니다. 이러한 특성들을 종합하면 리플 네트워크는 보안성과 신뢰성이 잘 이해된, 빠르고 저비용의 글로벌 결제 네트워크로 기능할 수 있습니다.

우리는 식 (1)과 (3)에서 제시된 한계가 충족되는 한 리플 프로토콜이 보안이 증명됨을 보였습니다. 그러나 이들은 최대치 한계일 뿐이며, 실제로 네트워크는 훨씬 덜 엄격한 조건에서도 안전할 수 있다는 점을 주목할 가치가 있습니다. 그러나 이러한 한계의 충족은 RPCA 자체에 내재된 것이 아니라, 모든 사용자의 UNL 관리에 달려 있음을 인식하는 것도 중요합니다. 모든 사용자에게 제공되는 기본 UNL은 이미 충분하지만, 사용자가 UNL을 변경하려면 위의 한계에 대한 이해를 기반으로 해야 합니다. 또한 전역 네트워크 구조를 모니터링하여 식 (3)의 한계가 충족되고 합의가 항상 유지됨을 보장할 필요가 있습니다.

우리는 RPCA가 분산 결제 시스템에 있어 중요한 도약이라고 믿습니다. 낮은 지연(latency)은 기존의 더 높은 지연 합의 방식에서는 어렵거나 불가능했던 다양한 유형의 금융 거래를 가능하게 하기 때문입니다.

6. 감사의 글

리플 랩스(Ripple Labs)는 리플 프로토콜 합의 알고리즘 개발에 참여한 모든 이들에게 감사를 표합니다. 특히, 거래 집합에 대한 작업을 수행한 아서 브리토(Arthur Britto), 리플 프로토콜 합의 개념을 최초로 제시한 제드 맥칼렐(Jed McCaleb), 합의 과정에서 “합의 실패는 연기 합의”라는 측면을 연구한 데이비드 슈워츠(David Schwartz)에게 특별히 감사드립니다. 또한 이 논문을 준비하고 검토하는 데 기여한 노아 영스(Noah Youngs)에게도 감사를 표합니다.

참고 문헌

- [1] Nakamoto, Satoshi. “Bitcoin: A peer-to-peer electronic cash system.” Consulted 1.2012 (2008): 28.
 - [2] Lamport, Leslie, Robert Shostak, and Marshall Pease. “The Byzantine generals problem.” ACM Transactions on Programming Languages and Systems (TOPLAS) 4.3 (1982): 382-401.
 - [3] Attiya, C., D. Dolev, and J. Gill. “Asynchronous Byzantine Agreement.” Proc. 3rd. Annual ACM Symposium on Principles of Distributed Computing. 1984.
 - [4] Fischer, Michael J., Nancy A. Lynch, and Michael S. Paterson. “Impossibility of distributed consensus with one faulty process.” Journal of the ACM (JACM) 32.2 (1985): 374-382.
 - [5] Martin, J-P., and Lorenzo Alvisi. “Fast byzantine consensus.” Dependable and Secure Computing, IEEE Transactions on 3.3 (2006): 202-215.
 - [6] Alchieri, Eduardo AP, et al. “Byzantine consensus with unknown participants.” Principles of Distributed Systems. Springer Berlin Heidelberg, 2008. 22-40.
-