리플

백서 번역본

The Ripple Protocol Consensus Algorithm

David Schwartz david@ripple.com; Noah Youngs nyoungs@nyu.edu; Arthur Britto arthur@ripple.com

본 번역본은 가상자산 투자자들을 위한 참고용입니다. 본 번역본은 투자 권유를 목적으로 만들어지지 않았으며 원문을 단순히 번역한 것으로, 내용의 정확성은 원문 작성 주체에 달려 있으며 크립토닷컴 코리아는 투자 결과에 대하여 어떠한 책임도 지지 않습니다. 본 번역본의 내용에 대한 자세한 문의는 백서 원문을 참고하시길 바랍니다.

개요

여러 Byzantine Generals Problem을 해결하기 위한 합의 알고리즘이 존재하지만, 분산 결제 시스템과 관련하여 많은 알고리즘이 네트워크 내 모든 노드가 동기적으로 통신해야 하는 요구사항으로 인해 높은 지연(latency)을 겪습니다. 본 연구에서는 이러한 요구사항을 우회하여 더 큰 네트워크 내에서 집합적으로 신뢰할 수 있는 서브 네트워크를 활용하는 새로운 합의 알고리즘을 제안합니다. 이러한 서브 네트워크에서 요구되는 “신뢰”는 실제로 최소화될 수 있으며, 원칙적인 구성원 선택으로 더 줄일 수 있음을 보입니다. 또한, 전체 네트워크에서 합의를 유지하기 위해 필요한 최소 연결성(minimal connectivity)에 대해서도 설명합니다. 그 결과, 비잔틴 실패에 직면했을 때에도 강인성을 유지하면서 낮은 지연을 자랑하는 합의 알고리즘이 탄생하였습니다. 이 알고리즘의 Ripple 프로토콜 내 구현을 소개합니다.

1. 서론

분산 합의 시스템에 대한 관심과 연구는 최근 몇 년 동안 현저히 증가했으며, 중심적인 초점은 분산 결제 네트워크에 있습니다. 이러한 네트워크는 중앙화된 출처에 의해 통제되지 않는 빠르고 저비용의 거래를 가능하게 합니다. 이러한 시스템의 경제적 이점과 단점은 그 자체로 많은 연구를 필요로 하지만, 본 연구는 모든 분산 결제 시스템이 직면해야 하는 기술적 도전 과제에

초점을 맞추고 있습니다. 이러한 문제는 다양하지만, 우리는 이를 세 가지 주요 범주로 그룹화합니다: 정확성, 합의, 그리고 유용성.

정확성이라는 것은 분산 시스템이 올바른 거래와 사기 거래를 구별할 수 있어야 한다는 의미입니다. 전통적인 신뢰 설정에서는 기관 간의 신뢰와 거래가 실제로 주장하는 기관으로부터 온 것임을 보장하는 암호화 서명을 통해 이 작업이 수행됩니다. 그러나 분산 시스템에서는 네트워크 내 모든 구성원의 신원이 알 수 없기 때문에 그러한 신뢰가 존재하지 않습니다. 따라서 정확성을 보장하기 위해 대체 방법을 활용해야 합니다.

합의는 분산 회계 시스템에서 단일 글로벌 진실을 유지하는 문제를 의미합니다. 정확성 문제와 유사하지만, 차이점은 네트워크의 악의적인 사용자가 사기 거래를 생성할 수는 없지만 (정확성을 위반하는 경우), 서로 인지하지 못하는 여러 개의 올바른 거래를 생성할 수 있다는 점입니다. 이러한 거래들이 결합되어 사기 행위를 일으킬 수 있습니다. 예를 들어, 악의적인 사용자가 계좌에 각각의 구매를 개별적으로 처리할 수 있는 충분한 자금만 있지만 동시에 두 개의 구매를 시도할 경우, 각 거래는 개별적으로는 올바르지만 동시에 분산 네트워크 전체가 두 거래 모두를 인식하지 못하는 상황에서는 명확한 문제가 발생하게 되며, 이는 일반적으로 "이중 지출 문제(Double-Spend Problem)"라고 불립니다. 따라서 합의 문제는 네트워크에 전 세계적으로 인식되는 하나의 거래 집합만 존재해야 한다는 요구사항으로 요약할 수 있습니다.

유용성은 약간 더 추상적인 문제로, 분산 결제 시스템의 "유용성"으로 일반적으로 정의되지만, 실제로는 대부분 시스템의 지연(latency)으로 단순화됩니다. 예를 들어, 올바르고 합의된 분산 시스템이지만 거래를 처리하는 데 1년이 걸리는 시스템은 명백히 실용적이지 않은 결제 시스템입니다. 유용성의 추가적인 측면으로는 정확성과 합의 과정에 참여하는 데 필요한 컴퓨팅 파워의 수준이나 네트워크에서 사기를 피하기 위해 최종 사용자가 요구되는 기술적 숙련도가 포함될 수 있습니다.

이러한 문제들은 현대의 분산 컴퓨터 시스템이 등장하기 오래 전부터 "비잔틴 장군 문제(Byzantine Generals Problem)"라는 문제를 통해 탐구되었습니다. 이 문제에서는 군대의 일부를 통제하는 여러 장군이 각자 메시지를 보내어 공격을 조율해야 합니다. 장군들이 낯선 적대적인 지역에 있기 때문에, 전령이 목적지에 도달하지 못할 수 있습니다 (분산 네트워크의 노드들이 실패하거나 의도된 메시지 대신 손상된 데이터를 보낼 수 있는 것과 유사합니다).

문제의 추가적인 측면은 일부 장군들이 배신자일 수 있으며, 개인적으로 또는 함께 음모를 꾸며 충실한 장군들에게 실패할 수 있는 잘못된 계획을 생성하는 메시지를 보낼 수 있다는 점입니다 (마찬가지로, 분산 시스템의 악의적인 구성원들이 사기 거래를 수용하도록 시스템을 설득하거나 이중 지출을 초래할 수 있는 동일한 진실한 거래의 여러 버전을 제출하려 할 수 있습니다). 따라서 분산 결제 시스템은 표준 실패뿐만 아니라 네트워크 내 여러 출처에서 조정되고 발생할 수 있는 이른바 "비잔틴" 실패에도 견고해야 합니다. 이 작업에서는 분산 결제 시스템의 특정 구현인 Ripple 프로토콜을 분석합니다. 우리는 정확성, 합의, 유용성이라는 목표를 달성하기 위해 사용된 알고리즘에 초점을 맞추며, 이러한 목표가 모두 충족됨을 (필요하고 미리 정해진 허용 오차 한도 내에서, 잘 이해된 대로) 보여줍니다. 추가로, 네트워크 크기, 악의적인 사용자 수, 메시지 전송 지연 시간을 매개변수화할 수 있는 합의 프로세스를 시뮬레이션하는 코드를 제공합니다.

2. 정의, 형식화 및 이전 연구

우리는 Ripple 프로토콜의 구성 요소를 정의하는 것부터 시작합니다. 정확성, 합의 및 유용성 속성을 증명하기 위해, 먼저 이러한 속성들을 공리로 형식화합니다. 이러한 속성들이 함께 묶여서 합의라는 개념을 형성합니다: 네트워크의 노드들이 올바른 합의에 도달하는 상태입니다. 이후에는 합의 알고리즘과 관련된 이전 결과들을 강조하고, 마지막으로 우리의 형식화된 프레임워크 내에서 Ripple 프로토콜의 합의 목표를 설명합니다.

2.1 Ripple 프로토콜 구성 요소

Ripple 네트워크의 설명을 시작하기 위해 다음 용어를 정의합니다:

-(Server): 서버는 Ripple 서버 소프트웨어를 실행하는 모든 엔티티를 말합니다 (사용자가 자금을 송수신하는 Ripple 클라이언트 소프트웨어와는 다릅니다). 서버는 합의 과정에 참여합니다.

-원장 (Ledger): 원장은 각 사용자의 계정에 있는 통화의 양을 기록하며 네트워크의 “기본 진리”를 나타냅니다. 원장은 합의 과정을 성공적으로 통과한 거래로 반복적으로 업데이트됩니다.

-최종 종료 원장 (Last-Closed Ledger): 최종 종료 원장은 합의 과정을 통해 비준된 최신 원장으로, 네트워크의 현재 상태를 나타냅니다.

-열린 원장 (Open Ledger): 열린 원장은 노드의 현재 운영 상태를 나타냅니다 (각 노드는 자신의 열린 원장을 유지합니다). 사용자가 시작한 거래는 해당 서버의 열린 원장에 적용되지만, 거래가 합의 과정을 통과할 때까지 최종적인 것으로 간주되지 않으며, 이 시점에 열린 원장이 최종 종료 원장이 됩니다.

-고유 노드 목록 (Unique Node List, UNL): 각 서버 \(s \)는 고유 노드 목록을 유지합니다. 이는 합의를 결정할 때 \(s \)가 쿼리하는 다른 서버들의 집합을 의미합니다. 합의를 결정할 때 \(s \)의 UNL의 다른 구성원들의 투표만 고려되며 (네트워크의 모든 노드를 고려하는 것이 아님), 따라서 UNL은 네트워크의 하위 집합을 나타냅니다. 이 집합은 집합적으로 \(s \)가 네트워크를 사기치려는 시도를 하지 않을 것이라고 “신뢰하는” 노드들로 구성됩니다. 이 “신뢰”의 정의는 UNL의 개별 구성원들 각각에 대한 신뢰를 요구하지 않습니다 (섹션 3.2 참조).

-제안자 (Proposer): 모든 서버는 합의 과정에 포함될 거래를 방송할 수 있으며, 새로운 합의 라운드가 시작될 때 모든 유효한 거래를 포함하려고 시도합니다. 그러나 합의 과정 동안, 서버 \(s \)는 \(s \)의 UNL에 있는 서버들로부터의 제안만을 고려합니다.

2.2 형식화

"비장애 노드(nonfaulty node)"라는 용어는 네트워크에서 정직하게 오류 없이 동작하는 노드를 지칭합니다. 반대로, "장애 노드(faulty node)"는 오류가 발생하는 노드로, 이는 데이터 손상, 구현 오류 등으로 인한 정직한 오류일 수도 있고, 악의적인 오류(Byzantine errors)일 수도 있습니다. 거래를 검증하는 개념을 단순한 이진 결정 문제로 축소합니다: 각 노드는 자신에게 제공된 정보를 바탕으로 0 또는 1의 값을 결정해야 합니다.

Attiya, Dolev, 및 Gill, 1984 [3]에서처럼, 우리는 합의를 다음 세 가지 공리로 정의합니다:

1. (C1): 모든 비장애 노드는 유한한 시간 내에 결정을 내린다.
2. (C2): 모든 비장애 노드는 같은 결정 값을 도달한다.
3. (C3): 0과 1은 모든 비장애 노드에 대해 가능한 값이다. (이 공리는 모든 노드가 제공된 정보에 관계없이 0 또는 1을 결정하는 자명한 해결책을 제거합니다.)

2.3 기존 합의 알고리즘

비잔틴 오류(Byzantine errors) 상황에서 합의를 달성하는 알고리즘에 대한 많은 연구가 진행되었습니다. 이러한 기존 연구에는 네트워크의 모든 참가자가 사전에 알려지지 않는 경우, 메시지가 비동기적으로 전송되는 경우(개별 노드가 결정을 내리는 데 걸리는 시간에 제한이 없음), 강력한 합의와 약한 합의의 개념 구분이 포함됩니다.

기존 합의 알고리즘에 대한 중요한 결과 중 하나는 Fischer, Lynch, 및 Patterson, 1985 [4]의 연구로, 비동기적인 경우에 합의 알고리즘의 비종료(non-termination)는 항상 가능하다고 증명합니다. 이는 수렴(convergence) 또는 적어도 반복적인 비수렴(non-convergence) 상황을 보장하기 위해 시간 기반의 휴리스틱이 필요하다는 것을 소개합니다. Ripple 프로토콜에 대한 이러한 휴리스틱은 섹션 3에서 설명합니다.

합의 알고리즘의 강도는 일반적으로 허용할 수 있는 장애 프로세스의 비율로 측정됩니다. 비잔틴 장군 문제(Byzantine Generals problem)에 대한 해결책은 (n-1)/3의 비잔틴 오류를 허용할 수 있으며, 이는 네트워크의 33%가 악의적으로 행동할 수 있음을 의미합니다 [2]. 이 해결책은 노드 간에 전달된 메시지의 검증 가능한 진위 여부(디지털 서명)를 요구하지 않습니다. 만약 메시지의 위조 불가능성에 대한 보장이 가능하다면, 동기화된 경우 훨씬 높은 오류 허용도를 가진 알고리즘이 존재합니다.

비동기적 상황에서 비잔틴 합의를 위한 여러 복잡한 알고리즘이 제안되었습니다. FaB Paxos [5]는 n개의 노드로 구성된 네트워크에서 (n-1)/5의 비잔틴 실패를 허용하여, 네트워크의 최대 20%의 노드가 악의적으로 협력하는 것을 허용합니다. Attiya, Doyev, 및 Gill [3]은 비동기적 상황을 위한 단계별 알고리즘을 소개하며, 이는 (n-1)/4의 실패를 허용하여 최대 25%의 네트워크를 허용합니다. 마지막으로, Alchieri et al., 2008 [6]은 BFT-CUP를 제시하며, 이는 참가자가 알려지지 않은 경우에도 비동기적 상황에서 비잔틴 합의를 달성하며, (n-1)/3의 최대 실패 허용도를 가지지만, 기반 네트워크의 연결성에 추가적인 제한이 있습니다.

2.4 형식적 합의 목표

이 작업의 목표는 Ripple 프로토콜이 사용되는 합의 알고리즘이 각 원장 마감 시 합의를 달성할 것임을 보여주는 것입니다(합의가 모든 거래가 거부되는 자명한 합의일지라도). 또한 비잔틴 오류가 발생하는 경우에도 자명한 합의가 알려진 확률로만 도달할 것임을 보여주고자 합니다.

네트워크의 각 노드가 신뢰할 수 있는 노드 집합(자신의 UNL에 있는 다른 노드)의 제안에 대해서만 투표하고, 각 노드가 서로 다른 UNL을 가질 수 있기 때문에, 모든 노드 사이에서 오직 하나의 합의만이 도달할 수 있음을 보여줍니다. 이 목표는 네트워크에서 “포크(fork)”를 방지하는 것으로도 언급됩니다. 포크란 두 개의 분리된 노드 집합이 각각 독립적으로 합의를 도달하고, 각 노드 집합에서 서로 다른 마지막으로 닫힌 원장이 관찰되는 상황을 의미합니다.

마지막으로, Ripple 프로토콜이 (n-1)/5의 실패에도 이러한 목표를 달성할 수 있음을 보여줄 것입니다. 이는 문헌에서 가장 강력한 결과는 아니지만, Ripple 프로토콜이 여러 다른 바람직한 특성을 가지고 있어 그 유용성을 크게 향상시킨다는 점을 보여줄 것입니다.

3. Ripple 합의 알고리즘

Ripple 프로토콜 합의 알고리즘(RPCA)은 네트워크의 정확성과 합의를 유지하기 위해 모든 노드에 의해 몇 초마다 적용됩니다. 합의가 도달하면 현재 원장이 "닫힌" 것으로 간주되며, 마지막으로 닫힌 원장이 됩니다. 합의 알고리즘이 성공적으로 작동하고 네트워크에서 포크가 발생하지 않는다고 가정하면, 네트워크의 모든 노드가 유지하는 마지막으로 닫힌 원장은 동일할 것입니다.

3.1 정의

RPCA는 라운드 단위로 진행됩니다. 각 라운드에서:

- 초기에는, 각 서버는 합의 라운드 시작 전까지 본 모든 유효한 거래 중 이미 적용되지 않은 거래(서버의 최종 사용자가 시작한 새로운 거래, 이전 합의 과정에서 보류된 거래 등 포함)를 취합하여 "후보 집합(candidate set)"이라는 형태로 공개합니다.
- 각 서버는 자신의 UNL에 있는 모든 서버의 후보 집합을 통합하고, 모든 거래의 진위에 대해 투표합니다.
- 최소 비율 이상의 “예(yes)” 투표를 받은 거래는 다음 라운드로 전달되며, 충분한 투표를 받지 못한 거래는 폐기되거나 다음 원장에 대한 합의 과정 시작 시 후보 집합에 포함됩니다.
- 최종 합의 라운드는 서버의 UNL에서 80%의 최소 비율이 거래에 동의해야 합니다. 이 요구 사항을 충족하는 모든 거래는 원장에 적용되며, 해당 원장이 닫히고 새로운 마지막으로 닫힌 원장이 됩니다.

3.2 정확성

정확성을 달성하기 위해, 최대 비잔틴 오류 수를 고려했을 때, 합의 과정 중에 사기 거래가 확인될 수 없음을 보여야 합니다. 이는 오류 노드의 수가 허용 한계를 초과하지 않는 한에서만 가능합니다. RPCA의 정확성 증명은 다음과 같이 직접적으로 이루어집니다: 거래는 서버의 UNL의 80%가 동의할 때만 승인되므로, UNL의 80%가 정직하다면 사기 거래는 승인되지 않습니다. 따라서 네트워크의 UNL이 n 노드일 때, 합의 프로토콜은 다음 조건을 만족하는 한 정확성을 유지할 것입니다:

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

여기서 \(f \)는 비잔틴 오류의 수를 의미합니다. 실제로, (n-1)/5+1 비잔틴 오류에 직면하더라도 정확성은 여전히 유지됩니다. 합의 과정은 실패하겠지만, 사기 거래를 확인하는 것은 여전히 불가능합니다. 부정확한 거래가 확인되려면 (4n+1)/5 비잔틴 오류가 필요합니다. 이 두 번째 경계를 약한 정확성의 경계(bound for weak correctness)라고 부르며, 첫 번째 경계를 강한 정확성의 경계(bound for strong correctness)라고 부릅니다.

모든 "사기" 거래가 위협을 가하지는 않는다는 점도 주목할 필요가 있습니다. 예를 들어, 사용자가 두 개의 거래에서 자금을 이중 지출하려 시도하더라도, 두 거래가 모두 합의 과정 중에 확인되더라도 첫 번째 거래가 적용된 후에는 두 번째 거래가 실패하게 됩니다. 이는 자금이 더 이상 사용 가능하지 않기 때문입니다. 이 강건성은 거래가 결정론적으로 적용되며, 합의가 네트워크의 모든 노드가 동일한 거래 집합에 대해 결정론적 규칙을 적용하도록 보장하기 때문입니다.

조금 다른 분석을 위해, 임의의 노드가 음성적인 카르텔에 합류하기로 결정할 확률을 \(p_c \)라고 가정합시다. 그러면 정확성의 확률은 \(p^* \)로 주어지며, 다음과 같습니다:

$$p^* = \sum_{i=0}^{\lceil (\frac{n-1}{5}) \rceil} \binom{n}{i} p_c^i (1-p_c)^{n-i} \qquad (2)$$

이 확률은 비잔틴 오류의 최대 임계값 이하로 악의적인 카르텔(nefarious cartel)의 크기가 유지될 가능성을 나타냅니다. 이 가능성은 이항 분포를 따르므로, \(p_c \)가 20%를 초과하면 예상되는 카르텔의 크기가 네트워크의 20%를 초과하여 합의 과정을 저지할 수 있습니다. 실제로, UNL은

무작위로 선택되지 않고, p_c를 최소화하려는 의도를 가지고 선택됩니다. 노드는 익명이 아니고 암호학적으로 식별 가능하기 때문에, 대륙, 국가, 산업, 이념 등 다양한 조합의 노드로 UNL을 선택하면 p_c 값이 20%보다 훨씬 낮아집니다. 예를 들어, 반명예훼손연맹(ADL)과 웨스트보로 침례교회가 네트워크를 속이기 위해 협력할 확률은 확실히 20%보다 훨씬 작습니다. UNL의 p_c가 상대적으로 크다고 하더라도, 예를 들어 15%일 경우, UNL에 200개의 노드만 있어도 정확성의 확률은 매우 높습니다: 97.8%.

UNL 크기에 따른 정확성의 확률이 어떻게 변하는지를 나타내는 그래픽 표현은 그림 1에 나와 있습니다. 여기서 수직 축은 악의적인 카르텔이 합의를 방해할 확률을 나타내며, 따라서 낮은 값은 합의 성공 확률이 더 높다는 것을 의미합니다. 그림에서 볼 수 있듯이, p_c가 10%와 같이 높더라도, UNL의 크기가 100 노드를 초과하면 합의가 방해받을 확률은 매우 빠르게 무시할 수 있을 정도로 낮아집니다.

3.3 합의 (Agreement)

합의 요구 사항을 충족시키기 위해, 모든 비 오류 노드가 그들의 UNL에 관계없이 동일한 거래 집합에 대해 합의에 도달해야 함을 보여야 합니다. 각 서버의 UNL이 다를 수 있으므로, 합의는 정확성 증명만으로는 보장되지 않습니다.

예를 들어, UNL의 멤버십에 대한 제한이 없고, UNL의 크기가 전체 네트워크의 노드 수인 n_{total}의 0.2배보다 크지 않은 경우, 포크(fork)가 가능할 수 있습니다. 이는 그림 2에서 설명된 간단한 예제로 보여집니다: UNL 그래프 내에 두 개의 클리크(연결된 노드 집합)가 각각 $0.2 \times n_{\text{total}}$보다 큰 경우를 상상해 보십시오. 여기서 클리크란, 각 노드의 UNL이 동일한 노드 집합인 경우를 의미합니다. 이 두 클리크가 어떤 멤버도 공유하지 않기 때문에, 각 클리크가 독립적으로 올바른 합의에 도달할 수 있어 합의가 위반될 수 있습니다. 두 클리크의 연결성이 $0.2 \times n_{\text{total}}$을 초과하면, 클리크 간의 불일치가 80% 합의 임계값을 충족하지 못하도록 하여 포크가 더 이상 가능하지 않습니다.

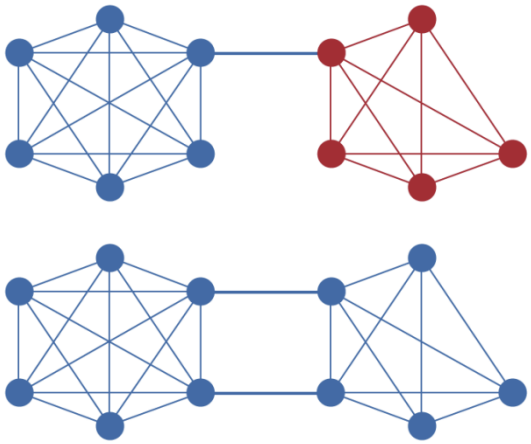

그림 2. 두 UNL 클리크 사이의 분기를 방지하기 위해 필요한 연결성의 예.

합의를 증명하는 데 필요한 상한선은 다음과 같습니다:

$$|UNL_i \cap UNL_j| \geq \frac{1}{5}\max(|UNL_i|, |UNL_j|) \forall i, j \quad (3)$$

이 상한선은 UNL의 클리크 유사 구조를 가정합니다. 즉, 노드들은 자신의 UNL에 다른 노드들이 포함된 집합을 형성합니다. 이 상한선은 두 개의 클리크가 상충되는 거래에 대해 합의에 도달할 수 없음을 보장합니다. 이는 합의에 필요한 80% 임계값에 도달하는 것이 불가능해지기 때문입니다. 간접적인 UNL 간의 연결을 고려할 때 더 엄격한 상한선이 가능합니다. 예를 들어, 네트워크 구조가 클리크 유사하지 않다면, UNL의 얽힘이 더 커지기 때문에 분기를 달성하기가 훨씬 더 어려워집니다.

흥미롭게도, 교차하는 노드의 성격에 대한 가정은 없습니다. 두 UNL의 교차점에는 결함이 있는 노드가 포함될 수 있지만, 교차점의 크기가 합의를 보장하는 데 필요한 상한선보다 크고, 결함이 있는 노드의 총 수가 강한 정확성을 보장하는 데 필요한 상한선보다 적다면, 정확성과 합의가 모두 달성됩니다. 즉, 합의는 결함이 없는 노드의 교차점 크기와는 무관하게 노드의 교차점 크기에만 의존합니다.

3.4 유용성

유용성의 많은 구성 요소는 주관적이지만, 실제로 증명 가능한 것 중 하나는 수렴입니다. 즉, 합의 프로세스가 유한한 시간 내에 종료된다는 것입니다.

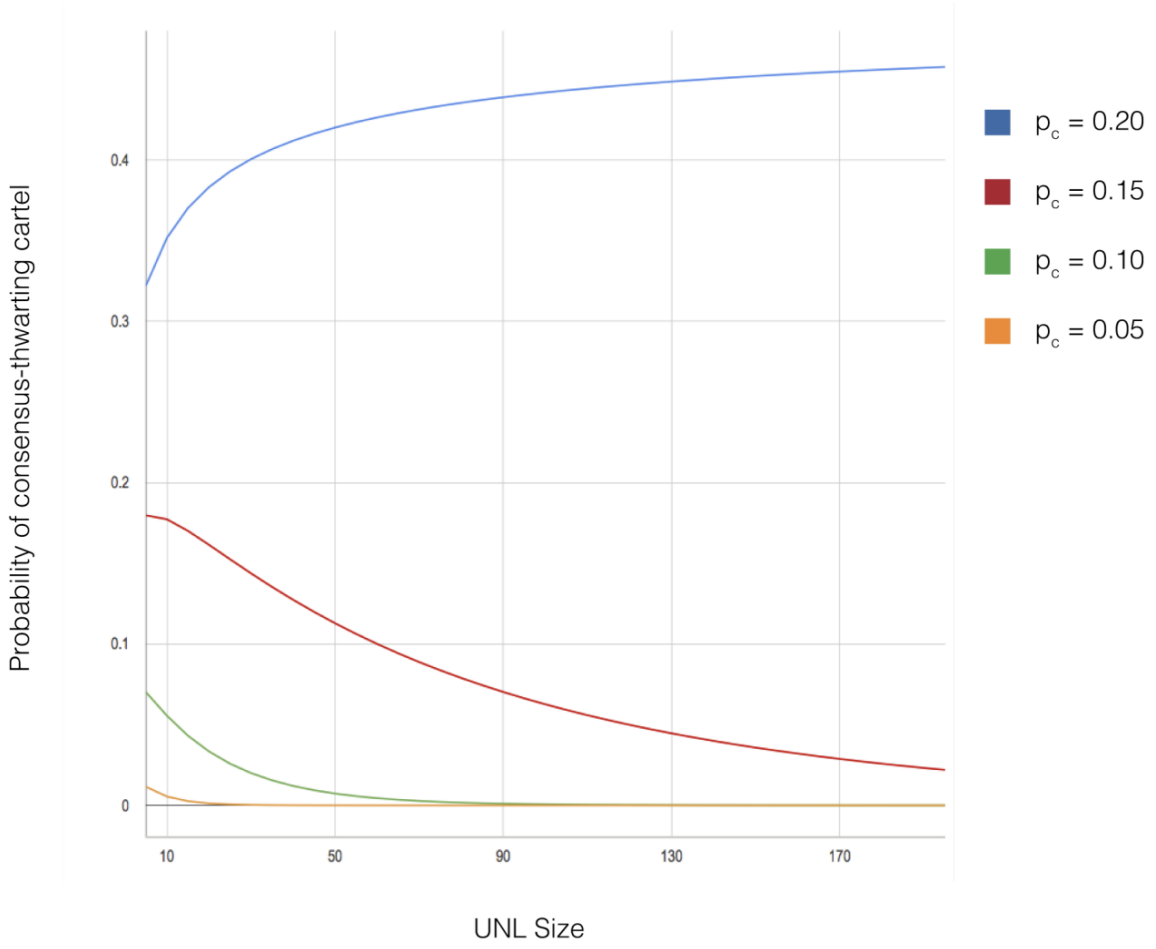

그림 1. UNL의 크기에 따른 악의적인 카르텔이 합의를 방해할 확률의 함수, 다양한 pc 값에 대해. 여기서 pc는 UNL의 어떤 구성원이 다른 이들과 공모할 확률을 나타냅니다. 여기서 낮은 값은 합의 성공 확률이 높다는 것을 의미합니다.

3.4.1 수렴

수렴은 RPCA가 원장에 대해 강한 정확성으로 합의에 도달하고, 그 원장이 마지막으로 닫힌 원장이 되는 지점을 정의합니다. 기술적으로 약한 정확성도 알고리즘의 수렴을 나타내지만, 이는 단순한 경우의 수렴일 뿐입니다. 이 경우 C3 조건이 위배되며, 거래는 결코 확정되지 않습니다. 위의 결과로부터, 강한 정확성은 최대 (n-1)/5 비잔틴 오류에 직면하더라도 항상 달성 가능하며, UNL 연결 조건이 충족되는 한 네트워크 전체에서 하나의 합의만 이루어진다는 것을 알고 있습니다 (방정식 3). 남은 것은 이러한 두 가지 조건이 모두 충족될 때, 합의가 유한 시간 내에 이루어지는 것을 보이는 것입니다.

합의 알고리즘 자체는 결정적이며, 합의가 종료되기 전에는 t의 사전 설정된 라운드 수가 있으며, 현재의 거래 세트가 승인되었거나 승인되지 않았다고 선언됩니다 (이 시점에서 거래가 80%의 합의 요구 사항을 초과하지 않고, 합의가 단순 합의일지라도). 알고리즘 종료의 제한 요소는 노드 간의 통신 지연입니다. 이 값을 제한하기 위해 노드의 응답 시간을 모니터링하며, 지연이 사전 설정된 경계 b를 초과하는 노드는 모든 UNL에서 제거됩니다. 이는 합의가 상한 tb 내에서 종료될 것임을 보장하지만, 위에서 설명한 정확성과 합의의 경계는 최종 UNL이 모든 제거될 노드가 제거된 후 충족되어야 한다는 점에 유의해야 합니다. 모든 노드의 초기 UNL이 조건을 충족하지만, 지연으로 인해 일부 노드가 네트워크에서 제거되면, 정확성과 합의 보장이 자동으로 유지되는 것이 아니라 새로운 UNL 세트에 의해 충족되어야 합니다.

3.4.2 휴리스틱 및 절차

위에서 언급한 바와 같이, Ripple 네트워크의 모든 노드에 대해 지연 경계 휴리스틱이 적용되어 합의 알고리즘이 수렴되도록 보장합니다. 또한 RPCA에 유용성을 제공하는 몇 가지 다른 휴리스틱과 절차가 있습니다.

- 모든 노드는 합의의 각 라운드에서 초기 후보 세트를 제안하는 필수 2초 윈도우가 있습니다. 이는 각 합의 라운드에 대해 2초의 하한을 도입하지만, 합리적인 지연을 가진 모든 노드가 합의 과정에 참여할 수 있도록 보장합니다.

- 각 합의 라운드에서 투표가 원장에 기록됨에 따라, 일부 일반적이고 쉽게 식별 가능한 악의적인 행동을 하는 노드는 플래그가 지정되고 네트워크에서 제거될 수 있습니다. 이러한 행동에는 모든 거래에 대해 "아니오"라고 투표하거나, 합의에 의해 검증되지 않은 거래를 지속적으로 제안하는 노드가 포함됩니다.

- 사용자는 기본적으로 제공되는 UNL 목록을 제공받습니다. 이 목록은 3.2절에서 설명한 대로 pc를 최소화하도록 선택됩니다. 사용자가 자신의 UNL을 선택할 수 있고 선택해야 하지만, 이 기본 노드 목록은 심지어 초보자도 높은 확률로 정확성과 합의가 달성되는 합의 과정에 참여할 수 있도록 보장합니다.

- 네트워크 분할 탐지 알고리즘도 사용되어 네트워크의 분기를 방지합니다. 합의 알고리즘은 마지막으로 닫힌 원장의 거래가 정확하다는 것을 인증하지만, 네트워크의 연결이 좋지 않은 서로 다른 하위 네트워크에서 여러 개의 마지막으로 닫힌 원장이 존재할 가능성을 금지하지 않습니다. 이러한 분할이 발생했는지 식별하기 위해, 각 노드는 자신의 UNL의 활성 멤버 크기를 모니터링합니다. 이 크기가 갑자기 사전 설정된 임계값 이하로 떨어지면 분할이 발생했을 가능성이 있습니다. UNL의 큰 부분이 일시적인 지연을 겪는 경우의 오탐지를 방지하기 위해, 노드는 "부분 검증"을 게시할 수 있으며, 이 경우 거래를 처리하거나 투표하지 않지만 다른 연결이 끊긴 하위 네트워크에서의 합의 과정과는 달리 여전히 합의 과정에 참여하고 있음을 선언합니다.

- RPCA를 한 라운드에서만 적용하는 것이 가능하지만, 여러 라운드를 통해 유용성을 얻을 수 있습니다. 각 라운드는 증가하는 최소 요구 합의 비율을 가지고 있으며, 최종 80% 요구 사항이 있는 라운드가 있습니다. 이러한 라운드는 네트워크의 거래 속도에 병목 현상을 일으키는 몇몇 지연 노드를 감지할 수 있습니다. 이러한 노드는 낮은 요구 사항의 라운드 동안 초기에는 따라갈 수 있지만, 임계값이 증가함에 따라 뒤처지게 되며 식별됩니다. 합의가 한 라운드일 경우, 80% 임계값을 초과하는 거래가 너무 적어서 느린 노드도 따라갈 수 있으며, 전체 네트워크의 거래 속도를 낮출 수 있습니다.

4. 시뮬레이션 코드

제공된 시뮬레이션 코드는 RPCA의 한 라운드를 보여주며, 매개변수화 가능한 기능들(네트워크의 노드 수, 악의적인 노드 수, 메시지의 지연 등)을 포함하고 있습니다. 시뮬레이터는 초기에는 완전한 불일치 상태로 시작합니다(네트워크의 절반의 노드는 처음에 "예"를 제안하고, 나머지 절반의 노드는 "아니오"를 제안합니다). 그 후, 합의 과정이 진행되며 각 단계에서 노드들이 UNL 구성원들의 제안에 따라 자신의 제안을 조정하는 과정에서 네트워크의 예/아니오 투표 수가 표시됩니다. 80% 임계값에 도달하면 합의가 이루어집니다. 독자들이 다양한 조건에서 합의 과정을 익힐 수 있도록, "Sim.cpp"의 시작 부분에서 정의된 상수들의 다양한 값을 실험해 보시기 바랍니다.

5. 논의

우리는 위에서 설명한 정확성, 합의, 유용성 조건을 만족하는 RPCA를 설명했습니다. 결과적으로 Ripple 프로토콜은 한 라운드의 합의가 완료되는 몇 초 만에 안전하고 신뢰할 수 있는 거래를 처리할 수 있습니다. 이 거래는 3절에 설명된 경계까지 증명된 안전성을 가지며, 비록 비동기 비잔틴 합의에 대한 문헌에서 가장 강력한 경계는 아닐지라도, 빠른 수렴과 네트워크 구성원에 대한 유연성을 허용합니다. 이들 특성이 결합되어 Ripple 네트워크는 잘 이해된 보안 및 신뢰성 속성을 갖춘 빠르고 저렴한 글로벌 결제 네트워크로 기능할 수 있습니다.

우리는 Ripple 프로토콜이 방정식 1과 3에서 설명된 경계를 만족하는 한 증명된 안전성을 가지지만, 이러한 경계는 최대 경계이며 실제로는 네트워크가 훨씬 덜 엄격한 조건에서도 안전할 수 있다는 점을 주목할 가치가 있습니다. 또한, 이러한 경계를 만족하는 것은 RPCA 자체에 내재된 것이 아니라, 모든 사용자의 UNL 관리가 필요하다는 점을 인식하는 것이 중요합니다. 모든 사용자에게 제공되는 기본 UNL은 이미 충분하지만, 사용자가 UNL을 변경해야 하는 경우에는 위의 경계를 인식한 상태에서 변경해야 합니다. 또한 방정식 3의 경계가 충족되도록 보장하고 합의가 항상 이루어질 수 있도록 글로벌 네트워크 구조에 대한 일부 모니터링이 필요합니다.

우리는 RPCA가 분산 결제 시스템에 있어 중요한 발전을 의미한다고 믿습니다. 낮은 지연 시간 덕분에 이전에는 다른 높은 지연 합의 방법으로 인해 어렵거나 불가능했던 많은 종류의 금융 거래가 가능해졌습니다.

6. 감사

Ripple Labs는 Ripple 프로토콜 합의 알고리즘 개발에 참여한 모든 사람들에게 감사를 표합니다. 특히, 거래 세트에 관한 작업을 한 Arthur Britto, 원래의 Ripple 프로토콜 합의 개념을 제안한 Jed McCaleb, 그리고 합의의 "합의하지 않음은 연기를 위한 동의" 측면에 대한 작업을 한 David Schwartz에게 감사를 드립니다. Ripple Labs는 또한 이 논문을 준비하고 검토한 Noah Youngs의 노력에 감사의 뜻을 전합니다.