콘스텔레이션
Constellation
(별자리)

백서 v0.1.
2018 년 5 월

번역 면책조항

번역자 : Blockchain Garage 팀
번역개시시점 : 2018 년 5 월 7 일

본 문서의 아래 내용 전체는 번역 개시시점 당시 Constellation Lab 의 홈페이지에 게재된 (https://constellationlabs.io/) 백서(White Paper v0.1.)를 내려받아 작성되었으며, 영어 버전을 이해하는 데에 유용한 정보를 담고 있습니다. 번역자는 영어 버전의 내용을 정확하게 제공될 수 있도록 최선을 다하지만, 본 문서가 영어 버전의 최신 상태를 유지하거나 진실만을 게시 혹은 번역 과정상의 오류 등에 대해 어떠한 종류의 보증도 제공하지 않습니다. 번역자는 아래의 모든 내용 전체에 대하여 정확성, 완전성, 시의 적절성, 법적 위반의 부재, 특정 합목적성과 이와 유사한 모든 명시적 묵시적 보증에 대해 법적 책임을 지지 않습니다.

투자자 및 이해관계인은 Constellation Lab 이 제공하는 최신 영어 버전을 기준으로 투자 의사를 결정하여야 하며, 그 어떠한 상황에서도 본 번역본과 관련하여 피해에 대한 책임에

대해서 권고를 받았는지 여부에 관계 없이 수익, 손실, 계약, 호의, 데이터, 정보, 수입, 예상되는 절감분, 비지니스 관계로부터 귀결되는 사항들을 포함하되 이에 국한되지 않고 그 어떤 종류의 부수적, 간접적, 결과적인 기타 피해에 대하여 법적 책임을 지지 않습니다.

중요 공지사항

이 문서 ("백서")는 어떤 정부 기관에서도 보증하지 않습니다. https://constellationlabs.io 에서만 사용할 수 있으며 Constellation Lab 의 사전 서면 동의없이 어떤 목적으로도 다른 사람에게 재배포, 복제 또는 양도하거나 일부 또는 전체를 게시 할 수 없습니다. 이 백서 또는 그 일부는 배포 또는 보급이 금지되거나 제한되는 모든 국가 또는 지역으로 전송되거나 전송되어서는 안됩니다. 이 백서를 소유하고있는 개인 또는 법인은 관련 법 또는 규제상의 제한 사항을 스스로 통지하고 준수해야하며 필요한 모든 전문적 조언을 구해야합니다. 이 백서를 접하는 개인 또는 법인 ("귀하"또는 "귀하의")은 이 요구 사항에 준수 할 것에 동의합니다.

Constellation

블록체인 마이크로서비스 운영체제

2017 년 11 월 25 일

초록

비트코인, 이더리움 등과 같은 현재의 블록체인 기술은 태생적으로 동기화 제한(Synchronous Limitation)을 가지고 있어 광범위한 소비자 어플리케이션들의 기술적 아키텍쳐로 채택하여 이를 기반으로 하기에 적합하지 않습니다. 또한 기존 합의 메커니즘 및 스마트 컨트랙트 아키텍처는 소비자애플리케이션들이 작동하는

데 필요한 수준까지 확장할 수 없습니다. 우리는 모바일 클라이언트와 같이 전체 노드(Full Node)들을 실행할 수 있는 내결함성(Fault Resistance), 수평적 확장성(Horizontally Scalable)을 지닌 분산 운영 체제를 제안합니다. 그리하여 우리는 비동기 ExtendedTrustChain, Proof-of-Meme 합의 모델, JVM 을 이용한 조합가능한 마이크로서비스로서의 스마트 컨트랙트를 사용하여 초기 관여자(Actor) 기반의 유한 상태 머신(Finite State Machine)으로 구현된 최신 서버리스(Serverless) 아키텍처에 암호학적으로 안전한 합의의 재구성을 제시합니다. 이 아키텍처는 높은 트랜잭션 처리량을 보장하여, Constellation 플랫폼에 소비자 등급의 분산 애플리케이션들 (decentralized application)을 구축할 수 있게 해줍니다. 우리는 이에 따라 최신 서버리스 아키텍쳐 위에 비동기화 확장신뢰체인인 Proof of MEME 컨센서스 모델을 사용하여 안전한 암호화된 합의 모델을 재구성하였습니다. 이 모델에서 구현된 스마트 컨트랙트는 초기에 구현된 유한 상태머신(Finite State Machine) [1] 기반의 관여자(Actor)일 뿐만 아니라 JVM(Java Virtual Machine)을 사용하여 작성가능한 마이크로서비스입니다. 이 아키텍처는 높은 트랜잭션 처리량을 보장하여, Constellation 위에 소비자 등급의 분산 애플리케이션들을 구축할 수 있게 해줍니다.

서론

분산화라고 하는 목표에도 불구하고, 암화화폐는 점점더 중앙화 되어가는 네트워크에 의해 운영되고 있으며, 네트워크의 보안을 제어하는 채굴 조직에 의해 보안화 되고 있습니다.[2] 이더리움과 같은 비트코인 네트워크 후발주자들의 등장으로 금융거래의 처리라고 하는 초기의 목표는 EVM 및 스마트 컨트랙트 시스템 로직에 의해 신뢰가 필요없는 분산 컴퓨팅을 제공할 수 있도록 확장되었습니다. 이러한 분산된 프로토콜들의 다양하고 창의적인 구현들이 존재하는 반면에, 기존의 주류 소프트웨어로부터 격리되어 있기 때문에 진입 장벽이 높습니다.[3] 때문에 기업체에서는 친숙하지 않은 코딩 언어와 디자인 패턴을 사용한 강력한 분산 어플리케이션을 개발하고 배포하며 유지보수하는데 매우 많은 비용과 시간이 소요됩니다. 또한 현재의 최첨단 퍼블릭 블록체인 조차도 애플리케이션을

[1] https://goo.gl/LHyC5M

[2] Digital Currency Group 'Bitcoin Scaling Agreement at Consensus 2017' medium.com/@DCGco

[3] J. Evans 'Blockchains are the new Linux, not the new internet' techcrunch.com

일반적인 기업의 사용 사례의 성능 요구에 맞게 확장하는 것은 불가능합니다. 이러한 기술적 과제들을 극복하기 위해 블록체인 기술은 수평적으로 확장 가능하고 비용측면에서 효과적이며 효율적인 분산 운영 체제로서 기능하는 자체 유지(self-sustaining) 프로토콜이 필요합니다.

비트코인은 금융 거래에 대한 분산된 합의 문제를 해결하기 위해 만들어졌지만 작업 증명(Proof of work) 으로 알려진 에너지 집약적인 합의 프로세스에 의존합니다. 이는 잘 개발된 엄선된 소수의 개인들로부터 수직적인 계산 파워를 모으고 있으므로 즉, 중앙화된 컴퓨팅 파워라고 할 수 있으며 네트워크 보안과 보상을 동일한 엄선된 소수의 손에 부여합니다.[4] 확장성 논의가 이 조직 안에서 확대되면서 비트코인 캐쉬 및 비트코인 골드와 같은 여러 가지 비트코인 포크 생성을 야기했습니다. 이러한 분리된 정치적 파벌은 매우 분열적이고 생태계에 해를 끼칩니다. 구식의 처리량 요구 사항(1MB 블록 크기 제한)과 시간 집약적인 알고리즘으로 인해 트랜잭션 시간과 트랜잭션 거래 수수료가 급증했습니다. 이더리움은 비트코인과 유사하나 ASIC 이 불가능한 작업 증명 합의 알고리즘을 사용하고 있으며, 지분 증명(Proof of stake) 합의 메카니즘인 캐스퍼(Casper)로 옮겨가고 있습니다. 지분 증명은 가장 많은 돈을 가진 사람들이 합의를 통해 네트워크의 상태를 선택할 수 있는 민주적 불균형 문제를 내포하고 있습니다. 또한 이더리움은 스마트 컨트랙트의 사용을 개척했으나, 동기화 실행으로 인해 일반 소비자를 위해 준비된 분산 어플리케이션들에게서 심각한 병목 현상이 발생합니다. 병행성(Concurrency)은 소비자 중심의 제품들을 제공하는 핵심이며, Constellation 은 마이크로 아키텍쳐라고 하는 고유의 스마트 컨트랙트를 통해 이를 달성합니다.

무엇이 Constellation 을 차별화 하는가?

블록체인 개발연합은 확장성 문제의 해결을 위하여 새로운 분산 원장 기술을 도입하고자 기대해 왔지만 현재까지 그 해답을 찾지 못하였습니다. 어떻게 하면 적은 시간과 적은 비용으로 더 많은 트랜잭션을 처리하고 더 많은 서비스를 수행할 수 있을것인가하는 문제는 여전히 숙제로 남아 있습니다. 따라서 우리는 MapReduce 에서 비슷하게 적용된 기술인 수평 확장 접근법을 제안합니다.

수평 확장성은 동시 프로그래밍 어플리케이션입니다. 이는 유저가 네트워크에 참여할수록 트랜잭션 처리량이 증가함을 의미합니다. MapReduce 는 계산을 간단한 오퍼레이션으로 쪼개서 처리하는데,

[4] 'Has Bitcoin Become Centralized?' www.trustnodes.com

이는 비동기 DAG(Directed Acyclic Graphic) 계산으로 공급 될 수 있기 때문에 동시프로그래밍의 효율성이 향상됩니다.

Constellation 프로토콜은 모바일 장치에 배포할 수 있는 가십(Gossip) 프로토콜로 알려진 P2P 레이어와 Extended Trust Chain 으로 알려진 수평적 확장성 블록체인 아키텍처를 구현합니다. Constellation 은 마이크로서비스 아키텍쳐라는 스마트 컨트랙트 개념을 도입하여 각 마이크로서비스의 서비스단 합의(SLA, Service Level Agreement)의 타입별 서명을 이해하는 방법을 통해 분산된 어플리케이션을 연결하거나 구성할 수 있습니다.

가십 프로토콜은 대규모 네트워크가 기존 블록체인 기술보다 훨씬 큰 규모로 전체 네트워크 상태를 통신할 수 있도록 합니다.[5] 이 유형의 네트워크에서 네트워크의 각 구성원은 이웃(Neighbors)을 추적하며, 새 메시지를 받으면 해당 메시지를 차례대로 모든 이웃들에 전파합니다. 이는 몇가지 흥미로운 수학적 특성을 가지고 있지만, 우리의 경우, 연결 장치의 큰 풀이 네트워크의 상태를 공유하게 되면 현재의 블록 체인 구현에서 알 수 없는 규모로 합의 구조를 구성할 수 있습니다.

트랜잭션 처리량(tps)			
비트코인	이더리움	아이오타(Tangle)	Hylochain(1200 노드의 경우-더 증가하지 않음)
3-4	15	500-800	4000-4800

Figure 1 : 비교가능한 블록체인간의 처리량 비교표

Extended Trust Chain 은 수평확장성을 허용하는 블록체인 컨센서스에 대한 접근방법 중 하나로써 프로토콜의 여러 측면을 비동기적으로 관리하기 위해 네트워크에 자체적인 책임과 역할을 가진 여러가지 노드 유형으로 이루어져 있습니다.

[5] R. van Renesse 'A Blockchain Based on Gossip?' zurich.ibm.com

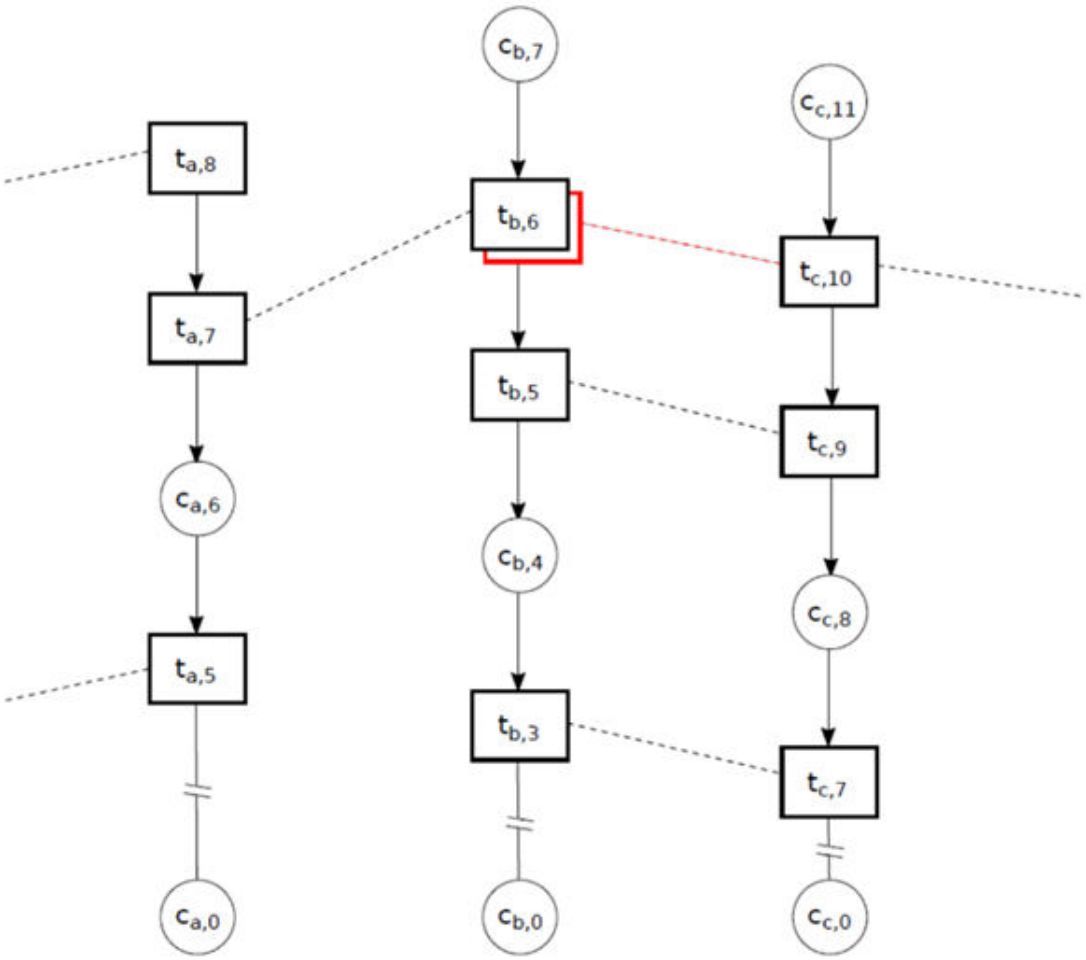

Figure 2. Extended Trust Chain 다이어그램, K.Cong 외

노드

특히, 트랜잭션을 보내고 구성원의 개별 체인을 호스트하는 기본 노드(basic node)가 있습니다 (트랜잭션이 만들어지고 선형 블록으로 해시 될 때 각 개별 체인이 네트워크의 다른 노드와 DAG 를 형성 함)

체크포인트 프로세스

트랜잭션 (버퍼를 채우고 체크 포인트 블록에 포함)에 대한 합의를 수행하는 노드 프로세스가 있습니다. 이러한 채집된 트랜잭션은 해쉬되어 메인 체인의 다음 블록이 됩니다.

유효성 검증자 프로세스

마지막으로 유효성 검사 프로세스를 실행하고 현재 체인 상태를 호스트하여 네트워크의 노드에 체인 상태 / 기록을 시드하는 유효성 검증자 노드가 있습니다.

EVM(Ethereum Virtual Machine) 과 JVM(Java Virtual Machine)

스마트 컨트랙트에 대한 MapReduce 작업과 같은 분산 애플리케이션들의 공식화는 이더리움에 플라즈마 업그레이드를 통해 제안되어 왔습니다.[6] 그러나 EVM 에 적용할 경우, 비용측면에서 효율적이고 지연 시간이 적은 방식으로 확장하기 위한 동기화 아키텍처의 동일한 단점은 여전히 남아있습니다. Solidity 프로그래밍 언어는 스마트 컨트랙트 개발에서 비결정성의 위험을 다루기 위해 개발되었습니다. 결정성 명령의 필요성은 스마트 컨트랙트 실행 비용을 계산하고 DDoS 공격 및 네트워크에 스팸 메시지를 보내는 것에 대한 예방책의 필요성으로부터 기인했습니다. 그러나 컴파일 된 바이트 코드가 체인에서 공증될 수 있고 JVM 에서 검증될 수 있다면, 결정적 컴파일은 더 불필요한 요구 사항이 될 것입니다.[7] 그러면 가스 소비의 감소, 가스 제한의 우회 및 마이크로서비스의 복잡성을 지닌 스마트 컨트랙트는 실현 가능해집니다.

기존의 마이크로서비스인 각 부분이 공통된 빌딩 블록들의 조합으로 분산 애플리케이션을 설계하는 것은 마치 레고를 조립하는 것과 같이 합리적입니다. 한 걸음 더 나아가 우리는 블록체인 회사들이 분산된 어플리케이션을 다른사람들이 재사용하여 더 복잡한 분산 어플리케이션을 만들기 위한 빌딩

[6] V. Buterin et al. https://plasma.io/plasma.pdf

[7] M. Hearn 'Corda: A distributed ledger' docs.corda.net

블록을 제공할 수 있도록 On-Chain 상호 운용성을 구현하였습니다. 진정으로 분산된 생태계는 모놀리틱(monolithic) 스마트 컨트랙트로는 존재할 수 없으며 그것에 의하여 만들어진 유형의 코드 재사용은 불가능하고 비효율적입니다.

JVM 을 분산 애플리케이션의 통로로 사용하는 것은 빅데이터 커뮤니티의 산업 표준입니다. 고객 애플리케이션들은 JVM 이 제공하는 손쉬운 프로비저닝을 통한 자동 확장 그룹 위에 마이크로서비스의 집합으로 전개됩니다. 이러한 시나리오에서는 수천 개의 노드 클러스터들이 AWS 가상 인스턴스에서 프로비저닝되고 자동 확장될 수 있습니다. 필요에 따라 리소스를 프로비저닝 할 수 있게 해주는 Constellation 의 마이크로서비스 아키텍처에서도 동적으로 확장 가능한 분산 애플리케이션 아키텍처를 제공하여 동일한 확장성이 구현될 수 있습니다.

이러한 유형의 분산 아키텍처는 본질적으로 함수형 프로그래밍이 제공하는 모듈성을 요구합니다. 따라서 우리는 어떤 JVM 언어로도 구현될 수있는 인터페이스를 구축하기 위해 Scala 프로그래밍 언어를 사용합니다. 마이크로서비스가 구체적인 유형들로 설계될 때 분산 애플리케이션들은 그것들의 소스 코드로 직접 구성될 수 있으므로 컴파일시에 유효성 검사가 가능합니다. 개발자들은 적용 가능한 유형의 스마트 컨트랙트들을 복합 애플리케이션들에 연결하여 새로운 기능을 만들 수 있습니다. 따라서 Constellation 의 계산 로직은 하나 이상의 마이크로서비스 또는 분산 애플리케이션으로 구성될 수 있습니다. Constellation 의 스마트 컨트랙트 인터페이스는 공변하고 반변하는 유형을 염두에두고 설계되기 때문에 호환되지 않는 마이크로서비스 또는 분산 애플리케이션을 혼합하면 즉시 알게 될 것입니다. 이런 사실 때문에 우리는 Cross Chain 유동성 문제에 대한 해결책(Polkadot 과 같은)을 기다리고있는 다른 블록체인들의 스마트 컨트랙트들을 포함하는 Constellation 을 만들 수도 있습니다.[8]

우리의 함수형 프로그래밍 방법론을 고려할 때, 우리의 합의 프로토콜은 hylomorphism(질료 형상론)의 범주 이론적 정의에 의해 기술될 수 있음에 주목할 가치가 있습니다.

[8] G. Wood et al. https://github.com/w3f/polkadot-whitepaper/raw/master/PolkaDotPaper.pdf

HyloChain - 합의 아키텍처

우리는 Constellation 의 기본 컨센서스 아키텍처가 질료형상론(hylomorphic)에 기반하기 때문에 우리의 합의 모델을 HyloChain 으로 명명했습니다. 합의 라운드에서는 이전 라운드의 결과 해시 블록을 가져 와서 일반 트랜잭션으로 트랜잭션 풀에 추가합니다. 트랜잭션 풀의 채우기는 전개(Unfold) 작업과 동형입니다. 이 체크 포인트 블록이 이전 라운드와 새 트랜잭션으로 채워지게 되면 이것은 해시됩니다. 이는 결집(Fold) 작업과 동형입니다.

HyloChain 의 정의

hylomorphism h : A → C 는 분리된 anamorphic 과 catamorphic 부분으로 정의될 수 있습니다. hylomorphism[9]의 정의는 다음과 같습니다.

$$h = [(c,\oplus),(g,p)] \tag{1}$$

Anamorphic 부분은 반복적인 적용 또는 전개(unfolding)를 통해 정의된 B 의 요소 목록 및 종료 조건을 제공하는 조건부 p : A → Boolean 을 통해 단일 함수 g : A → BA 로 정의될 수 있습니다.

Catamorphic 부분은 전개를 위한 초기값 c ∈ C 와 전개를 수행하는데 사용된 이진 연산자 ⊕ : B × C → C 의 조합으로 정의될 수 있습니다.

우리의 경우 메시지의 가시핑(Gossiping)은 우리의 Anamorphism 이며, 그 메시지들의 해싱과 이전 블록의 결과의 합은 우리의 Catamorphism 입니다.

[9] https://en.wikipedia.org/wiki/Hylomorphism_(computer_science)

n 번째 반복에 대해 연산자가 암호화 해시 함수 ⊕ = CHash 및 g n = (n,n−1) 및 p n = False(우리 체인은 종료 조건이 없으므로)로 정의됐다면, HyloChain 은 명확히 다음과 같이 정의되는데, 여기서 GenesisBlock 은 우리가 HyloChain 을 배포할 때 사용되는 우리의 시작 요소, 즉 제네시스(기원) 블록입니다.

$$\text{HyloChain} = [(\text{GenesisBlock},\oplus),(g,p)] \tag{2}$$

Extended Trust Chain 과의 관계

우리의 합의 아키텍처인 HyloChain 은 Extended Trust Chain 아키텍처를 토대로 구축됐으며 다음과 같은 방식으로 확장됩니다. Extended Trust Chain 에서 각 노드는 계정으로서 작동하고, 자체 체인의 기록을 유지하며, 유효성을 위해 트랜잭션 순서에 의존합니다(HashGraph 와 비슷함[10]). 트랜잭션들은 개시자와 상대방에 의해 서명되고 네트워크로 (가십을 통해)브로드캐스트됩니다. 대표자(Delegate)들은 체크 포인트 블록 내에서 트랜잭션에 대한 합의를 수행하고 그들의 평판에 따라 선발됩니다. 합의 결과가 다시 브로드캐스트 되고 다음 블록에 정상 트랜잭션으로 추가됩니다[11].

[10] L BAIRD et al. http://www.swirlds.com/downloads/SWIRLDS-TR-2016-01.pdf

[11] Kelong Cong, et al. https://repository.tudelft.nl/islandora/object/uuid:86b2d4d8-642e-4d0f-8fc7-d7a2e331e0e9?collection=education

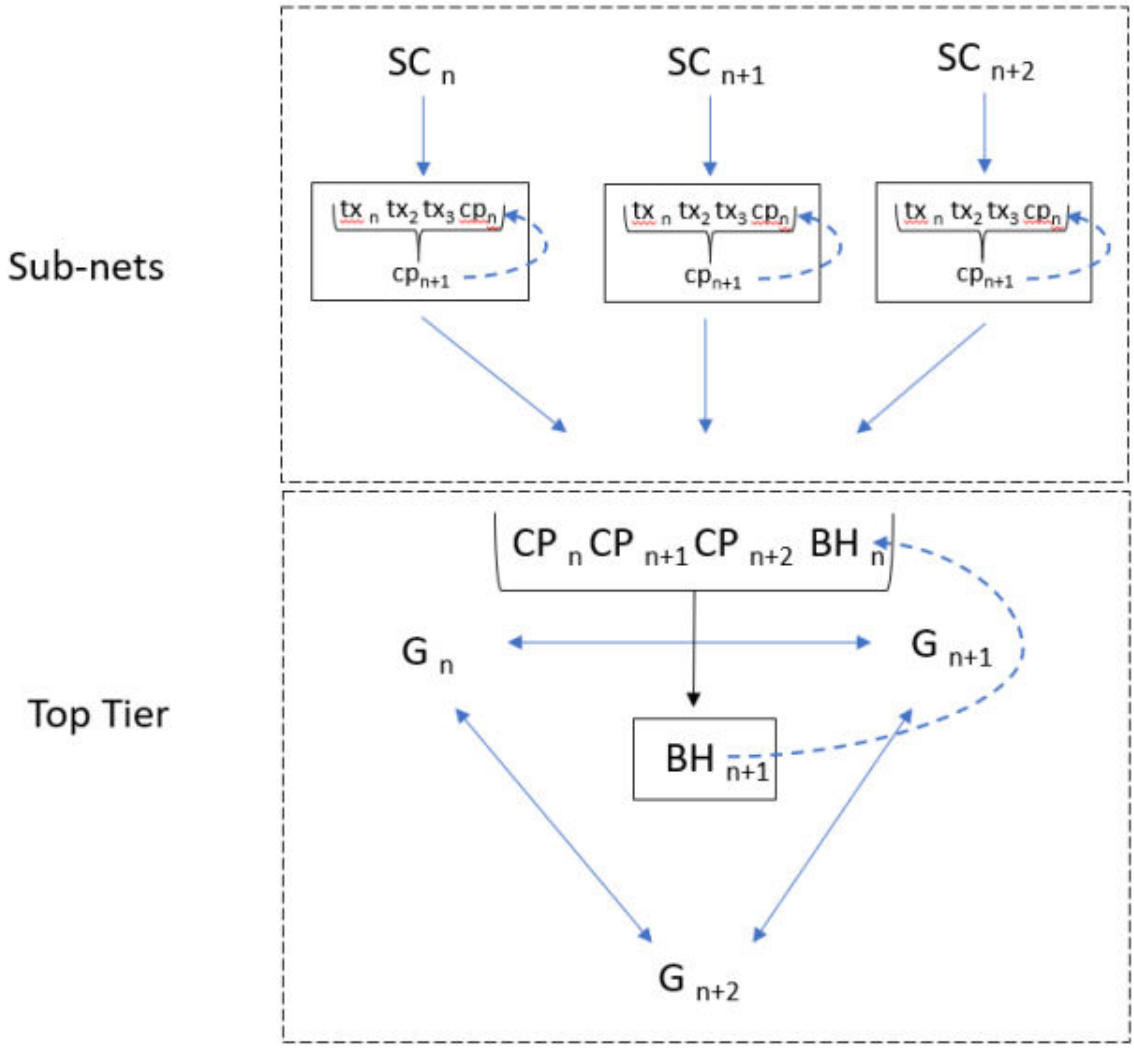

Figure 3. HyloChain[12]

결함 허용(Fault Tolerance)

Constellation 의 HyloChain 의 결함 허용은 전형적인 Byzantine 합의 모델을 따릅니다. 상대방이 네트워크 합의의 통제를 가져갈 확률은 1/2 에 근접하며 합의 참가자들의 그것에 대한 통제는

[12]트랜잭션은 서브넷에서 Delegate(SC, Starcluster)에 의해 생성된 Locality Sensitive Hash Block(CP, Check Point)에 해싱이 되며, 이렇게 생성된 Locality Sensitive Hash Block 은 다시 부모넷에서 Validator(G, Galaxy)에 의해 해싱되어 Block(BH, Black Hole)에 저장된다. (역자주)

1/3[13]에 근접합니다. 네트워크의 보안을 보장하기 위해 우리는 결함 허용을 조정하기 위한 특정 매개변수들을 선택하기 위하여 이러한 경계들을 사용할 수 있으며 트랜잭션 처리량을 최대화하고 대기 시간을 줄일 수 있습니다.

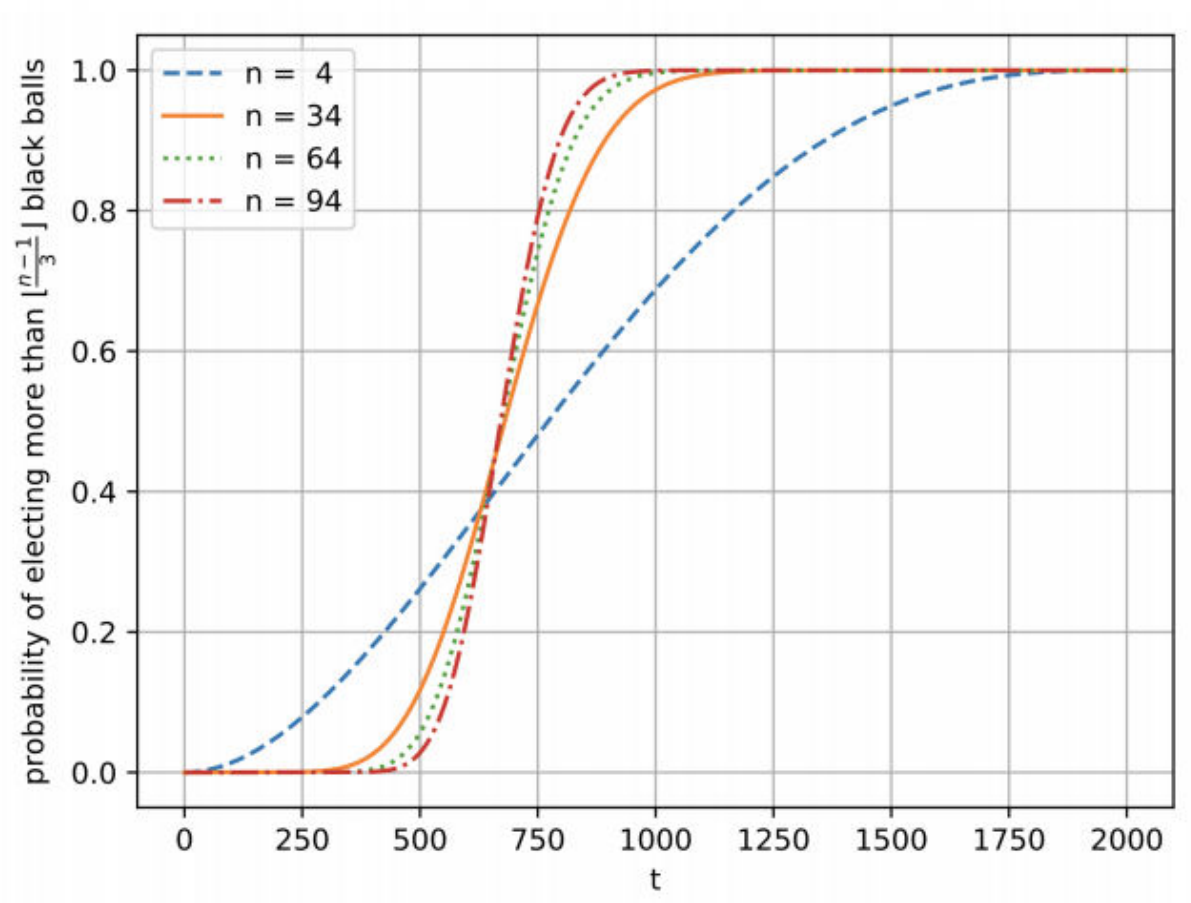

Figure 4 : 모집단 크기가 2000 에 고정되어 있고 t 의 다른 값 (t 는 byzantine 행위자의 수)에 대해 (초기하학적 분포의 설명에서 나온 것처럼 검은 색 볼로 표시되는) (n-1) / 3 이상을 선택할 확률, K.Cong 외

또한 우리의 Byzantine 합의 절차(우리는 Honeybadger BFT 에서 사용되는 M. Ben-Or 등이 구현 한 ACS 를 사용하고 있음)와 거래의 이중 서명을 기반으로 네트워크를 포크하는 것은 불가능합니다. 부정확한 합의의 유일한 결과는 체크 포인트 블록에서의 개별 거래들에 대한 검열, 즉 합의 도중

[13] M. Pease, R. shostak and l. Lamport, 'Reaching agreement in the presence of faults', journal of the ACM(JACM), vol 27, no.2

거래 유실입니다. 이로 인해 자본 손실이 발생하지는 않으며 유실의 완화는 간단히 그냥 네트워크에 다시 보내면 됩니다.

처리량의 관계는 수평적으로 확장할 수 있는 능력이 주어진 참가자들의 수에 대해 선형이지만, 결함 허용의 선형 증가는 통신 비용의 다항식(polynomial) 증가를 초래합니다[14]. 따라서 제한된 결함 허용과 트랜잭션 확인 시간으로 최대 처리량을 보장하는 합의 매개 변수들을 결정해야만 합니다. 그러한 범위 내에서 네트워크를 어떻게 확장해야할까요? 우리의 접근 방식은 네트워크의 블록체인을 서브 넷(Figure 1)들의 계층으로 구성하여 각자가 자체 합의를 수행하는 것입니다. 그 결과 서브넷들은 이러한 체크 포인트 블록들을 일반적인 트랜잭션으로 처리하는 상위 계층으로 올라가게 됩니다. 이 접근법은 Chain Fibers (검증자를 사용하는 경우도 마찬가지로)와 매우 비슷하지만, 트랜잭션 풀의 크기를 증가시키기 위해 Locality Sensitive Hashing 의 차원수 감소 기술을 적용한다는 점에서 다릅니다. 이 아키텍처의 효과는 Honeybadger BFT 에서 합의 알고리즘이 대표자(delegate) 당 O (1)로 감소 될 수 있다는 사실을 이용하여, 노드 당 O (n^3) 또는 2 차 노드의 평균 사례로부터 합의의 복잡성을 줄이는 것이므로[15], 우리의 복잡성은 다음과 같은 관계가 성립합니다.

$$O(n) \rightarrow O(m) : m << n \tag{3}$$

따라서 이것은 서브넷 내의 노드들의 수에 대하여 부 선형적입니다.

이 경우 우리는 이러한 체크 포인트 블록을 Locality Sensitive Hash Block[16]으로 적용합니다. 각 서브넷은 Locality Sensitive Hash 를 만들어 부모넷(Parent Net)으로(아래에 설명된 Galaxies) 보냅니다. 그런 다음 이 부모넷은 Locality Block Hash 를 다음 합의의 일반적인 트랜잭션으로 처리합니다. 네트워크의 성능을 높이기 위해서는 각 노드가 자체 트랜잭션 기록을 추적하여

[14] pp 50, K. Cong, et al.

[15] M. Ben-Or and R. El-Yaniv. Resilient-optimal interactive consistency in constant time. Distributed Computing, 16(4):249?262, 2003

[16] Locality Sensitive Hash Block : Figure 3 에서 Subnet 내부의 각 Block 으로 합의의 복잡성을 줄여 블록체인의 속도를 높이는 핵심적인 개념 (역자 주)

서브넷에서 이웃노드의 선택은 대기 시간의 최소화에만 의존해야 합니다.

각 규모의 레벨에서 가십(Gossiped) 메시지 전달의 동일한 자체 유사(self-similar) 아키텍처가 채택되어, 동일한 노드가 노드들에 의해 제공되는 리소스의 변경만으로 각 수준에 참여할 수 있습니다.

Constellation 의 모든 노드들은 "sleepy"상태가 될 수 있습니다. 즉, 언제든지 네트워크에 가입하거나 탈퇴 할 수 있습니다. 새 노드들이 네트워크에 가입하면 그들의 자원들이 서브넷에 할당됩니다. 위에서 설명한 대로 트랜잭션 처리량 및 암호 보안 임계값 (조력자 수 대 참가자 수)에 도달하면 유입 노드들은 새 서브넷을 형성해야 합니다. 따라서 구성원들이 네트워크를 떠나고 가입할 때 새 서브넷이 동적으로 할당됩니다. 아래에 요약된 이 자체 유사(self-similar) 구조 및 대표자(delegate) 선발 모델의 목표는 네트워크 리소스를 기반으로 동적 자동 확장을 가능하게 하여 일관된 처리량을 허용하는 것입니다. 이 아키텍처가 분산 컴퓨팅과 같은 복잡한 시스템에서 결함 허용에 적용 할 수있는 것으로 유명한 확장제한 없는(scale-free) 네트워크의 고유한 전력 법칙을 드러낸다는 것을 보여주는 것은 매우 쉽습니다[17].

[17] Ravasz, E.; Barabasi (2003). “Hierarchical organization in complex networks”. Phys. Rev. E. 67: 026112.

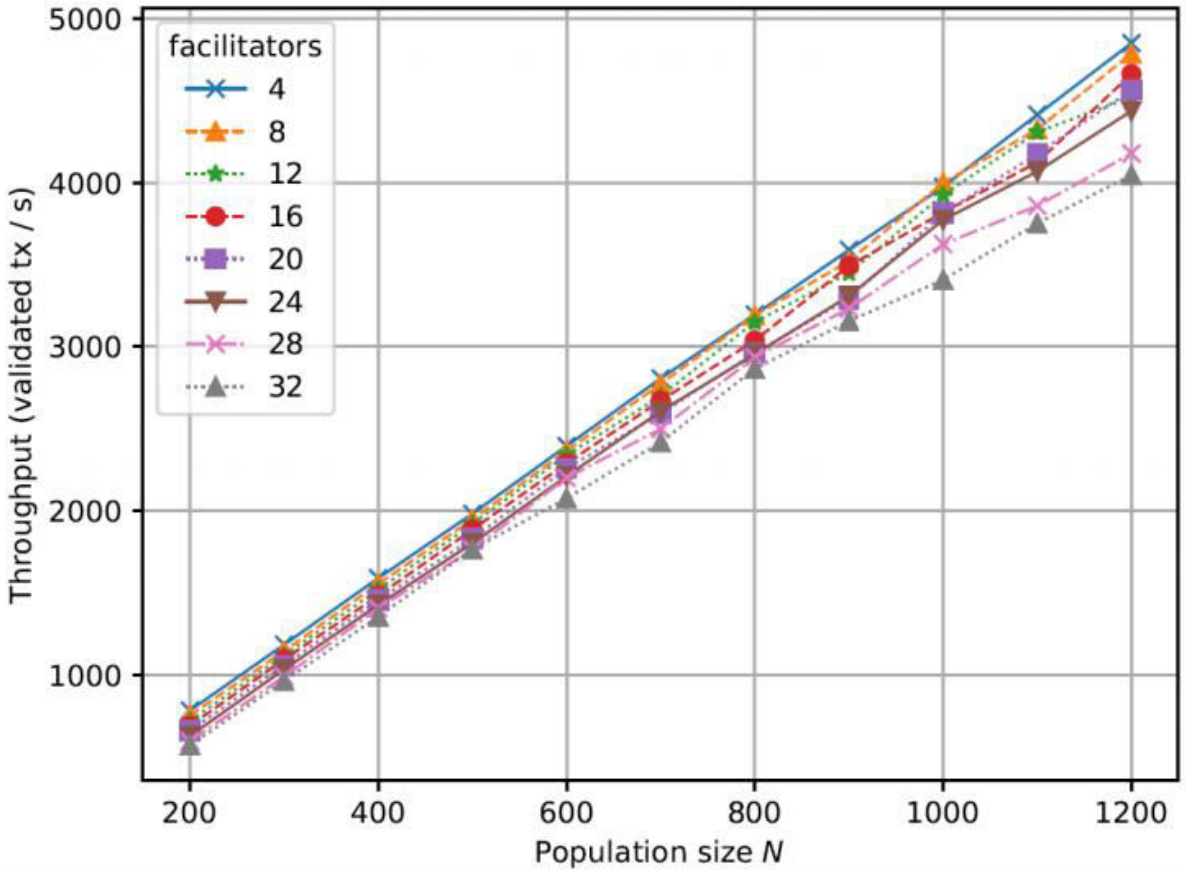

Figure 5. 모집단 크기 대비 HyloChain 처리량, K.Cong 외

대표자 선출 모델 : 밈의 증명(Proof-of-Meme)[18]

TrustChains 의 내결함성은 대표자를 선출하는 평판시스템을 통해 개선할 수 있습니다[19]. 그러므로 우리는 노드의 역사적인 참여정도를 대표자 선출에 결합할 수 있는 분산된 합의 방법인 "Proof-of-Meme"를 제시합니다. 여러분이 아는 바와 같이, 밈(Meme)은 모방할 수 있는 생각이나 퍼질 수 있는 행동을 나타내는 단위입니다. 밈은 우리의 경우에 Constellation 내에서 보상을 받거나 노드의 전반적인 평판을 개선하기 위해서 모방되어야 하는 호의적인 행동으로 표현할 수 있습니다. 밈의

[18] W. Meldman-Floch, Blockchain Cohomology. https://arxiv.org/pdf/1805.07047v1.pdf

[19] pp 50, K. Cong. et al.

증명은 자본적 권위주의인 지분증명(Proof of Stake) 방식에 비해 평등한 능력주의를 표방합니다.

우리에게 있어 밈은 각 노드의 계정에 대응하는 특징 벡터로서, 가장 단순한 경우로 결정론적 머신러닝 알고리즘의 입력값으로 사용되는 소수 값의 행렬입니다. 우리는 REGRET 의 발자취(군생하는 사회를 위한 평판 모델)를 따라, 네트워크에서 노드의 평판을 기술하기 위해 특징의 온톨로지를 사용합니다. 기술적으로 이것은 특징 공간의 tensor product 입니다[20]. 고유의 특징공간으로 온톨로지(밈)를 사용하는 해당 결정론적 머신러닝 모델은 노드의 평판점수 및 상기 노드가 합의에 참여하도록 선택될 확률을 전이적으로 결정합니다.

밈은 평판점수의 기초이며 평판점수는 합의를 위해 선정될 확률을 나타냅니다.

확률론적 대표자 선출은 비동기 합의 메커니즘에서 BFT 를 개선하고자 하는 맥락에서 광범위하게 연구되어 왔습니다.[21] 특히 GURU 에 관한 연구에서(A. Biryukov 외) 비잔틴 합의에서 1/3 악의적 인 노드의 일반적인 내결함성이 최대 1/2 (일부 경우에는 1/2 이상)까지 향상 될 수 있음을 보여주었습니다. 우리는 밈의 증명(Proof of Meme)에서 대표자 선출을 위해 GURU 의 검증 프레임 워크를 통합하였습니다.

Extended Trust Chain 과 관련하여 시빌(Sybil) 저항성 평가모델을 해결하기 위한 주요업무가 있습니다. 특히 P. Otte[22]는 NetFlow 알고리즘을 사용하여 시빌 공격과 PageRank 의 수정, 즉 임시 PageRank 를 사용하여 평판 상태를 업데이트합니다. Constellation 은 퍼포먼스 분석을 하는 도중에 먼저 임시 PageRack 를 복제한 다음 가능한 때에 이를 개선합니다. 밈의 증명을 사용하여 대표자

[20] J. Sabater, REGRET: A reputation model for gregarious societies
http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.8.7554rep=rep1type=pdf

[21] Guru: Universal Reputation Module for Distributed Consensus Protocols A. Biryukov etal.
https://eprint.iacr.org/2017/671.pdf

[22] P. Otte, “Sybil-resistant trust mechanisms in distrubuted systems”
http://repository.tudelft.nl/islandora/object/uuid:17adc7bd-5c82-4ad5-b1c8-a8b85b23db1f/datastream/OBJ/

선출에 대한 평판 기반 득점을 개별 계정으로서의 각 노드의 역할과 함께 사용하면 양호한 행동을 유도하는 투명성을 강화함으로써 허가받지 않은 네트워크에서도 허가 된 시스템의 이점을 부여 할 수 있습니다. 모든 노드 (및 경유 마이크로 서비스)에 대한 모든 거래 및 합의 기록은 공개적으로 공증됩니다. 이는 거래 비용과 불평등 한 합의 메커니즘으로 좋은 행동을 강요하는 기존 기술에서 소홀히 하고 있는 일종의 신뢰를 우리에게 부여합니다.

밈의 증명(Proof-Of-Meme) 설명

우리는 분산된 합의 담합 이슈를 해결하는 세 가지를 논의하였습니다. (Hylochain, Fault Tolerance, Proof of Meme). 이들은 다음과 같이 대표자 선출에 대한 모든 구성요소입니다. 우리는 REGRET 에서 사용 된 접근 방식을 따라 밈에 대한 온톨로지를 개발할 것이며, 이것은 밈의 평판점수를 위한 특징 공간이 될 것입니다. 우리는 평판 상태를 업데이트하기 위해 임시 PageRank 를 조정할 것입니다. 우리는 GURU 의 검증 프레임 워크를 대표자 선출 방법에 통합할 것입니다. 자세한 내용은 개발 프로세스 중에 추가될 예정입니다. 평판 온톨로지를 구축하고, 평판 점수에 대한 결정적 모델을 훈련하며, 대표자 선출을 위한 샘플링 알고리즘을 구현하는 것이 우리 개발의 핵심 구성 요소입니다.,

이제 이 점수를 대표자 선출에 사용하는 방법에 대해서 자세히 설명해 보고자 합니다. 각 특징은 네트워크에 대한 노드의 유용성을 나타냅니다. (컨센서스 중 주목할 만한 결함이 있었는지, 컨센서스에 얼마나 많은 메모리를 할당 할 수 있는지 등). 밈은 위에 설명 된 것과 같이 숙련된 모델의 함수를 통해 숫자 값으로 변환할 수 있는데, 이는 밈의 평판을 계량화하는 방법입니다. 새로운 노드가 자신의 밈을 향상 시키도록 촉진하는 메커니즘을 제공하기 위해 밈 점수의 확률 분포가 고정 된 크기의 버킷에 그려지는데, 여기서 분포는 특정 버킷에서 점수를 가진 밈이 선택 될 확률을 설명합니다. 전산적인 관점에서 볼 때 이것은 노드를 서브 루틴으로 다시 클러스터링하는 클러스터링 알고리즘으로 구현 될 것입니다. 이것은 동일한 수준의 내결함성 및 확인 시간을 유지하면서 조력자의 수를 줄임으로써 처리량을 최대화하기위한 것입니다. 평판이 좋은 밈에게 우선권이 주어지지만, 새로운 밈은 네트워크에 참여하고 평판을 쌓을 기회를 갖게됩니다. 성능에 대한 로그는 체인에서 공증 될 것이므로, 결함이 있는 자원을 제공하거나 적대적인 방식으로 수행하는 밈은 평판이 낮아지고 성능이 좋고 신뢰할 수있는 밈은 증가 할 것입니다. 대표자 선출에 대한 우리의 귀납적(경험적)인 아이디어는 다음과 같습니다.

1) 합의가 이루어 집니다.
2) 해시 블록은 각 대표자에 대해 업데이트 된 밈 점수를 출력하는 결정적 알고리즘에 공급됩니다. 이것은 조력자 선정에 대한 책임이 있는 최고 합의 계층 (갤럭시)에서 수행됩니다.
3) 이 상수는 우리의 분포를 섞는 데 사용됩니다 (밈 성능의 새로운 데이터를 기반으로 버킷간에 밈 이동)
4) 이전 블록 해시 (직전 합의 결과)는 각 버킷의 내용을 정렬하는 데 사용되며 (확률 분포에 따라) 각 버킷 중 상위 N 개가 이 라운드에서 선택됩니다. 잘못되었거나 악의적으로 행동하는 (로그 내에서 검증 가능한) 합의 참여자의 밈은 다음 컨센서스 참가자 선거에서는 제재 받게 됩니다.

밈 평판이 더 많은 가치를 갖기 시작하면, 거래 수수료에 대한 필요성이 대체됩니다. 마이크로서비스는 서비스 호스팅을 위해 평판이 좋은 밈 찾을 것이고, 이러한 행위는 합의 수행을 위한 거래 수수료를 얻는 것보다 수익성이 높을 것입니다. 대기시간이 길어질 때 가격을 인상하는 대신 평판점수가 낮은 밈을 줄임으로서 그들의 트랜잭션은 더 낮은 우선순위로 수행이 될 것입니다. 이 경우 밈은 (합의에 참여하여) 리소스를 제공하게 되고, 자체 처리량을 늘리고 네트워크 대역폭 문제를 해결할 것입니다.

밈을 소유하기 (Own Your Meme)

Constellation 은 네트워크 거래 수수료가 없으며 평판은 대기 시간과 적대 공격을 방지하기위한 고유 메커니즘이 될 것입니다. 네트워크에 시드를 하기 위해서는 제네시스 블록 이후에 토큰발행이 중지되는 정해진 날짜까지 수익이 감소하면서 새로 발행되는 토큰의 형태로 제공될 것입니다. 이 새로이 발행되는 토큰은 각 밈의 평판에 따라 배정이 될 것입니다.

판매 시스템, 특히 자동판매기의 시나리오를 고려하시기 바랍니다. Constellation 에서 운영되는 자동판매기는 합의에 참여하기로 선택한 마이크로서비스로 구현될 수 있습니다. 밈의 평판은 시간이 지남에 따라 성장할 것이며, 매우 높은 네트워크 트래픽 상황에서도 매우 짧은 트랜잭션 시간이 보장될 것입니다. 이는 실제 응용 프로그램이 경쟁력있는 서비스를 제공하기 위해 높은 처리량을 필요로 하므로 반드시 필요합니다. 이는 자동판매기의 고객이 낮은 평판을 지니고 있다고 할지라도 자동판매기의 높은 평판은 고객이 높은 트래픽 처리기간에도 편리하게 상품을 수령을 가능할 수

있게 함을 의미합니다.

P2P 아키텍쳐 (Peer to Peer Architecture)

이전 섹션에서는 P2P 계측의 구조에 대해서 간략히 설명하였으며, 이제 보다 자세히 설명해 보고자 합니다.

스타(Stars) - 별

스타는 Constellation 의 기본 오브젝트 입니다. 네트워크에서 직접 상호 작용하기 위해 사용자는 장치에서 스타 노드를 인스턴스화하고 이 스타 인스턴스를 통해 트랜잭션을 발행합니다. 각 스타에는 네트워크상의 기록으로 구성된 로컬 체인이 있습니다. 이 로컬 체인은 주문을 시행하는 데 사용되며 공개 키로 식별됩니다. 개인 키는 트랜잭션에 서명하는데 사용됩니다. 스타 자체는 사용자가 네트워크와 상호작용할 수 있는 경량 클라이언트이며 기본적으로 모바일 장치와 호환됩니다. 그러나 스타가 원한다면 합의에 참여하여 밈의 평판을 높이 평가할 수 있습니다. 스타가 합의에 참여하기로 결정한 순간 스타 집합으로 합쳐지게 되고, 우리는 이를 스타 클러스터라고 부릅니다.

스타 클러스터(Star Clusters) - 성단

스타 클러스터는 합의에 참여하기로 결정한 스타의 집합입니다. 총 스타의 수는 상한에 의하여 제한되며, 위에서 설명한 바와 같이 충분한 실험 및 통계적 분석에 따라 향후 결정될 예정입니다. 이 임계값에 도달하게 되면 새로운 클러스터가 생성이 되고, 각 스타 클러스터는 Locality Sensitive Hash Block 을 형성하며, Locality Sensitive Hash Block 은 다시 일반 트랜잭션으로 취급되어 갤럭시(Galaxies)에 의해 블랙홀(Block Hole)로 해시 됩니다.

갤럭시(Galaxies) - 은하

갤럭시는 Extended Trust Chain 에서 유효성 검사기의 역할과 동형이지만 자동 스케일링 그룹으로도 활동하며 새로운 스타 클러스터에 대한 자원을 제공하고 평판을 유지합니다. 밈 평판은 합의 실적의 로그를 통해 계산됩니다. 컨텐츠의 제안된 블록, 대기시간 및 실패와 같은 메타 데이터가 Locality Sensitive Hash Block 과 함께 갤럭시로 전송됩니다. 갤럭시가 해당 데이터를 받으면, 새로운 샘플이 다음 합의를 위해 선택되기 전에 밈의 평판을 업데이트 합니다. 갤럭시는 유효성 검사자가 되므로 유효하지 않은 트랜잭션을 제거하고 스타 클러스터에서 대표자를 선출하기 위한 정확한 기반(source of truth)을 제공합니다. 갤럭시의 성능에 대한 메타 데이터는 블랙홀에 저장됩니다. 스타의 평판이 임계값에 도달하면 갤럭시 노드로 작동할수 있는 권리를 얻을 수 있습니다.

블랙홀(Black Holes)

블랙홀은 해시된 Locality Sensitive Hash Block 의 블록입니다. 블록체인에서 블록으로 참조하는 것과 동일합니다. 갤럭시는 블록체인의 전체 기록을 저장합니다.

마이크로서비스로서의 스마트 컨트랙트

고 가용성, 탄력성 분산 시스템은 서버리스 아키텍쳐에서 번영할 것입니다. 분산 운영체제의 경우 분산 마이크로 서비스 네트워크로 구현할 수 있습니다. 이에 따라 Constellation 은 스마트 컨트랙트가 JVM 에서 실행되는 마이크로 서비스로서 트랜잭션을 전송하고, 상대방으로서 서명하고, 합의를 수행할 수 있으며, 궁극적으로 마이크로 서비스 자체가 해당 밈과 함께 스타로서 합의된 금액에 따라 운영하는 것을 목표로 하고 있습니다. 이더리움 혹은 카운터파티(Counterparty)에서 스마트 컨트랙트와 동일한 역할을 수행할 수 있지만, JVM 생태계의 기존 코드베이스를 활용하여 보다 복잡한 로직을 제공할 수 있을 것이며, RPC 인터페이스를 통해 외부 프로그램과 통신 할 수도 있을 것입니다. 이러한 마이크로 서비스가 구체적인 서비스 레벨 계약(SLA) 혹은 type signature[23] 수준으로 구축되면 복합 로직을 체인으로 연결하여 분산 응용 프로그램을 보다 직관적으로 구성 할

[23] https://en.wikipedia.org/wiki/Type_signature

수 있습니다. 이것이 MapReduce 운영자가 이루고자 했던 것입니다. 스마트 컨트랙트 마이크로서비스는 (비동기 아키텍쳐의 경우) 계산상의 복잡성을 개선하고 새로운 어플리케이션에 맞게 다시 사용할 수 있는 데이터 모델을 주고 받을 수 있도록 설계할 수 있을 것입니다.

위의 천체의 구성요소로 은유하여 표현한것을 생각해보시면 Constellation 에 있는 분산된 응용프로그램 자체가 별자리(Constellation)임에 유의할 필요가 있습니다. 각 마이크로 서비스가 스타일 때, 연결되거나 그리고/또는 구성된 마이크로서비스 집합은 어플리케이션을 나타내는 선(line)으로으로 연결될 수 있습니다. 이것을 머릿속에 한번 그려본다면, 이것이 별자리(Constellation)을 만든다는 사실을 알아채기는 매우 쉬울 것입니다.

블록체인 운영시스템 Constellation

위의 서버리스 아키텍처는 분산 운영 체제의 사례입니다. 운영 체제의 목표는 기본 하드웨어의 리소스를 활용할 수있는 인터페이스를 제공하는 것입니다. 이는 응용 프로그램 개발에 필수적이며 높은 수준의 프로그래밍 언어와 직관적인 사용자 인터페이스를 제공합니다. 분산 운영 체제는 분산 컴퓨팅 클러스터의 기본 리소스에 대한 인터페이스를 제공한다는 점에서 다릅니다. 블록 체인은 본질적으로 분산 시스템이므로 블록 체인 호환 응용 프로그램을 작성하려면 기본 클러스터의 모든 기능을 활용하려면 분산 운영 체제가 필요합니다. 일반적으로 운영 체제에는 기본 하드웨어 상태를 유지 관리하는 프로그램이 필요합니다. 이것을 커널이라고합니다. 서버리스 아키텍처로 분산 운영 체제를 구축 할 수 있으며, MicroOs 와 같은 우리의 시나리오에서 운영 체제는 마이크로 서비스의 모음입니다. 이것을 응용 프로그램 개발자의 관점에서 보아 마이크로 서비스는 Constellation 으로 구축 된 포괄적 인 응용 프로그램을위한 레고 (Lego)와 같은 빌딩 블록입니다. 이 목표는 비전문가도 빌딩 블록으로 기존 마이크로 서비스를 사용하여 응용 프로그램을 구축 할 수있게하는 것입니다. 직관적 인 사용자 인터페이스와 결합하여 기술적이지 않은 사용자가 Constellation 에서 분산 응용 프로그램을 개발할 수 있도록 마찰없는 온 - 램프를 제공합니다.

위에서 설명한 것처럼 개인이 아이디어를 응용 프로그램으로 자연스럽게 (비 기술적으로) 구체화 할 수있는 능력은 Noosphere 에서 물질 세계로의 잠재적 에너지의 변형이라고 말할 수 있습니다. 이 때문에, 우리는 우리의 목표와 아이디어를 구성하는 비 공간 잠재 에너지에 비추어 우리의 통화를 noOs 로 명명하기로 결정했습니다.

결론

우리는 주류 애플리케이션이 블록 체인 기술을 사용할 수있게 해주는 최신 서버리스 아키텍처에 대한 암호학적으로 안전한 합의를 재구성했습니다. 우리의 접근법은 Locality Sensitive Hash 의 차원 축소 기술을 인터넷 규모에서 신뢰 체인을 허용하는 분할 및 정복 메커니즘을 적용했습니다. 우리의 밈 경제는 암호 보안의 더 깊은 계층과 트랜잭션 비용을 필요 없도록 구성할 수 있는 가능성을 제공합니다. 또한 분산 응용 프로그램과 자연에서 관찰 할 수있는 자체적인 유사 크기 조정 패턴 간의 직관적인 상관 관계를 그리는 분산 아키텍처를 천체에 대입하여 설계한 콘스텔레이션(별자리)를 여러분께 선보입니다.