Anoma: 풀스택 탈중앙화 애플리케이션을 위한 통합 아키텍처

Christopher Goes, Awa Sun Yin, Adrian Brink

사전 공개, 2022년 8월 21일

요약

프로그래밍 가능한 정산 아키텍처는 상대방 발견과 문제 해결 기능을 제공하지 않는다. 이 두 기능은 대부분의 상호작용하는 다자간 애플리케이션을 구축하는데 필수적이다. 프로그래밍 가능한 정산의 아키텍처적 제약으로 인해 현재의 애플리케이션 프로토콜들은 최소 하나의 Web2 구성 요소를 가지고 있으며, 이것이 중앙화 지점이 된다. 우리는 풀 스택 탈중앙화 애플리케이션을 위한 통합 아키텍처인 Anoma를 제시한다. Anoma 는 의도(intent) 중심성과 동질적 아키텍처/이질적 보안 이라는 원칙을 따라 설계되었으며, 이 둘은 탈중앙화 애플리케이션 구축을 위한 선언적 패러다임을 구성한다. 이 논문에서는 먼저 Anoma 아키텍처를 개괄하고, 설계 논리에 대한 직관을 제공하며, Anoma 가 어떻게 프로토콜과 보안의 선택을 분리하는지 설명한다. 그다음 Anoma 애플리케이션 프로그래밍 모델을 정의하고, 신기술을 사용하여 구축할 수 있는 여러 기존 및 신규 탈중앙화 애플리케이션을 나열한다. 마지막으로 Anoma를 구현하는데 사용되는 현재 구성 요소를 개괄하고 향후 연구 방향을 제시한다.

1. 배경 및 동기

2008년 비트코인 프로토콜의 출시는 스크립트 기반 정산의 시작을 알렸습니다. 이는 개별적인 속성과 통화 정책을 가진 암호화폐에 적합한 분산 원장 기술(DLT) 아키텍처의 한 유형입니다. 튜링 완전성(Turing-complete)을 갖추지는 못했지만, 비트코인 스크립트[1]는 네임코인(Namecoin)이나 컬러드 코인(Colored Coins)과 같은 화폐 외 다른 애플리케이션도 지원할 수 있습니다. 하지만 이더리움 백서[2]에서 논의된 바와 같이, 스크립트 기반 정산 위에 구축된 애플리케이션은 기능적이지만, 이 아키텍처는 많은 제약과 한계점을 드러냈습니다.

2014년 이더리움 프로토콜의 도입은 프로그래밍 가능한 정산의 시대를 열었습니다. 이는 튜링 완전성 가상 머신 실행을 활용해 탈중앙화 애플리케이션을 구축하는 새로운 아키텍처 유형을 제시하며, 정산 레이어에 훨씬 더 풍부한 표현력을 부여했습니다. 프로그래밍 가능한 정산은 스크립트 기반 정산으로는 지원할 수 없었던 애플리케이션의 더 발전된 형태를 위한 길을 열었습니다. 예를 들어, 대체 가능한 토큰(Fungible token) (ERC20)이나 이더리움 네임 서비스(ENS)는 오늘날 각각 컬러드 코인 및 네임코인 아이디어의 더 발전된 형태이며, 대체 불가능한 토큰(NFT), 탈중앙화 자율 조직(DAO) 또는 최근에 도입된 소울바운드 토큰(SBT)[3]과 같은 다양한 애플리케이션의 개발을 가능하게 했습니다.

이더리움 출시 이후 제안되고 배포된 블록체인 프로토콜들은 특정 아키텍처 구성 요소에 대한 상당한 개선을 가져왔습니다. 그 예로는 합의 메커니즘 (Tendermint [4], Avalanche [5]), 시빌 공격(Sybil attack) 저항 메커니즘(지분 증명, 저장 증명), 확장 솔루션 (샤딩, 롤업), 암호화 방식(영지식 증명) 등이 있습니다. 하지만 이러한 구성 요소들의 개선은 프로그래밍 가능한 정산의 기본 아키텍처를 바꾸지는 못합니다.

프로그래밍 가능한 정산은 특정 애플리케이션에는 충분하지만, 많은 최신 애플리케이션은 추가적인 요구 사항을 가집니다. 즉, 당사자들이 이미 무엇을 누구에게 정산할지 결정했을 때에만 정산만으로 충분하다는 의미입니다.

이러한 문제를 해결하기 위해, 기존 애플리케이션들은 보통 스택의 일부인 중앙화된 구성 요소(prover, solver, sequencer 등)에 의존하는 아키텍처를 채택했습니다. 이 구성 요소들은 주로 웹2(Web2) 서비스 형태로 구현됩니다. 예를 들어, 대체 가능한 자산을 위한 탈중앙화 거래소(0x, CoWSwap, Uniswap), 대체 불가능한 자산을 위한 거래소(Wyvern, LooksRare, OpenSea), 새로운 투표/펀딩 메커니즘(이차 투표/자금 조달, Gitcoin) 및 롤업(Optimism, Arbitrum, Starknet, zkSync) 등이 있습니다. 이들의 아키텍처는 최소 하나의 중앙화된 구성 요소를 포함하며, 이로 인해 종종 무허가성, 장애 내성, 또는 검열 저항성과 같은 분산형 시스템의 핵심 가치가 훼손됩니다.

아키텍처에서 중앙화 지점을 피하고자 하는 애플리케이션이 취하는 새로운 접근법 중 하나는 스택의 특정 구성 요소를 대체하기 위해 애플리케이션별 자체 주권 체인(sovereign chain)을 배포하는 것입니다. 이 접근법이 즉각적인 중앙화 문제를 해결할 수는 있지만, 네트워크 효과(애플리케이션 간 조합성 및 소프트웨어 재사용)의 손실과 더불어, 개발자와 사용자에게 불균형적인 복잡성을 가중시키는 단점이 따릅니다. 이는 사용자가 다양한 보안 레이어와 지연 시간 도메인을 고려해야 하기 때문입니다.

이 논문에서 우리는 Anoma를 소개합니다. Anoma는 풀스택 탈중앙화 애플리케이션을 위한 통합 , 의도(intent) 중심 접근 방식, 탈중앙화된 상대방 찾기, 그리고 유효한 상태 전환을 계산하는 해결자(solver)에게 NP 탐색 문제의 연산 아웃소싱을 위임하는 것을 특징으로 합니다. 이 아키텍처를 통해 최신 애플리케이션을 무허가성, 장애 내성, 또는 검열 저항성을 타협하지 않고 구축할 수 있습니다.
아키텍처로
Anoma의 아키텍처는 또한 멀티체인 원자성 정산과 같은 새로운 기능을 제공합니다. 이를 통해 다른 보안 선호도를 가진 사용자와 애플리케이션들의 원자성을 얻을 수 있습니다. 이러한 신기술과 여러 기능들은 기존 아키텍처로는 구축할 수 없는 애플리케이션 개발의 길을 열어줍니다. 이 중 일부는 섹션 5 '애플리케이션'에서 자세히 다룹니다.

2. Anoma 배포 로드맵

2.1. Galileo: Anoma 분산 OS(운영 체제)를 이더리움(기타 EVM 체인포함) 및 추가 시스템(SVM, CosmWASM 등)에 배포 Galileo는 개발자가 각 지원 플랫폼/아키텍처에서 OS의 모든 기능을 사용할 수 있도록한다. 구체적으로 개발자는 다음과 같은 기능을 수행할 수 있다.

- 트랜잭션이 아닌 의도 측면에서 애플리케이션을 작성
- 애플리케이션을 한 번만 작성하여 어디든

배포 가능

2.2. Dagon: 다중 도메인 동시성 및 정산 구현

Dagon은 OS의 모든 개별 배포를 모두 하나의 완전한 분산 시스템으로 네트워킹 한다.
구체적으로는 다음을 의미한다:

- 사용자는 특정 도메인의 의도를 다른 모든 도메인에서 사용 가능(예: 크로스체인 거래).
- 개발자는 다양한 사용 사례에 대해 서로 다른 기본 시스템을 활용 가능(예: Celestia의 DA, 이더리움의 순서 지정).

2.3. Ahra: Anoma 월드 컴퓨터의 하이퍼 스케일링을 위한 확장성 제한 없는 로컬 및 글로벌 합의

Ahra는 월드 컴퓨터를 완성한다. 네트워크에 연결된 모든 장치가 분산 OS의 일부가 될 수 있도록 한다.

- 기업용 체인이나 로컬 거래와 같은 로컬 합의 인스턴스의 배포를 가능하게 한다.
- 이 지점에서 Anoma는 월드 컴퓨터의 꿈을 완전히 구현한다.
- 솔라나보다 빠르고(로컬 합의는 항상 글로벌 합의보다 빠름) 비트코인보다 더 탈중앙화 된다(합의는 지구상의 모든 인간이 참여할 수 있음).

3. 토큰 유틸리티

Anoma의 네이티브 토큰인 XAN은 Anoma라는 분산 OS를 구동하는 글로벌 조정 메커니즘 이다.
사용자부터 해결자(solver), 합의 제공자에 이르기까지 관련된 모든 행위자 간의 조정을 촉진하며, 간단한 의도부터 복잡한 ZKP 증명에 이르기까지 모든 것에 사용된다. 전통적인 베이스 레이어가 하나의 배포에서 생성된 경제적 가치의 일부만을 포착하는 것과는 달리, 분산 OS(Anoma)는 어디에서나 조정되는 총 가치의 일부를 포착한다. 따라서 기존 비교 대상들보다 우수한 가치 포착 역학을 가진다. 근본적으로 이 유틸리티는 이더리움, 솔라나, 플라즈마, 주권(sovereign) 기업 체인 또는 3개의 휴대폰 간의 로컬 합의 등 어디에서 일어나든 글로벌 경제적 조정을 구동하는 것에서 나온다.

- 수동 또는 프로그래밍 방식의 사용자들은 실행 비용을 지불하기 위해 의도와 함께수수료를 지불한다.
- 마켓 메이커와 유사한 전통적인 매칭뿐만 아니라 ZKP 증명 또는 데이터 저장과 같은 다른 역할을 수행할 수 있는 해결자는 해당 서비스에 대해 수수료를 부과한다.
- 순서 지정을 제공하는 합의 제공자는 추가로 수수료를 부과한다.
- DA와 같은 다른 유형의 노드도 개별 수익 모델을 구성할 수 있다.

특정 Anoma 사례(인스턴스)들은 수수료를 전략적 준비금에 할당할지, 서비스 제공자에게 분배할지, 아니면 소각할지 선택할 수 있다. 경제적으로는 후자가 가장 유리할 가능성이 높으므로 대부분의 배포가 이 경로를 선택할 것으로 예상된다.

경제 흐름에 관련된 모든 행위자를 조정하는 것 외에도 XAN은 분산 OS, 많은 배포 및 이를 구동하는 핵심 소프트웨어의 지속적인 진화를 촉진한다.
따라서 토큰은 다음과 같이 사용될 수 있다:

- 특정 새 소프트웨어 릴리스로 업그레이드하려는 의사를 알림
- 업그레이드된 배포로 가치 이동
- 자체 거버넌스 시스템을 구축하고 배포할 필요 없이 로컬 또는 글로벌 인스턴스를 업그레이드하는 작업 수행

요약하자면, 단일 베이스 레이어에서 가스 지불에만 네이티브 자산을 사용하는 것과 달리, Anoma에서는 분산 OS의 모든 행위자가 토큰을 사용하며, 수수료

포착은 특정 배포에 제한되지 않고 Anoma가 촉진하는 모든 경제적 흐름에 비례할 수 있다. 토큰이 분산 OS(Anoma) 어디에서든 조정되는 총 가치의 일부를 포착할 수 있게 하므로, 기존 비교 대상들보다 우수한 가치 포착 역학을 가진다. 근본적으로 이 유틸리티는 이더리움, 주권 기업 체인, 또는 3개의 휴대폰 간의 로컬 합의 등 어디에서 일어나든 글로벌 경제적 조정을 구동하는 것에서 나온다.

4. 토크노믹스 & 베스팅

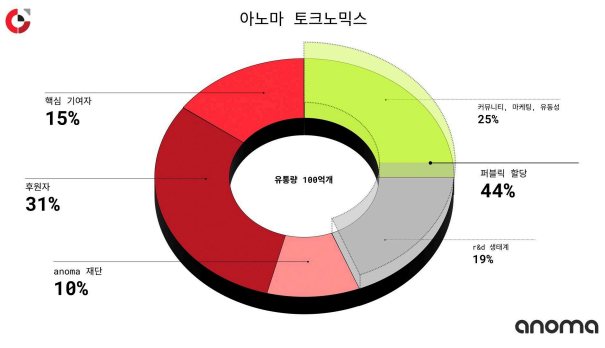
티커: XAN

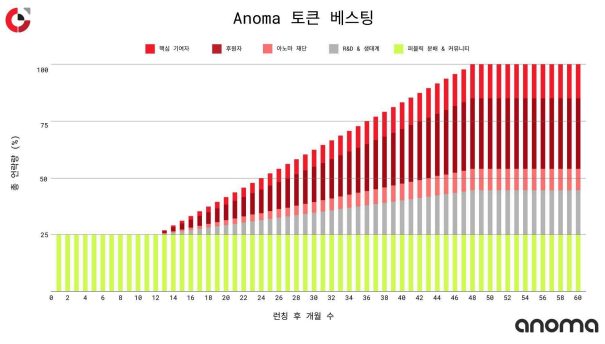
최대 공급량 : 10,000,000,000 XAN

초기 유통 공급량 : 2,5000,000,000 XAN

토큰 형식 : 네이티브 ERC20

락업 : XAN 공급량의 75%는 초기에 락업되어있으며, 아래의 토큰 언락 일정에 따라 3년에 걸쳐 해제됩니다.





5. 아키텍처 설계 철학

Anoma의 아키텍처는 두 가지 설계 원칙에 의해 구동된다. 첫째는 의도 중심성(intent- centricity)이고, 둘째는 이질적 보안 모델을 가진 동질적 프로토콜 아키텍처이다. 이러한 두 가지 설계 원칙 외에 다른 모든 아키텍처 선택들은 모듈화와 런타임 구성 매개변수의 문제다.

5.1. 의도(Intent) 중심성

의도는 사용자가 프로토콜과 상호 작용할 때 달성하고자 하는 것의 표현이다. 예를 들어 "X를 A에서 B로 전송"하거나 "X를 Y로 거래"하는 등이 있다. 실제로 의도는 사용자가 달성하고자 하는 상태 전환을 인코딩하는 오프체인 서명 메시지다.
트랜잭션과 달리 의도는 부분적이므로, 의도를 모든 사용자의 제약을 만족하는 최종 균형 트랜잭션을 형성하기 위해 다른 직접적 또는 간접적 부분을 보완으로 필요로 하는 트랜잭션의 일부로 생각할 수 있다.

기존 프로토콜은 트랜잭션을 가장 기본적인 단위로 설계되었다. Anoma는 근본적으로 다른 접근 방식을 취한다. Anoma의 아키텍처는 프로그래밍 가능한 의도를 중심으로 한다.

의도 중심 아키텍처는 상대방 발견을 가능하게 하는데 필요하며, 이는 매력적인 애플리케이션에 중요하다. 왜냐하면 이들은 다자간 조정을 필요로 하고 풀스택 탈중앙화 애플리케이션을 가능하게 하기 때문이다.

Anoma는 거래 상대방 발견, 해결 및 정산을 수직적으로 통합하며, 의도를 기본적이고 일반적으로 해석하고 처리할 수 있다. 앞서 설명한 바와 같이 현대 애플리케이션은 거래 상대방 발견, 해결 및 정산 모두를 필요로 한다. 의도는 사용자가 이러한 애플리케이션과 상호작용하는 지점이며, 의도 중심 설계는 이러한 두 프로세스가 함께

작동하고 검열 저항성 및 장애 내성 속성을 만족해야 하는 애플리케이션의 요구사항을 포착한다.

의도 중심 설계는 또한 애플리케이션 구축을 위한 선언적 패러다임을 구성한다. Anoma는 사용자가 정의한 대로 의도를 정산하도록 설계되었기 때문이다. 즉, 의도는 정의된 대로 정산되거나, 아예 정산되지 않는다. 이 선언적 모델은 사용자에게 기본 프로 토콜 기능과 실행 흐름을 이해할 필요 없이 훨씬 더 높은 수준의 제어를 제공하며, 이는 탈중앙화 애플리케이션이 대중적으로 채택되기 위해 매우 중요한 역할을 한다. 이 패러다임은 애플리케이션에 대한 명령형 모델을 기본으로 하는 기존 트랜잭션 중심 아키텍처와 비교하여 근본적으로 다른 접근 방식을 제시한다. 후자에서는 사용자가 보안과 데이터 지역성 보장의 혜택을 받기 위해 전체 실행 추적을 이해해야 하는데, 특정 상태 변경을 승인하는 대신 특정 실행 경로를 실행하기 때문이다.
실사례에서도 이 과정은 너무 어려워서 사용자들이 위험을 이해하지 못한 채 애플리케이션과 상호작용하는 경우가 흔하게 발생한다.

애플리케이션 개발자에게 Anoma의 의도 중심 아키텍처는 의도와 거래 유효성 판단 조건(validity predicate)의 조합 을 활용하여 구조적으로 더 안전한 애플리케이션을 구축할 수 있게끔 한다. 거래 유효성 판단 조건은 상태 전환 계산 작업과 상태 전환의 정확성 검증 작업을 깔끔하게 분리하는 스마트 컨트랙트 아키텍처다. 이는 계산과 검증을 얽히게 하는 메시지 전달 VM 실행 모델(현재 프로그래밍 가능한 정산 아키텍처에서 널리 사용됨)과 비교된다. 거래 유효성 판단 조건은 애플리케이션 개발자들이 다른 애플리케이션들이 어떻게 상호작용하는지 걱정하지 않고 자신의 애플리케이션이 만족했으면 하는 불변량에 대해 추론할 수 있게 해준다. 그들의 애플리케이션의 거래 유효성 판단 조건이 이러한 불변량을 직접 표현하기 때문이다.

5.2. 동질적 아키텍처, 이질적 보안

Anoma 프로토콜은 TCP/IP 프로토콜 스택과 마찬가지로 동질적 아키텍처 및 이질적 보안 원칙을 따른다. TCP/IP에서 인터넷 프로토콜의 다양한 계층은 표준화되어 있지만,

누구와 연결하고 누구에게 어떤 데이터를 맡길지 선택하는 것은 사용자에게 달려 있으며, 다른 사용자는 동일한 프로토콜 스택을 사용하면서도 다른 선택을 할 수 있다. Anoma에서는 거래 상대방 발견, 해결 및 정산의 다양한 계층도 유사하게 표준화되어 있지만, 어떤 보안 도메인을 신뢰할지와 누구에게 어떤 데이터를 보낼지의 선택은 것은 사용자에게 달려 있으며, 다른 사용자는 동일한 프로토콜 스택을 사용하면서도 다른 선택을 할 수 있다.

이 프레임워크에서 프로토콜은 아키텍처와 보안이라는 두 가지 차원으로 분석될 수 있다.

- 아키텍처:

시스템의 구조를 구성하는 추상화 및 관계다. 아키텍처는 구문적이며, 속성과 구문을 가지고 있지만 외부 세계와 관련하여 특징적인 의미를 가지지는 않는다. 단일 아키텍처로의 수렴은 사용자를 특정선택으로 제약하지 않으면서 시간과 검증 비용을 절약한다.

- 보안:

분산 시스템 운영에서 누구를 어떻게 신뢰할지 선택한다. 보안은 특정 사용 맥락의 특정 의미론과 분리할 수 없는 결정(decision)이다. 보안은 시스템 운영자에게 제공되는 정보와 그에 따른 선택 능력을 제한함으로써 어느 정도 경제적으로 추상화될 수 있지만, 운영자는 항상 다음과 같은 선택을 갖게

된다. 누구로부터 어떻게 메시지를 받아들일지, 언제 블록이나 다른 집계에 포함시켜 투표할지, 예외적 상황에 대응해 언제 투표를 중단하거나 정상 운영 절차를 변경할지 등이 있다. 이러한 책임을 누구에게 맡길지는 데이터베이스의 상태가 현실 세계에서 무엇을 나타내는지에 달려 있으며, 데이터베이스 사용자들의 이익과의 일치는 순전히 경제적인 것을 넘어선 상호 이익을 필요로 한다.

5.2.1. 플랫폼 분석

이 두 차원을 따라 분산 렛저 플랫폼들을 고려해보자. 프로토콜 아키텍처와 보안 모델, 그리고 동일한 플랫폼에서 실행되는 다른 애플리케이션들에 대해 이들이 동질적인지 이질적인지 알아본다. 프로토콜 아키텍처는 상태 레이아웃, 가상 머신, 언어 지원, 샤딩 메커니즘, 크로스 컨트렉트 메시징 모델 등을 의미한다. 아키텍처는 플랫폼용 애플리케이션을 작성하는 데 무엇이 필요한지를 결정하며, 애플리케이션은 특정 아키텍처에 특화된다.

- 분산동질적 아키텍처를 가진 플랫폼들은 모든 애플리케이션이 특정 형식(예: EVM 바이트코드(bytecode) 또는 WASM)으로 작성되어야 한다.
- 이질적 아키텍처를 가진 플랫폼들은 애플리케이션이 다른 형식으로 작성될 수 있도록 허용하며, 크로스 체인 프로토콜과 같은 경계에서의 일부 합의가 있을 수 있다.

보안 모델은 이론적 보안(합의의 장애 내성 속성, 포크 감지 및 처리)과 실제 보안 (이러한 아키텍처의 배포된 인스턴스를 운영하는 마이너나 검증자)을 모두 의미한다.

- 동질적 보안 모델을 가진 플랫폼들은 모든 애플리케이션에 대해 동일한 보안을 갖는다.
- 이질적 보안 모델을 가진 플랫폼들은 다른 애플리케이션들에 대해 다른보안 특성을 갖는다.

설명을 위해 표 1은 이 두 축에서 여러 플랫폼들을 위치시킨다. 표가 시사하는 바와 같이, 이 차원들은 일반적으로 상당한 상관관계가 있다. 동질적 아키텍처는 동질적 보안 모델과 함께오고, 이질적 아키텍처는 이질적 보안 모델과 함께 온다.

플랫폼	아키텍처	보안 모델
비트코인	동질적	동질적
이더리움	동질적	동질적
이더리움 2.0	동질적	동질적
폴카닷	이질적	동질적
니어	동질적	동질적
코스모스	이질적	이질적
멀티체인	이질적	이질적
Anoma	동질적	이질적

[표 1: 아키텍처와 보안 모델에 기반한 플랫폼 분석]

이들이 상관관계에 있는 시스템을 설계하는 것이 더 쉽다. 모든 것이 동질적이면 프로토콜들을 깔끔하게 맞출 수 있고, 크로스 컨트랙트 통신을 포함한 기능들이 더 쉬워진다. 반면 모든 것이 질적이면 프로토콜들은 블록체인 간 통신 프로토콜(IBC) [6]과 같은 상호작용의 경계에서만 합의하고, 보안의 복잡성을 처리하는 것은 사용자와 애플리케이션 개발자에게 달려 있다.

5.2.2. 이러한 차원을 분리하는 이유

Anoma의 프랙탈 인스턴스 아키텍처는 이러한 차원을 분리하고 아키텍처적으로 동질적이면서 이질적 보안 모델을 가진 플랫폼을 구축하도록 설계되었다. 이는 더 복잡하지만, 그렇지 않은 최고의 보안 모델이 무엇인지의 문제를 거의 확실하게 분리한다 (”자비로운 독점” a` la Git 또는 TCP/IP 가 있을 수 있는 곳).

프랙탈 인스턴스용으로 작성된 애플리케이션은 Anoma가 제공하는 아키텍처를 표준 화할 수 있으며, 이는 복잡한 상호운영성, 자동 확장 등을 허용할 만큼 충분히 잘 정의되어 있으면서도 단일 보안 모델에 합의할 필요가 없다. 더 나아가 일부 경우에는 이러한 선택의 유연성이 애플리케이션 사용자에게까지 확장될 수 있으며,이들은 독립적으로 선택할 수 있다.

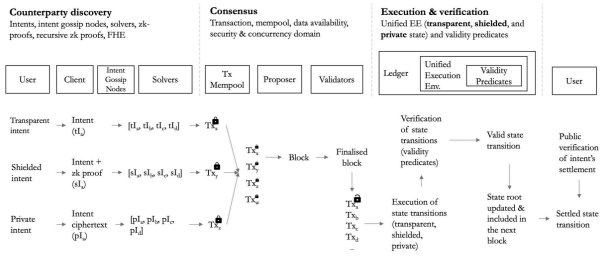
Anoma 인스턴스의 사용자 인터페이스는 다른 보안 모델로 배포된 동일한 애플리케이션을 지원할 수 있으며, 후자의 차이를 사용자들이 동일한 프로토콜 사용의 네트워크 효과를 유지하면서 신뢰 가정을 선택할 수 있는 방식으로 전달할 수 있다.

주목할 점은 Anoma의 아키텍처가 여러 배포 모델을 지원하므로 몸을 구속하는 구속복 같은 동질성이 아니라는 것이다. 프로토콜의 구성 요소는 계층화되어 있어서 프랙탈 인스턴스들이 참여할 부분을 선택할 수 있다. 심지어 탈중앙화된 상대방 발견과 해결 같은 특정 기능에 대해서는 Anoma를 활용하면서 최종 정산을 이더리움과 같은 다른 플랫폼에 고정하는 것도 포함한다. 그럼에도 불구하고 통합되고 수직적으로 통합된 아키텍처는 개발자와 사용자가 표준화의 혜택을 받을 수 있게 한다.

6. 아키텍처 토폴로지

Anoma의 아키텍처 토폴로지는 데이터 흐름에서의 역할에 따라 구분되는 논리적 추상화 집합으로 구성되며, 특정 표현 형태, 배포 모델, 암호화 구현 선택 등과는 독립적으로 구성된다.

그림 1 은 아키텍처 토폴로지의 개요를 제공한다. 특정 인스턴스화는 서로 다른 구체적인 성능과 보안 의미를 가지며, 해당 배포의 특정 요구사항에 따라 선택되어야 한다.
섹션7:애플리케이션에서 다양한 배포에 대한 Anoma가 선택한 스케치를 제공한다.



[그림1: Anoma 아키텍처에서 의도의 생명주기]

6.1. 노드 및 네트워크 계층

Anoma의 아키텍처 토폴로지는 우리가 노드라고 부르는 네트워크로 연결된 튜링 머신의 기판 위에서 작동한다. 노드들은 의도 소문 전파(intent gossiping), 솔루션 검색, 합의 투표와 같은 다양한 운영 역할을 수행할 수 있다. 다른 역할들은 다른 하드웨어 요구 사항을 가질 것이지만, 노드는 단일 클래스이며 런타임 구성 설정에 따라 노드가 수행하는 역할이 결정된다.

모든 노드는 결정론적으로 계산하며, 로컬 무작위성을 생성할 수 있는 능력을 가지고(예: 암호화에서 비밀 값으로 사용될 수 있음) 로컬 스토리지에 대한 읽기 및 쓰기 액세스 권한을 가진다. 노드 집합은 무제한적이고 동적이며, 노드들은 언제든지 들어오고 나갈 수 있다. 노드는 개방형 네트워크에서 부분적으로 연결되어 있으며, 다른 역할에는 다른 연결을 필요로 한다. 네트워크 계층은 신뢰할 수 없고(메시지가 임의로 삭제, 복제 또는 재정렬될 수 있음) 신뢰할 수 없는 가정을 갖는다(암호화되지 않은 데이터는 비밀이 아님). 특정 역할은 부분적 동기화와 같은 더 엄격한 네트워크 가정을 가질 수 있다.

6.2. 의도

의도(intent)는 부분적인 상태 전환을 설명하는 서명된 메시지이다. 의미적으로 의도는 상태 선호에 대한 정보를 포함한다. 예를 들어 Alice가 X를 Y로 교환하고 싶어한다거나, 속성 T를 가진 임의의 X를 속성 U를 가진 임의의 Y로 교환하고 싶어한다거나, Z를 어떤 자산 A로 바꾸고 싶지만 A가 이전에 Bob에 의해 소유되었고 Bob이 추가 서명을 제공하는 경우에만 그렇게 하고 싶어한다는 등의 예시가 있다. 더 일반적으로 의도는 정산 계층에 의해 런타임에 평가되는 임의의 코드다.

의도는 부분적이므로 특정 거래 상대방이 필요하지 않지만, 완전할 수도 있다(완전 한 상태 전환은 부분적인 상태 전환의 하위 집합임). 예를 들어, 의도는 Alice가 자산 A를 Bob에게 보내고 싶다는 것을 표현할 수 있으며, 이는 실행되기 위해 Alice 외에는 아무도 동의할 필요가 없는 상태변경이다. 이러한 의도도 특정 정보가 Alice에게 알려지지 않은 경우 해결사(solver)를 필요로 할 수 있다. 예를 들어, Alice는 베를린의 현재 온도에 비례하여 바운티 값을 설정하고 싶다고 표현할 수 있는데, 이 값은 그녀가 모르지만 오라클 키를 알고 있는 값이며, 오라클 데이터 접근을 가진 해결사가 제공할 수 있다. 상대방도 해결사도 필요하지 않은 의도는 즉시 트랜잭션으로 변환될 수 있다. 자산, 속성등의 표현의 특정 구문은 애플리케이션 수준에서 고정된다. 아키텍쳐 수준에서 의도는 불투명한 바이트 문자열(Bytestring)이 된다.

6.3. 의도 소문 전파(intent gossip) 계층

의도 소문 전파 계층은 의도 배포, 거래 상대방 발견 및 해결(해결사가 유효한 트랜잭션을 만들 기 위해 여러 의도를 결합할 때)을 위한 가상 오버레이(virtual space overlay) 네트워크다. 의도 소문 전파 계층은 희소한 단위로 네트워크로 연결된 의도 소문 전파 노드로 구성되며, 의도 소문 전파는 모든 노드가 수행할 수 있는 역할이다. 클라이언트가 해결이 필요한 의도를 작성하면, 서명된 의도 를 의도 소문

전파 노드에 브로드캐스트하고, 이 노드는 의도 소문 전파 계층을 통해 의도를 추가로 릴레이한다. 이 브로드캐스트는 노드가 해결 전문화나 다른 기준에 따라 특정 다른 노드들을 선택하는 방향성이 있을 수도 있고, 노드가 의도를 가능한 한 널리 브로드캐스트하는 무방향성일 수도 있다.

의도는 의도가 충족되고, 정산되고, 합의에 의해 확인된 경우에만 지불되는 정산 조건부 수수료를 포함할 수 있다. 더 나아가 이 수수료는 가십에 관련된 모든 노드와 최종 해결사(solver) 간에 분할될 수 있다. 의도는 해결사가 각 의도 배치에 대한 최상의 일치를 찾기 위해 경쟁하는 데이터 가용성 도메인에서 (암호화될 가능성이 높은) 의도의 확인 및 순서 지정을 위해 수수료를 지불할 수 있다.

6.4. 해결사(Solver)

해결사는 의도 또는 의도의 하위 집합을 관찰하기로 선택하고 집합에 대한 솔루션을 계산하는 노드다. 하나 또는 여러 해결사 알고리즘을 실행함으로써 이를 달성한다. 이러한 알고리즘들은 로컬이며 다른 해결사는 제시된 제약 시스템을 충족시키기 위해 서로 경쟁한다. 실제로 해결사는 대체 가능한 토큰 거래 또는 롤업 상태 계산과 같은 특정 애플리케이션에 전문화될 가능성이 높다. 해결사는 무허가(permissionless)이며 누구나 해결사 역할을 할 수 있다. 해결사는 어떤 의도를 수락할지 결정할 수 있으며 일반적으로 수수료나 거래에서 예상되는 스프레드로 인해 저장 및 대역폭 비용을 감당할 가치가 있는 것들만 고려해야 한다. 해결사 알고리즘은 정산 계층의 현재 상태와 알려진 의도 풀을 기반으로 가능한 솔루션의 공간을 검색하여, 정산 계층에서 받아들여지는 트랜잭션을 생성하기 위해 결합 가능한 의도의 하위 집합을 찾는 것을 목표로 한다.

6.5. 트랜잭션

트랜잭션은 현재 상태에서 새로운 상태로의 함수로 작동하는 완전한 상태 전환이다. 트랜잭션은 선언적 프로그래밍(declarative program) 모델을 따르며. 이를 계산하기 위한 명령형 단계가 아닌 원하는 최종 상태를 설명한다. 결과적으로 해결사나 일반 사용자와 같은 트랜잭션 제출자는 자신의 트랜잭션의 동작에 대해 추론할 때 실행 단계를 고려할 필요가 없다. 이더리움이나 다른 프로그래밍 가능한 정산 아키텍처와 같은 기존 시스템에서 제출자는 프록시 계약을 포함하여 모든 중간 실행 단계를 인식하고 신뢰해야 하는데, 이들이 명령형 계산을 수정하고 최종 상태 결과를 변경할 수 있기 때문이다 Anoma의 선언적 접근 방식을 사용 하면 제출자는 중간에 수행되는 계산을 걱정하지 않고최종 상태만 정확하게 명시하면 된다.

제출자는 Ferveo 분산 키 생성(DKG) 공개 키 [7]에 대해 트랜잭션을 암호화한다. 노 드는 암호화된 트랜잭션만 수신하고 가십(gossip)한다. 합의가 암호화된 바이트 문자열의 순서를 정한 후, 합의 노드의 2/3 과반수가 원래 트랜잭션을 해독하고 공개한다. Ferveo는 비상호작용적이므로, 원래 트랜잭션의 공개를 강제하기 위해 추가적인 경제적 보안 보장이 필요하지 않다.

6.6. 멤풀(Mempool)

멤풀은 트랜잭션을 위한 가상 밀집 분할 오버레이 네트워크다. 멤풀은 보안 및 동시성 도메인(프랙탈 인스턴스)을 기반으로 분할되며, 멤풀에 참여하는 노드는 관심 있는 프랙탈 인스턴스의 트랜잭션만 가십한다. 의도 소문 전파 네트워크와 대조적으로, 멤풀은 특정 프랙탈 인스턴스의 검증자가 해당 인스턴스로 향하는 모든 트랜잭션을 수신해야 한다는 점 에서 밀집되어 있다고 볼 수있다. 멤풀은 투명한 트랜잭션이 아닌 암호화된 바이트 문자열만

수신, 저장 및 가십하기 때문에 불투명하다.

6.7. 데이터 가용성 도메인

데이터 가용성 도메인은 논리적 시계 및 데이터 가용성 계층이다. 이러한 데이터 가용성 도메인은 모든 애플리케이션에 의해 프로그래밍 가능하다. 애플리케이션이 특정 시간 간격이 지난 후 동시에 한번에 모두 해독되는 의도 배치를 지정할 수 있게 한다. 의도는 특정 배치의 노드에 암호화된 형식(Ferveo 사용)으로 제출될 수 있다. 배치가 완료되면 검증자는 배치의 모든 의도를 해독하고 해독된 내용을 상태에 추가한다. 이러한 의도는 상태 머신에 의해 직접 실행되지 않고, 애플리케이션에 의해 정의된 측정 가능한 기준에 따라 최상의 솔루션을 제공하기 위해 경쟁하는 해결사에게 제공된다.

6.8. 보안 도메인

보안 도메인은 합의에서 특정 상태 전환 함수를 실행하는 암호학적으로 식별된 노드들의 집합으로, 일반적으로 $n \geq 3f + 1$ 이라는 프로토콜에 따라 행동하는 특정 비율의 노드들의 특정 가정 하에서 최종성과 정확성이 유지된다. 다양한 시빌(Sybil) 저항 메커니즘이 노드 집합을 선택하는데 사용될 수 있다. 그 예시로는 지분 증명(PoS), 작업 증명(PoW), 신원 증명(PoI) 또는 권위 증명(PoA) 등이 있다.

6.9. 동시성 도메인

동시성 도메인은 도메인 내의 트랜잭션 집합에 대한 전체 순서 지정으로, 다른 동시성 도메인에 대해 부분적으로 순서가 지정되거나 순서가 지정되지 않을 수 있다. 동시성 도메인은 항상 특정 보안 도메인 내에서 작동한다. 왜냐하면 전체 순서는 보안 도메인의 합의에 의해 강제되기 때문이다.

6.10. 합의

합의는 보안 도메인을 형성하고 시간을 양자화하는 다수 당사자(일부는 비잔틴일 수 있음) 간의 합의를 위한 알고리즘이다. 합의 알고리즘은 트랜잭션을 블록으로 그룹화하는 책임을 진다.

6.11. 실행

실행 환경은 현재 상태와 트랜잭션 집합을 가져와서 해당 트랜잭션을 상태에 적용하여 새 로운 상태를 생성하는 알고리즘이다. Anoma는 다양한 상태 전환을 처리할 수 있는 통합 실행 환경을 제공한다.

- 투명 데이터는 실행 노드와 관찰자에게 공개된다.
- 준(準)비산 데이터(Semi-discreet data)는 실행 노드 및 관찰자에게 알려져 있지 않지만 단일 사용자에게는 알려지며, 사용자는 ZKP를 사용하여 해당 속성을 증명할 수 있다.
- 비산 데이터 (Discreet data)는 누구에게도 독립적으로 알려지지 않으며, 다양한 형태의 동형 암호화(HE)를 사용해 암호화된 형태로 계산되고 저장된다.

Anoma는 수행되는 검증 종류와 무관하게 데이터의 지역성에 대해 추론할 수 있는 일반 적인 프레임워크를 제공하지만, 기본 암호화 체계의 성능 특성이 다양한 애플리케이션의 실제 실행 가능성 및 실행 비용을 결정한다. 영지식 또는 낙관적(optimistic) 롤업과 같은 기술은 지원되는 모든 지원되는 상태 전환과 함께 사용될 수 있다.

6.12. 애플리케이션

애플리케이션은 많은 사용자가 상호 작용할 수 있는 특정 상태 파티션의 형식과 논리를 관리하는 의미론적 도메인다. 애플리케이션은 다음으로 구성됩니다:

● 상태: 여러 프랙탈 인스턴스 및 해당 인스턴스 내의 샤드에 걸쳐 분할될 수 있다.

● 애플리케이션 거래 유효성 판단 조건: 애플리케이션 상태의 변경을 관리한다.

● 사용자 거래 유효성 판단 조건 구성 요소: 사용자가 애플리케이션과의 특정 상호 작용을 승인하기 위해 포함할 수 있다.

● 의도 형식: 클라이언트가 의도를 생성하고, 해결사가 추론하며, 애플리케이션 유효성 술어가 처리할 수 있도록 한다.

● 해결사 알고리즘: 해결사가 특정 애플리케이션 또는 여러 다른 애플리케이션의 의도를 만족시키는 트랜잭션을 만들 수 있도록 한다.

● 인터페이스: 사용자에게 애플리케이션과 상호 작용하기 위한 시각적, 공간적 및 시간적 추상화를 제공한다.

6.13. 프랙탈 인스턴스(Fractal instance)

프랙탈 인스턴스는 네트워크로 연결된 검증자 집합에 의해 운영되는 Anoma 합의 및 실행 프로토콜의 인스턴스다. 일반적으로 프랙탈 인스턴스는 보안 도메인 (특정 검증자 집합에 의해 운영되며, 사용자는 그 정족수(quorum)을 신뢰해야 함), 동시에 도메인(실행하는 트랜잭션들만의 전체 순서를 유지), 데이터 가용성 도메인(외부 관찰자가 프랙탈 인스턴트에 쿼리해 상태의 일부를 검색할 수 있음) 이다.

프랙탈 인스턴스는 지속적으로 올바르게 실행되기 위해 프랙탈 인스턴트 그래프의 다른 부분에 의존하지 않는다는 점에서 주권적이다. 다만 검증자 집합이 겹칠 수 있으며, 이는 멀티체인 원자적 정산을 제공하는 특정 경우에 활용될 수 있는 속성이다. 프랙탈 인스턴스들은 네트워크의 모든 기능과 호환되기 위해 사양에 따라 Anoma 합의 및 정산 프로토콜을 구현해야 하지만, 선택한 시빌 저항 메커니즘, 실행 가격 책정, 프로토콜 버전 관리의 로컬 거버넌스, 경제적 분배 체제 및 비정규 상태 전환 처리에서 다를 수 있다.

[그림 2: 최종 사용자 입장에서의 아노마 사용처]

7. 프로그래밍 모델

시스템 상태에 대한 선호도를 가진 사용자의 관점에서 Anoma의 아키텍처를 고려할 때, ”애플리케이션이 존재하는 이유가 뭔가?”라는 질문을 던질 수 있다. 사용자가 단순히 자신의 선호도를 명확히 표현하고 시스템이 추가적인 구성요소의 중재 없이 이를 실행할 수는 없는가? 원칙적으로는 가능하지만, 해결사의 검색 공간과 원장 상태와 세계 상태 간의 관계를 조정 하는 어려움은 특정 표현 형태와 선호도 표현 및 정산의 특정 논리에 대한 조정 없이는 계산적으로 다루기 어렵다. 애플리케이션은 의도를 표현, 매칭 및 정산하고 위해, 그리고 사용자에게 간단하고 정확한 인터페이스를 제공하기 위해 조정이 필요한 이러한 특정 형태들을 설명한다.

7.1. 애플리케이션 구성 요소

Anoma 아키텍처의 애플리케이션은 의도 형식, 애플리케이션 상태 거래 유효성 판단 조건(validity predicate), 사용자 거래 유효성 판단 조건 구성 요소, 해결사 알고리즘, 그리고 하나 또는 여러 사용자 인터페이스로 구성된다.

● 의도 형식은 애플리케이션에서 활용하는

특정 의도의 형식과 의미론을 설명한다. 이는 사용자 인터페이스에 의해 생성되고, 의도 소문 전파(intent gossip) 노드에 의해 이해되며, 해결사에 의해 매칭되고, 애플리케이션의 거래 유효성 판단 조건에 의해 검증되어야 한다.

- 애플리케이션 상태 거래 유효성 판단 조건은 애플리케이션 상태의 유효한 상태 전환 관리하는 관계를 인코딩한다.
- 사용자 거래 유효성 판단 조건 구성 요소는 사용자가 이 애플리케이션과의 안전한 상호 작용을 허용하기 위해 승인할 수 있는 관계를 인코딩한다.
- 해결사 알고리즘은 이 애플리케이션의 의도를 매칭하고 유효한 트랜잭션을 형성하는 방법을 해결사에게 지시한다.
- 마지막으로 사용자 인터페이스는 해당 애플리케이션에 대한 그래픽 또는 텍스트 뷰와 컨트롤러를 사용자에게 제공한다

7.2. 애플리케이션 이식성(Application portability)

기본적으로 애플리케이션은 프랙탈 인스턴스 간에 이식 가능하며, 애플리케이션 상태 유 효성 술어는 이러한 도메인 안에 활동하는 애플리케이션 사용자들 간의 안전한 상호 작용을 허용하기 위해 보안 및 동시성 도메인에 대해 추론할 수도 있다. 특정 인터페이스를 특정 애플리케이션에 연결하는 매개체는 없지만, Anoma의 의도 소문 전파 네트워크는 안전하게 동기화된 인터페이스 및 애플리케이션 버전을 허용하는 방식으로 인터페이스 코드를 위한 데이터 가용성 계층 역할을 할 수 있다.

7.3. 애플리케이션 보안 모델

Anoma에서 사용자는 애플리케이션을 신뢰하지 않는다. 애플리케이션은 사용자 상태를 수정할 수 있는 무제한 액세스 권한을 부여받지 않는다. 모든 상태 항목은 명시적인 소유 자를 가지며, 해당 소유자와 관련된 거래 유효성 판단 조건은 해당 상태에 대한 모든 변경을 승인해야 한다. ‘transferFrom‘과 같은 방식으로 승인하는 대신, 사용자는 특정 애플리케이션과의 특정 상호 작용을 허용하는 구성 요소를 거래 유효성 판단 조건에 추가하며, 애플리케이션이 그렇게 할 수 있는 라이선스를 부여받은 경우 사용자의 관점에서 비상호작용적으로 수행될 수 있다. 이러한 구성 요소는 언제든지 변경되거나 취소될 수 있으며, ”심층 방어(defence-in-depth)”를 허용한다 (예: 시간 제한 t 내에 X 이상의 전송 방지).

7.4. 애플리케이션 상태 모델

Anoma는 클라이언트가 상태를 가진다라고 가정한다. 즉,이들은 분산 시스템의 구성 요소로 취급된다. 메시지는 한 번만 전송되며, 전달된 것으로 표시될 수 있고, 이 경우 보관되지 않는다. 메시지 기록은 과거 트랜잭션 아카이브를 재처리하여 재구성할 수 있다.

8. 애플리케이션

Anoma의 아키텍처는 특정 의미적 도메인에 대한 특정 형태의 선호도 형식에 대해 거래 상대방 발견, 해결 및 정산을 제공하려는 모든 애플리케이션에 적합하다. 여기서는 Anoma가 애플리케이션 개발자에게 제공하는 몇 가지 기능들을 나열한다. 그런 다음 현대 탈중앙화 애플리케이션의 몇 가지 예시와 이들이 Anoma의 아키텍처가 어떻게 혜택을 받을 것인지를 나열한다. 이어서 기존 아키텍처의 제약으로 인해 지금까지 개발이 실용적이지 않거나 개발이 불가능했던 새로운 탈중앙화 애플리케이션에 대한 설명이 이어진다.

8.1. 애플리케이션을 위한 새로운 기능(Primitive)

Anoma는 애플리케이션 개발자에게 몇 가지 새로운 기능을 제공한다:

● 인센티브화된 데이터 가용성: 미래 트렌젝션 생성에 사용될 것으로 예상되는 데이터를 위한 것으로, 가십 계층에 의해 제공됨 (섹션 8: 아키텍처 인스턴스화 참조)

● 프로그래밍 가능한 해결사: 해결사 알고리즘을 실행하는 의도 소문 전파 노드들에 의해 제공되며, 모든 참가자의 선호도를 동시에 만족하는 다자간 원자적 상태 전환(트래잭션)을 찾는 계산 작업을 아웃소싱할 수 있음

● 프로그래밍 가능한 임계값 암호 해독: Ferveo [7]에 의해 제공되며, 온디맨드 배칭을 구현하고 논리적 시간의 양자화된 기간 내에 제출된 애플리케이션별 상태 전환 처리에 대한 구성 가능한 공정성 속성을 강제하는데 사용될 수 있다. ZKP 시스템과의 완전 동형 암호화(FHE): 데이터 자체에 대한 지식으로부터 데이터의 속성 검증을 분리하는데 사용될 수 있음. 애플리케이션 개발자들은 이를 활용해 동일한 애플리케이션에서 다양한 상태모드를 처리하는 애플리케이션을 구축할 수 있음.

이러한 기능들이 결합되어 탈중앙화된 이차 투표(quadratic voting)과 이차 펀딩(quadratic funding), 인센티브화 된 데이터 가용성을 통한 투표, 해결사를 통한 정산, ZKP 및 FHE를 통한 영수증이 필요 없는 등의 기능과 같이 원하는 게임 이론적, 데이터 지역성 및 지연 시간 속성을 제공하는 복잡하고 사용자 친화적인 애플리케이션을 구축하는 데 필요한 유연성을 제공한다.

8.2. 애플리케이션 예시

8.2.1. 현대 탈중앙화 애플리케이션

여기서는 우리는 의도, 거래 상대방 발견, 해결 설계 패턴을 따르지만 현재는 애플리케이션별로 존재하며 최소 하나의 단일 운영자 구성 요소에 의존하는 현대 애플리케이션 모음의 예시들을 나열한다.

8.2.1.1. 탈중앙화 거래소 Ox, CoWSwap, Uniswap, Wyvern 및 Seaport와 같은 대체 가능(fungible)과 대체 불가능 토큰(non-fungible)을 위한 현대 탈중앙화 거래소는 배치/공정 실행과 같은 다른 요구 사항 외에도 거래 상대방 발견, 해결 및 정산 모두를 필요로 한다. 현재 이러한 프로젝트들은 거래 상대방 발견을 위해 블록체인 자체를 사용하거나(Uniswap) 특정 당사자에 의해 통제되는 단일 운영자 오더북을 운영하며(0x, Wyvern, Seaport, CoWSwap), 이들은 공정한 순서 지정과 최적 실행에 대해 신뢰받는 경향이 있다. Anoma를 사용하면 이러한 당사자들을 임의의 거래를 통해 일반화하는 피어 투 피어 의도 소문 전파(P2P intent gossip)과 분산 해결 계층으로 대체할 수 있다. 특정 자산을 사고 파는 주문들은 의도로서 의도 소문 전파 네트워크를 통해 브로드캐스트 되고, 해결사에 의해 매칭되며, 해결사는 거래의 균형을 맞추기 위해 원하는 수만큼의 의도를 수집하고, 해당 자산을 보유한 프랙탈 인스턴스에 정산을 위해 제출할 수 있다. 임계값 암호 해독은 배치간 공정성을 위해 사용될 수 있다.

8.2.1.2. Arbitrum, Optimism과 같은 낙관적(optimistic) 롤업과 ZkSync 또는 StarkNet과 같은 영지식 롤업 모두를 포함한 기존 롤업 아키텍처들은 트랜잭션 순서 지정, 상태 업데이트 계산 및 업데이트 된 상태를 루트 체인(이 경우 이더리움)에 제출하는 역할을 하는 단일 운영자 시퀀서(sequencer) 와 해결사로 운영된다. 이 시퀀서는 공정한 순서 지정 및 최적 해결을 위해 신뢰되며, 선택적으로 트랜잭션을 생략할 수 있으므로 일부 프로젝트는 시퀀서를 탈중앙화하고 싶다고 표현한다. 탈중앙화된 시퀀서는 단순히 합의

인스턴스이므로 이러한 롤업은 Typhon 합의를 사용하여 Anoma 프랙탈 인스턴스를 인스턴스화하여 시퀀서를 운영하고, 현재 하는 것과 같이 이더리움 실행에 있어 영지식 또는 낙관적 증명을 제출할 수 있다.

8.2.1.3. Gitcoin에서 구현된 이차 펀당(Quadratic funding)(QF)은 거래 상대방 발견, 해결(자금 제공자의 지급액이 개별 기부에 따라 달라지므로) 및 정산 모두를 필요로 한다. Anoma 를 사용하면 QF는 뛰어난 UX를 제공하는 방식으로 구현될 수 있다(예: 프로젝트에 기부하는 데 수수료가 없음). 자금 제공자, 프로젝트 생성자 및 모든 개별 기부자는 각각 자금 투입, 프로젝트 실행 및 기부 의사를 반영하는 의도를 작성한다. 해결사 알고리즘은 이러한 의도를 매칭하고 QF 라운드 종료 시 정산을 위한 단일 트랜잭션을 생성하며, 자금 제공자는 정산 수수료를 지불할 수 있다. 기부 금액은 QF 계산을 수행하기 위해 공개 되어야 하지만, 개별 신원은 Anoma의 실행 환경을 사용하여 안전하게 유지될 수 있다. 표현력이 풍부한 의도는 더 단순한 QF 모델에서 표현하기 어려운 추가적인 차원도 포착할수 있다. 예를 들어, 많은 프로젝트는 어떤 일을 하기 위해 특정 금액의 펀딩을 필요로 하며, 특정 임계값을 충족되어야만 펀딩을 받고 (행동에 대한) 약속을 하고자 한다. 이것은 의도에 제약 조건으로 표현될 수 있으며, 해결사는 임계값을 충족하기에 충분한 자금을 찾거나 원하는 대로 프로젝트를 생략하여 최종 정산 트랜잭션이 유효하도록 해야 한다.

8.2.2. 새로운 애플리케이션

여기서는 Anoma의 아키텍처를 사용하여 구축할 수 있는 새로운 탈중앙화 애플리케이션 (DAO 2.0, 런타임 롤업, 다자간 다변량 물물교환, 경매 및 로컬 에피소드 게임)을 간략하게 설명한다.

8.2.2.1. 탈중앙화 자율 조직(DAO)은 조직 운영의 투명성(의사결정 규칙이 누구나 읽을 수 있는 동일한 코드로 표현되고 실행됨)과 운영 검증 가능성(조직의 과거 행동이 이 규칙 집합과 일치한다는 것을 제3자에게 증명할 수 있음)이라는 이중 약속을 지닌다. 하지만 현재의 구현에서는 공개 블록체인에서 실행함으로써 투명성과 검증 가능성을 얻는다.

운영 공개를 통해 조직은 의사 결정의 모든 측면을 공개하지 않고도 조직의 입력 및 출력에 대한 특정 데이터(예: 비영리 단체의 분기별 펀딩 공개)를 제시하고 검증 가 능성으로 증명할 수 있다. 의사결정의 모든 측면을 공개하는 것은 실제로 일어나느 일을 잘못 표현하기 위해 누군가가 마음대로 선별할 수 있는(cherry-pick) 할 수 있는 많은 데이터이거나, 다른 의제를 가진 대중 구성원들(경쟁 조직을 운영하거나 해당 조직의 구성원과 개인적인 원한을 가진)이 사소한 논쟁을 시작하거나 조직 운영을 방해하는데 사용할 수 있다.

Anoma의 아키텍처는 그러한 타협을 할 필요가 없는 복잡한 DAO 생성을 허용한다. 즉, 의사결정 규칙과 데이터 지역성을 모두 유지하여 조직 내 당사자에게만 보이게 하면서도, 자신이 선택하는 대로 각각의 임의 속성을 세상에 증명할 수 있다.

특히 이 시스템은 복수 화폐 시스템[8]과 같은 것을 인스턴스화하는 데 사용될 수 있다. 커뮤니티는 커뮤니티 구성원들에 의해 통제되고, 내부 커뮤니티

통화, 커뮤니티 소유 SALSA 할당 자산, 커뮤니티 외부로의 자산 이전에 대한 제한/세금을 가진 복잡한 DAO를 스스로 만들 수 있다.

8.2.2.2. 런타임 롤업

"롤업"을 계산과 검증의 분리로 정의해보자. 검증은 향상된 장애 내성을 위해 적절하게 복제될 수 있지만 계산은 그럴 필요가 없다. 명령형 의미 체계에 의존하고 최종 사용자가 특정 명령형 실행 경로에 서명하는 시스템에서는 롤업이 장기간 지속되며 사용자에 의해 구체적으로 지정되어야 한다. Anoma의 선언적 아키텍처에서 사용자는 특정 실행 경로가 아닌 실행이 충족해야 하는 속성을 표현하는 의도에 서명하므로, 롤업은 동적 수요에 따라 런타임에 생성될 수 있으며, 더 저렴한 경우 복제 대신 계산을 위한 시장이 사용될 수 있다.

8.2.2.3. 다자간, 다변수 복잡한 물물교환

세 친구 Alice, Bob, Charlie, 호텔 운영자 David, 페스티벌 프로듀서 Eve, 그리고 기차회서 Deutsche Bahn이 있다고 해보자. 페스티벌은 7월 포츠담 근처에서 3주 동안 열린다.Alice, Bob, Charlie는 같은 주말에 함께 페스티벌에 참석하고 싶어하며, 각자의 고향인 베를린, 취리히, 암스테르담에서 기차를 타고 가려고 한다. 그들은 특정 주말에 더 유연한 일정을 가지고 있고, 기차표, 호텔 방, 페스티벌 패스의 총 가격이 가능한 한 낮기를 원한다. Eve는 비용과 마크업을 기반으로 한 고정 가격의 페스티벌 티켓을 판매하고 싶어하지만, 때로는 일찍 구매한 후 참석할 수 없다는 것을 깨달은 당사자들에 의해 재판매된다. Deutsche Bahn은 수요에 따른 가변 가격으로 기차표를 판매하고, David는 호텔 방에 대해서도 마찬가지로 판매한다(1인용 싱글룸과 4인용 쿼드룸을 모두 보유). Alice, Bob, Charlie는 페스티벌에 참석하는 다른 사람과 룸메이트가 되는 것을 마다 않는데, 이를 우정이 좋을 가능성의 증거로 본다.

오늘날의 세상에서 Alice, Bob, Charlie는 페스티벌 웹사이트에서 티켓이 있는지를 확인한 다음, 가능한 3번의 주말에 걸쳐 호텔 및 기차 가격을 확인하고 비용이 얼마나 될지 파악하기 위해 스프레드시트를 작성하려고 할 것이다. 물론, 그들이 스프레드시트를 작성하는 동안, 여행을 계획하는 다른 사람이 그들의 호텔 방이나 기차 좌석을 예약할 수 있고, 그렇게 되면 운이 없는 경우가 될 것이다. 설상가상으로, 특정 주말에 호텔 방을 예약한 후, 기차표를 구할 수 없다는 것을 알게 되고 호텔 방을 변경할 수 없게 될 수도 있다 (적어도 취소 수수료를 지불하지 않고는 말이다).

Alice, Bob, Charlie, David, Eve, Deutche Bahn은 모두 Anoma를 다자간 복합 물물교환의 기반으로 사용할 수 있다. 각 당사자는 자신의 선호도를 담은 의도를 작성하고, 모든 의도는 원자적으로 일치하거나(Alice, Bob, Charlie 모두의 기차표, 호텔 방, 페스티벌 패스가 한번에 예약됨을 의미) 전혀 일치하지 않게 된다. 복잡한 물물교환을 사용하면 모든 당사자가 원하는 것은 공개되지만, 누구인지는 공개될 필요가 없다.

이는 대체 가능한 토큰과 같은 더 간단한 경우에도 사용될 수 있다. 사용자는 시장 및 지정가 주문의 의미를 포착하는 의도를 작성할 수 있으며, AMM과 같은 더 복잡 한 알고리즘도 작성할 수 있다. 의도 형태로 표현된 AMM 주문은 단순히 두 자산을 교환하려는 가격 곡선이다 (xy = k). 사용자는 전체 가격 범위 또는 하위 범위에 대해 AMM 의도를 작성할 수 있다 (Uniswap v3와 유사). 온체인 AMM과 달리, Anoma는 트랜잭션을 보내거나 자산을 잠글 필요가 없다.

8.2.2.4. 복잡한 경매

장기적인 이유와 별개로, 경매는 종종 게임 이론적인 이유로 데이터 지역성의 혜택을 받는다. 봉인 입찰 2차 가격 경매는 입찰자에게 자신의 진정한 가치를 입찰할 이유를 제공하지만, 작동하려면 입찰 공개가 필요하다. 첫 번째이자 가장 즉시 실현 가능한 방법은 프로그래밍 가능한 임계값 암호 해독을 사용하여 경매 마감일이 지날 때까지 모든 입찰을 암호화된 상태로 유지한 다음, 한 번에 모두 해독하고, 최고 입찰자를 낙찰자로 선택하고 2차 최고 가격을 청구하는 것이다. 이는 신원 은폐를 위한 다른 공개 기술과 결합될 수 있다. FHE는 암호문으로 제출된 입찰에 대한 연산을 직접 수행함으로써 복잡한 경매를 구현하는 데도 사용될 수 있다.

8.2.2.5. 로컬 에피소드 게임

디지털에서 포커 게임을 플레이 한다고 생각해 보자. 포커 게임은 에피소드적으로 발생한다. (베팅이 이루어지고 승자가 리더보드에 보고되더라도) 다른 게임들 간에는 상호작용이나 순서 지정이 일어나지 않는다. 사용자가 행동을 제출하는 경우, 같은 게임 내 사용자가 취한 행동은 그 게임의 다른 사용자가 취한 행동과 관련해서는 순서가 정해져야 하지만, 다른 사용자들이 취한 행동과는 그렇지 않다. Anoma의 프랙탈 인스턴스 아키텍처는 이 구조를 효율적으로 인스턴스화할 수 있다. 플레이어들이 포커 게임을 시작할 때, 해당 게임 내에서 상태 전환의 순서를 지정하기 위해 (단순히 그들 사이에서 운영되는) 임시 합의 인스턴스를 시작한 다음, 마지막에 더 장기간으로 실행되는 프랙탈 인스턴스의 포커 추적/통계 애플리케이션에 결과를 제출한다. 이 프랙탈 인스턴스는 낮은 지연 시간을 위해 LAN에서 실행될 수 있으며, 트랜잭션은 비용을 필요로 하지 않는다 (제출할 수 있는 집합이 플레이어로 제한되기 때문).

포커는 또한 개별 핸드를 비밀로 유지하고 주기적으로 카드를 공개하는 데이터 공개와 (덱 셔플을 위한) 무작위성을 필요로 하며, 이는 각각 Anoma의 실행 시스템과 임계값 암호 시스템의 임계값 서명에 의해 제공될 수 있다

9. 아키텍처 인스턴스화

Anoma 아키텍처는 다양한 성능, 복잡성 및 인체공학적 절충안을 가지고 다양한 방식으로 인스턴스화될 수 있는 개별적으로 복잡한 많은 하위 구성 요소를 많이 필요로 한다. 여기서 우리는 필요한 하위 구성 요소에 요구되는 추상적 인터페이스를 간략하게 설명하고, 이를 인스턴스화 하는 현재 개발 방향을 요약한다.

9.1. 가십(gossip)(소문 전파)

Anoma 가십 시스템은 가명으로 식별되고, 경로 인증되며, 장애 책임을 지는 희소 오버레이 네트워크(Sparse Overlay Network)다. 기존의 피어 투 피어(P2) 가십 네트워크와 달리 이 시스템은 선택적 증명으로 작동하도록 설계되었다. 노드는 암호화 키로 식별되며 모든 메시지는 수신자에게 암호화되고 발신자가 서명한다. 노드는 메시지 유효성, 재브로드캐스트 및 보존에 대한 로컬 규칙을 만들고 시행한다. 정산 렛저 및 경로 인증 기반 수수료와 결합하여, 이는 트랜잭션 관련 데이터를 위한 인센티브화된 데이터 가용성 계층을 제공하며, 이는 Anoma 아키텍처 내에서 사용자가 의도를 브로드캐스트하는 데 사용된다. 이 의도는 해결사가 거래 상대방을 찾고, 트랜잭션을 생성하고, 정산을 위해 프랙탈 인스턴스에 제출할 때까지 전달된다. 노드는 메시지 내용 유효성과 재브로드캐스트 기준에 관한 로컬 신뢰 그래프와 규칙 집합을 유지한다. Anoma 가십 시스템은 명시적 신뢰 모델을 사용하며, 여기서 기본 물리적 네트워크는 신뢰하지 않는다. 또한 새로운 노드는 신뢰할 수 있는 피어

공개 키 집합으로 부트스트랩하며, 노 드는 시간이 지남에 따라 신뢰 관계를 유지하여 누가 누구에게 소개했는지 추적하고 신뢰 그래프를 따라 재귀적으로 신뢰의 변화를 적용한다.

9.1.1. 노드 모델

가십 네트워크의 노드는 개인 키를 소유한 것으로 가정하며, 해당 공개 키는 식별에 사용 된다. 노드는 고유한 모든 전송 메시지에 대한 순서를 완전히 지정하고 서명해야 한다. 같은 논스(nonce)로 두 메시지에서 서명하는 것을 보내는 모든 메시지를 완전히 정렬하고 서명해야 하며, 이는 고유합니다. 동일한 논스로 두 메시지에 서명하는 것은 장애로 간주된다. 기존 P2P 가십 시스템에서 노드는 주로 IP 주소로 식별되며, 이는 물리적 네트워크 대상을 참조하고 장기간 지속된다고 가정된다. 반면 Anoma 가십 시스템에서 노드는 주로 공개 키로 식별되며, 이는 잠재적으로 도달할 수 있는 IP 주소를 나열하고 주기적으로 교체할 수 있다(그러나 발신자가 메시지를 보내기 위해 반드시 알 필요는 없음). 이는 일종의 가상 가십 네트워크로 볼 수 있으며, 물리적 기반을 자유롭게 이동할 수 있는 비밀 정보(개인 키)를 기반으로 한 신원 지속적을 가진다. 로공간적으로 비(非)로컬적인 가상화된 네트워크를 공간적으로 로컬인 물리적 네트워크로 이동하여 상대적으로 효율적인 매핑을 유지하기 위해 물리적 라우팅 지연 시간의 로컬 캐시가 유지된다.

이러한 구조 선택은 또한 장애 내성 하위 시스템의 개념적으로 명쾌한(elegant) 가상화를 허용한다. 임계값 암호 시스템과 합의의 조합(순서 지정을 제공하기 위해)은 효과적으로 많은 노드를 하나의 노드로 가상화하며, 임계값 해독과 임계값 서명을 위한 공유는 재구성을 위해 내부적으로 재브도르캐스트 된다(개별 노드 공개 키로 암호화됨). 다른 블록체인 시스템과 달리 Anoma의 가십 네트워크는 보안 도메인을 기반으로 분할되지 않고(독립적인 블록체인 멤풀과 비교) 단순히 희소하며, 실시간 수요가 연결 선택과 라우팅 테이블에 정보를 제공할 수 있다.

9.1.2. 경로 인증

Anoma의 가십 시스템은 경로 인증을 제공한다. 메시지 수신자는 원래 발신자까지 재귀 적으로 서명 체인을 검증할 수 있어서, 메시지 체인의 각 당사자가 다음 전송을 승인했음 을 검증할 수 있고, 가십에 참여한 대가로 잠재적으로 지불을 받고 메시지 간의 일관성 없는 순서 지정에 대한 책임을 모두 질 수 있다. 이는 단순히 메시지 헤더에 순서가 지 정된 서명 목록을 유지함으로써 달성되며, 수신자는 정확성과 연결 일관성을 위해 모두 확인할 수 있다. 효율성을 위해 유효성 검사는 압축될 수 있고 내부 경로 신원은 ZKP를 사용하여 숨길 수 있으며, 수신자는 메시지를 받는 일부로 이를 검증한다.

9.1.3. 가십 인센티브

위에서 설명한 경로 인증 시스템은 사용자가 의도를 브로드캐스트하고 최종적으로 정산 으로 이어지는 가십 체인에 참여하는 모든 노드에 지불을 제안하는 가십 인센티브의 한 형태를 제공하는 데 사용될 수 있다. 사용자는 단순히 가십 노드들이 변조할 수 있또록 허용하는 작은 수수료(런타임에 의미(semantics)가 선택됨)를 포함하여, 각 노드가 메시지를 전달할 때 수수료의 일부를 자신을 위해 가져갈 수 있다. 노드가 다른 의도와 결합하여 유효한 트랜잭션을 만들고, 적절한 프랙탈 인스턴스에 제출하여 의도를 정산할 수 있으면 즉시 수수료를 청구할 수 있다. 그렇지 않은 경우 노드는 메시지를 전달하기 전에 자신의 몫으로 얼마의 수수료를 가져갈지 선택할 수 있다. 물론 모든 수수료를 가져갈

수도 있지만, 그렇게 되면 다른 노드가 의도를 재브로드캐스트하거나 정산할 이유가 없으므로 노드는 아무것도 받지 못하게 될 것이다. 따라서 노드는 잠재적 정산의 기대 이익이 스스로 정산할 수 있는 잠재적 기회 비용을 능가할 만큼만 수수료를 가져가면서 의도를 재브로드캐스트할 것으로 예상된다(하지만 사용자는 많은 당사자에게 의도를 브로드캐스트할 수 있으므로, 즉시 정산할 수 없는 개별 노드가 첫 번째로 정산할 수 있을 가능성은 낮다).

9.2. 합의

합의 구성 요소는 많은 노드를 하나의 가상 노드로 추상화할 수 있는 알고리즘으로, 구성 노드의 정확성에 대한 특정 가정(일반적으로 >2/3)하에서 정확한 모습을 보인다. 개별 노드가 결정론적(deterministic) 상태 머신을 운영하고 로컬 전체 순서로 메시지를 보내고 받는 것처럼, 합의 알고리즘을 사용하여 생성된 가상 노드는 결정론적(복제된) 상태 머신을 운영하고 전체 순서로 메시지를 보내고 받는다. 합의 알고리즘은 가십, 순서 지정 및 트랜잭션(들어오는 메시지) 실행을 통해 많은 노드를 이 가상 노드로 추상화한 다음, 업데이트된 상태(나가는 메시지)를 검증 가능한 방식으로 최종화하는 역할을 한다. 현재 Anoma의 합의 구성 요소는 Heterogeneous Paxos [10], Narwhal [11], Tendermint [4]에서 상당 부분 차용한 Typhon[9]에 의해 인스턴스화된다.

9.2.1. 순서 지정

합의의 순서 지정 구성 요소는 실행 전에 트랜잭션의 순서를 지정하는 책임을 지며, 합의에 참여하는 노드는 순서 지정에 동의하고 그렇게 순서가 지정된 모든 트랜잭션이 실행을 위해 사용 가능하다는 것을 보장해야 한다.

9.2.2. 실행

합의의 실행 구성 요소는 이미 순서가 합의된 트랜잭션을 실행하고, 트랜잭션 실행 결과 를 반영여 상태를 업데이트하며, 외부 당사자가 비용 효율적으로 그 속성을 검증할 수 있도록 업데이트 된 상태를 최종화하는 책임을 진다.

9.3. 실행 환경

Anoma의 실행 환경은 서로 신뢰하지 않는 프로그램의 안전한 상호 운용을 허용하기 위해 상태와 코드를 분할하고 권한을 부여한다. 또한 상태 변경을 추상화하고 암호화 작업을 위한 적절한 기능을 제공하며, 프랙탈 인스턴스 간 상태 검증과 동기 및 비동기 프랙탈 인스턴스 간 메시징을 처리하는 런타임이다. 이러한 추상화의 세 가지 책임은 세 가지 구성 요소로 직교화(orthogonalized) 된다.세 가지는 다음과 같다. 거래 유효성 판단 조건 하위 시스템, 통합 실행 환경(Taiga), 투명 실행 환경(Typhon EE).

9.3.1. 거래 유효성 판단 조건(Validity predicate) 하위 시스템

거래 유효성 판단 조건(VP) 하위 시스템은 서로 신뢰하지 않는 프로그램의 안전한 상호 운용을 허용하기 위해 상태와 코드를 분할하고 권한을 부여하는 책임을 진다. 이는 상태의 키 공간을 상호 배타적인 접두사 공간으로 분할함으로써 달성되며, 키의 첫 부분은 해당 접두사 내의 센티넬 키에 저장된 특정 거래 유효성 판단 조건에 의한 소유권에 해당한다. 특정 접두사 내의 상태가 변경될 때마다,

해당 접두사와 관련된 거래 유효성 판단 조건이 호출되며, 트랜잭션을 수락하거나 거부할 수 있다. 거래 유효성 판단 조건은 다른 거래 유효성 판단 조건도 수락하도록 요구할 수도 있다.

거래 유효성 판단 조건 하위 시스템은 그 자체가 거래 유효성 판단 조건으로 구현되며 원칙적으로 재귀적으로 인스턴스화 될 수 있다. 하위 시스템은 또한 VP가 액세스할 수 있는 데이터에 대한 제한을 시행하는 책임을 진다.

9.3.2. Taiga 통합 EE

Taiga 통합 실행 환경은 데이터 액세스 및 작업을 처리하는 책임을 진다.

9.3.2.1. 데이터 지역성 도메인

투명 데이터는 키를 읽고, 쓰고, 삭제할 수 있고 접두사를 반복할 수 있는 변경 가능한 키-값(Key-value) 트리로 표현된다.

준비산(Semi-discreet) 데이터는 각각 노트가 한번 또는 여러 번 소비될 수 있는 변경 불가능한 추가 전용(append-only) 노트 집합으로 표현된다. 각 노트는 키, 값, 소유자 키를 포함하며, 노트 내용의 암호화가 이용 가능해야 한다.

비산(Discreet) 데이터는 키를 읽고, 쓰고, 삭제할 수 있고 암호문이 특별한 동형 명령을 사용해 연살될 수 있는 키-> 암호문(key->ciphertext) 매핑으로 표현된다

9.3.3. Typhon 투명 EE

Typhon 실행 환경은 트랜적선 순서 지정과 동시 실행을 위해 Typhon이 요구하는 최소한의 요구사항과 구조만을 부과하도록 설계된 가장 낮은 수준의 실행 환경이다.

Typhon 실행 환경은 키-값 트리로 구성된 투명 상태만을 가진다. 트랜잭션은 읽고 쓸 키 공간의 모든 하위 트리의 부모(parents tree)를 선언한다. 이 정보를 사용하여 Typhon은 겹치지 않는 상태 영역만 건드리는 트랜잭션을 식별하고 동시 실행을 위해 트랜잭션의 순서를 정할 수 있다. 이 실행 환경 자체는 그 자체로는 데이터 또는 상태/코드 분할에 대한 상태 의미를 갖지 않는다. 추가 구조는 모든 트랜잭션의 일부로 호출되는 특정 센티넬 키에 저장된 루트 거래 유효성 판단 조건에 의해 지정된다.

Typhon EE는 또한 프랙탈 인스턴스 간의 비동기 메시지 전달과 키메라 체인 내의 동기(원자적) 메시지 전달을 처리하는 책임을 진다. EE는 전송, 순서 지정 및 검증을 처리하는 반면 메시지 의미는 더 높은 실행 추상화 계층에 맡겨진다.

9.4. 컴파일 스택

통합된 블랙박스 애플리케이션 개발 인터페이스를 제공하기 위해, Anoma 구현에는 새로운 언어인 Juvix와 컴파일러 스택이 포함되어 있으며, 이는 개발자들이 공식적으로 검증된 장애 내성 분산 애플리케이션을 작성할 수 있도록 함께 설계되었다. 컴파일러 스택에 대한 많은 연구 작업이 남아 있으며 이 섹션은 진행 중인 작업으로 간주되어야 한다. 여기에 설명된 여러 구성 요소는 부분적으로만 구현되었으며 대안은 여전히 활발하게 고려되고 있다.

9.4.1. Juvix

Juvix는 단순 유형 람다 미적분학의 변형으로 컴파일되는 고급 함수형 언어다. Juvix 로 작성된 프로그램은 공개, 준비산, 비산 데이터와 연산을 표현하고 추론할 수 있다. Juvix의 람다 계산법 출력 언어는 C를 통해 RISCV 또는 WASM으로 컴파일되거나, VampIR의 입력 언어인 다항식으로

인스턴스화될 수 있는 AVM의 추상 범주적 연산으로 컴파일될 수 있다.

9.4.2. Anoma VM

AnomaVM(AVM)은 분산 추상 범주론적 가상 머신이다. AVM은 구체적인 암호화 표현/ 인스턴스화 또는 운영 실행 의미를 고정하지 않고 다자간 상호작용과 계산의 정보 이론적 의미를 포착하도록 설계되었다. AVM 프로그램은 역할에 따라 에이전트를 직접 참조 하며, 이들은 실행 및 인증 증명을 통해 서로의 상태 전환 및 상태를 추론할 수 있고 서로 메시지를 보내고 받을 수 있다.

AVM 프로그램 자체는 에이전트를 추상적으로만 지정하지만, 자신이 수행하려는 역할을 지정해야 하는 모든 에이전트가 실행(또는 컴파일 후 실행)될 수 있다(이는 일종의 로컬 네이밍 시스템임). AVM은 투명 실행을 위해 LLVM을 통해 컴파일될 수 있고, ZKP 또는 FHE 실행에 적합한 회로를 위해 VampIR을 통해 컴파일될 수 있다.

운영 중립적인 추상 표현으로서 AVM은 Anoma 아키텍처가 비용 의미와 프로그램의 동일성을 정의하는 수준이다. 예를 들어, 다른 당사자는 AVM 프로그램을 다른 구체적인 암호화 및 투명 백엔드로 컴파일할 수 있다.

9.4.3. VampIR

VampIR은 다른 구체적인 증명 시스템으로 컴파일될 수 있는 다항식, 회로 및 제약 시스 템의 추상적 표현을 제공하도록 설계된 언어 및 컴파일러다. IR은 ZKP와 FHE를 포함한 다양한 증명 시스템과 암호화 백엔드에서의 인스턴스화의 운영 의미에 대해 구애받지 않는 상태를 유지하면서 회로의 지시적 의미를 포착하도록 설계되었다.

9.5. 프랙탈 인스턴스 구성 요소

9.5.1. 시빌(Sybil) 저항

프랙탈 인스턴스는 합의에서 투표권을 할당하기 위해 시빌 저항 메커니즘을 제공해야 한다. 이는 지분 증명(PoS), 권위 증명(PoA), 하이브리드(부분적으로 대체 가능한) 지분 증명 또는 암호화 신원 기판을 기반으로 위임 민주주의(liquid democracy)의 한 형태일 수 있다.

9.5.2. 거버넌스

프랙탈 인스턴스는 그러한 시스템이 없는 경우보다 (다소) 더 정규적인 프로세스를 통해 비정규적인 상태 변경을 시행하기 위한 거버넌스 메커니즘을 제공할 수 있다. 이 거 버넌스 메커니즘 자체는 시빌 저항을 필요로 하며, 이는 합의에서 사용되는 것과 동일하거나 이에서의 약간의 변형의 형태일 수 있다.

9.5.3. 리소스 가격 책정

프랙탈 인스턴스는 개방형 네트워크에서 누구나 보낼 수 있는 메시지를 수신할 때 비용이 많이 드는 계산 작업을 수행하기 위한 시빌 저항 메커니즘을 제공해야 한다. 이 시빌 저항 메커니즘은 네트워크 토큰으로 지불하는 수수료, 신원 기반 할당량 또는 계산, 저장 등의 구독 또는 네트워크 기반 속도 제한과 결합된 낮은 고정 메시지당 제한을 기반으로 할 수 있다.

10. 향후 방향성

10.1. 복잡한 거래 상대방 발견

거래 상대방 발견, 공정성 및 데이터 지역성 간의 절충 등의 분야는 매우 근본적이다. 무작위 테스트보다 더욱 효율적인 방식으로 거래 상대방을 찾으려면 선호도에 대한 정보를 제공해야 하며, 불확실한 정보(예: 가변 가격)의 경우 더 큰 당사자 집합에 걸쳐 공정성을 제공하려면 더 많은 의도를 한 번에 볼 수 있고 가장 공정한 솔루션을 찾기 위해

경쟁할 수 있는 더 큰 해결사 집합에 선호도를 공개해야 한다.

암호화된 해결은 원칙적으로 가능하지만, 이미 NP 문제인 해결 문제와 중량급 HE의 오버헤드를 결합하므로 가까운 미래에는 실현 불가능할 가능성이 높다. 하지만 개선된 알고리즘, 애플리케이션별 솔루션, 전용 하드웨어에 대한 연구는 시간이 지남에 따라 이러한 오버헤드 비용을 낮출 수 있다.

10.2. 종단 간(End-to end) 행동 검증

Anoma의 아키텍처는 노드 소프트웨어를 운영하는 (추상적) 튜링 머신부터 의도를 작성하는 (추상적) 사용자까지의 영역을 다루며, 전자의 행동에 대한 특정 가정이 주어졌을 때 후자와 관련된 시스템의 행동에 대한 보장을 제공한다. 실제로 Anoma 배포의 안전한 사용은 이 시스템의 정확성뿐만 아니라 노드에서 사용하는 하드웨어의 정확성과 사용자가 사용하는 인터페이스의 정확성에도 달려있다. 결국 이 겆믕은 인터페이스와 하드웨어 영역으로 더욱 확장될 수 있다

.

11. 감사의 글

References

[1] 비트코인 위키, "비트코인 스크립트," 2021.

[2] V. Buterin, "이더리움: 차세대 스마트 계약 및 탈중앙화 애플리케이션 플랫폼," 2014.

[3] E. G. Weyl, P. Ohlhaver, and V. Buterin, "탈중앙화 사회: Web3의 영혼 찾기," SSRN 4105763에서 이용 가능, 2022.

[4] E. Buchman, J. Kwon, and Z. Milosevic, "BFT 합의에 대한 최신 가십," arXiv preprint arXiv:1807.04938, 2018.

[5] T. Rocket, "눈송이에서 눈사태까지: 암호화폐를 위한 새로운 준안정 합의 프로토콜 제품군," 온라인 이용 가능. [접속: 2018년 4월 12일], 2018.

[6] C. Goes, "블록체인 간 통신 프로토콜: 개요," arXiv preprint arXiv:2006.15918, 2020.

[7] J. Bebel and D. Ojha, "Ferveo: BFT 네트워크에서 멤풀 프라이버시를 위한 임계값 암호 해독," Cryptology ePrint Archive, 2022.

[8] M. Prewitt, "복수 화폐: 새로운 통화 설계," 2022.

[9] Heliax, "Typhon," 2022.

[10] I. Sheff, X. Wang, R. van Renesse, and A. C. Myers, "이질적 팍소스," OPODIS: 분산 시스템 원칙에 관한 국제 회의, 2021.

[11] G. Danezis, L. Kokoris-Kogias, A. Sonnino, and A. Spiegelman, "Narwhal and Tusk: DAG 기반 멤풀 및 효율적인 BFT 합의," 컴퓨터 시스템에 관한 17차 유럽 회의 회보, 2022, pp. 34-50.