

RASMART

WHITE PAPER

Ver 1.0

내용

1. Rasmart 시스템의 특성

2. 소개

2.1 문제 및 해결책

2.2 프로젝트에서 블록 체인의 역할

2.3 네트워크 레지스트리

3. 정의

3.1 네트워크 노드

3.2 마지막으로 저장된 블록

3.3 노트 동기화

4. 네트워크 합의

4.1 일치의 비교

4.2 주 및 작문 호스트의 개념

4.3 네트워크 장비

4.4 합의 도달

4.5 형성 및 레지스트리 초기화

4.6 등록되지 않은 트랜잭션

5. 거래 처리

5.1 거래

5.2 합의 찾기

5.3 트랜잭션 처리

5.4 레지스트리 입력 구조

5.5 RS 레지스트리 구조

5.6 블록 크기

5.7 거래 참여자 검색

5.8 데이터 링크

5.9 시스템 동작

5.10 유효성 검사에 트랜잭션 추가

5.11 거래 비용

6. Rasmart의 해부학

6.1 네트워크 구조

6.2 네트워크의 노드 식별 및 주소 지정

6.3 네트워킹

6.4 DHT

6.5 PEX

6.6 트래픽

6.7 레지스트리 구조

6.8 저장소와 머클 나무

6.9 트랜잭션 체인 및 블록 체인

6.10 무결성 동기화

7. 데이터베이스 엔티티

- 7.1** 지갑
- 7.2** 거래
- 7.3** 임의의 개체
- 7.4** 스마트 계약
- 7.5** 거래
- 7.6** 포크 문제 해결
- 7.7** 결정 블록
- 7.8** 합의의 실현
- 7.9** 트랜잭션 그래프의 병렬화
- 7.10** 트랜잭션 그래프를 부분으로 나누기
- 7.11** 수수료
- 7.12** 수수료 계산 방법

8. 스마트 계약

- 8.1** 소개
- 8.2** 개체
- 8.3** 스마트 계약 방식
- 8.4** 가상 실행 머신
- 8.5** 비용 표현
- 8.6** 스마트 계약 조건 이행
- 8.7** 데이터 소스
- 8.8** API

9. 보안 및 위협 가능성

- 9.1** 중개자의 공격
- 9.2** 공격 51
- 9.3** RSA 키 해킹
- 9.4** 시빌 공격
- 9.5** 이중 지출 문제
- 9.6** 트래픽 캡처 및 위장
- 9.7** 현지 보관소 변경

10. 계획 구현

- 10.1** 프로젝트 구현의 기술적 계획
- 10.2** ICO
- 10.3** Rasmart 암호 화폐

1. 시스템 특성

Rasmart는 블록체인 기술을 기반으로 한 플랫폼입니다. RS 전문 분야는 분산 레지스트리, 암호화 및 자체 실행 스마트 계약을 기반으로 금융 서비스를 사용하도록 권한을 부여합니다.

오늘날 블록 체인은 급속하게 발전하는 기술로서, Rasmart가 해결할 수 있는 많은 요구 사항을 발생 시켰습니다. 블록 체인 기반 플랫폼으로서 많은 개선점과 확장 기능을 제공합니다.

1. 거래율. 시스템 전체의 트랜잭션 수를 1초 단위로 나타내는 정 성적 특성.

2. 사용의 유연성. **Rasmart** - 유연하고 적응력있는 API를 제공하여 타사 서비스에 통합

3. 보안. 0.2 초의 차단 시간으로 인해 합의를 도출하기 위해 다양한 공격과 취약점이 수평을 이루고 있습니다.

4. 자기 스마트 계약. Rasmart 시스템의 스마트 계약은 시간이 많이 걸리는 (Lua)의 장점으로 빠른 실행뿐 아니라 거대한 계산 리소스, 큰 트랜잭션 용량 및 빠른 실행을 필요로하지 않는 작은 통역사와 다양한 기능을 갖춘 지속적으로 업데이트 된 SDK를 사용하므로 빠릅니다.

2. 소개

Bitcoin은 첫 번째 블록 체인 플랫폼으로서 참여자 들간의 새로운 분권화 된 상호 작용 방식의 개발에 자극을주었습니다. 사실, Bitcoin은 오히려 짧은 기간 동안 사용자들 사이에서 인기를 얻었습니다. Bitcoin 네트워크에는 중개자의 참여없이 사용자 간 교환이 하나만 있습니다.

Ethereum은 현명한 계약을 기반으로 작동하는 블록 체인 기술을 기반으로 모든 분산 된 온라인 서비스를 만들 수 있는 플랫폼입니다.

그러나 블록 체인을 기반으로 구축 된 첫 번째 플랫폼은 사용자로부터 아무런 인식을 얻지 못했고 개별 팔로워들만있었습니다. 또한 초당 작동 수가 적은 저속은 대중을 늘릴 수 없었습니다. 모든 후속 블록 체인 플랫폼은 Bitcoin에 처음 적용된 고전적 방식의 추종자였습니다.

오늘날, 유일한 금융 산업은 참가자 간의 분산 된 (직접적인) 상호 작용의 적극적인 실행에 저항합니다. 기술적이고 조직적인 관점에서 보면 분산 금융 서비스 시스템을 만드는 것이 고전적인 은행보다 쉽습니다. 이를 위해서는 필요한 지식과 경험이 모두 필요합니다.

따라서 분산 레지스트리를 기반으로 구축 된 가장 편리한 분산 형 금융 서비스 시스템을 만들려면 다음이 필요합니다:

1. 높은 데이터 처리 속도 (초당 수십만 건의 트랜잭션), 운영 비용은 최소화되어야합니다.

2. 분권화 된 금융 서비스 구현을 위한 모든 참가자와 요소를 통합 할 수 있는 사용자 시스템을 위한 단일의 작성: 개인 자금 지불 센터, 사용자 개인화, 신용 기록 국, 암호 유출 현금 인출 단말기 등. RS 플랫폼은 이러한 모든 작업을 위한 솔루션입니다.

Rasmart는 모든 금융 서비스 참가자를 통합하고 분산 된 레지스트리의 원칙을 기반으로 트랜잭션을 빠르고 안전하게 수행 할 수 있는 단일 분산 형 기술 플랫폼입니다. 연합 시스템 및 자체 이행 스마트 계약은 고유 한 솔루션을 작성하여 다양한 금융 상품 간의 상호 작용을 구축 할 수 있는 기회를 제공합니다. Rasmart는 금융뿐 아니라 기술적 인 (저속) 및 높은 거래 비용의 형태로 인한 재정적 제약으로 인해 분산 레지스트리를 사용할 수 없는 다른 산업에서도 블록체인 프로젝트를 사용할 수 있는 큰 잠재력을 발휘할 수 있는 기회를 제공합니다.

2.1 문제

네트워크 대기 시간은 많은 블록 체인 프로젝트의 주요 제한 사항 중 하나입니다. 단일 Bitcoin 트랜잭션의 평균 시간은 10 분 이상입니다. 동시에 은행 거래가 문자 그대로 즉시 발생하며 1 초가 넘지 않습니다.

기술적 문제

저희는 다음과 같은 문제를 강조합니다 :

대역폭은 네트워크가 주어진 양의 데이터를 처리하는 데 걸리는 시간입니다.

트랜잭션 거래 속도;

대기 시간 - 예상과 비교 한 시스템의 실제 응답 시간.

저장된 데이터의 양 - 네트워크에 저장된 데이터의 양 (**Ethereum**은 정기적으로 수십기가 바이트까지 증가합니다).

높은 비용은 모든 작업을 수행하기위한 핵심 매개 변수이며 특히 두 가지 주요 그룹입니다:

1. 사물의 인터넷 운영, 당신은 하나의 글로벌 네트워크에 다양한 항목을 결합 할 수 있습니다;
2. 거래 비용이 적은 소액 결제 - 예: 커피와 같은 소액 구매, 상점의 제품 지불, 모든 종류의 소액 결제.

이러한 모든 문제와 한계로 인해 금융 업계를 포함한 많은 산업 분야에서 블록 체인 기술의 확산이 어려워졌습니다.

블록 체인 기술을 사용하면 다음과 같은 속성으로 임의의 데이터베이스를 구현할 수 있습니다:

가용성. 분산 데이터베이스는 네트워크의 모든 노드에 동시에 저장됩니다. 즉, 나머지 네트워크와의 연결이 없더라도 모든 노드에서 모든 정보에 액세스 할 수 있습니다.

무결성. 블록 체인을 사용하면 데이터를 위조 할 수 없으며 사용자가 개입 사실 및 장소를 신속하게 감지 할 수 있습니다.

기밀 유지. 비대칭 데이터 암호화로 인해 수신자 만 정보에 액세스 할 수 있습니다. 전자 서명은 정보가 실제로 발신자에게 속해 있음을 보증합니다.

신뢰성. 각 노드에 데이터베이스를 저장하면 노드 중 하나가 실패한 후 정보를 완전히 복원 할 수 있습니다. 네트워크는 손상되지 않은 유일한 노드에서도 완벽하게 복구 할 수 있습니다.

확장 성. 새로운 노드를 도입함으로써 네트워크의 크기를 빠르고 쉽게 늘리거나 줄일 수 있습니다. 네트워크에 연결하려면 지금 등록하고 기존 노드에 연결하십시오.

통합. API 덕분에 분산 데이터베이스에서 정보를 수신하고 저장할 수 있는 모든 소프트웨어를 분산 데이터베이스에 연결할 수 있습니다.

위에 나열된 특성으로 인해 데이터베이스의 범위는 재무 환경을 넘어선다. 이는 안전한 정보 교환, 전자 문서 순환, 보고 자동화 및 다양한 정보 수집의 보편적인 방법입니다. 프로세스의 자동화에 많은 수의 참가자가 필요하고 높은 정보 저장 안정성이 요구되는 모든 산업에서 응용 프로그램이 필요할 것입니다.

2.2 프로젝트에서 블록체인의 역할

블록체인은 상호 연결된 정보 블록 체인입니다. 이 블록들 각각은 체인에 있는 이전 블록의 암호화 기능 (또는 해시)을 저장합니다. 우리가 두 블록 만 가지고 있다면 체인의 첫 번째와 마지막을 말하면, 그 사이의 모든 블록이 신뢰할 수 있다는 것을 100 % 확신 할 수 있습니다.

이 원칙에 따라 구축 된 데이터베이스를 사용하면 새 값을 기록하고 이전 값을 변경하지 않고 정보를 추가 할 수 있으므로 일관된 변경 내역을 만들 수 있습니다. 정보 삭제는 제공되지 않습니다. 따라서 데이터베이스의 모든 활동을 수정하고 정당성을 검사 할 수 있습니다. 레지스트리를 열려면 완벽한 솔루션입니다.

그러나 공개 데이터베이스조차도 기밀 정보를 저장할 수 있습니다. 받는 사람의 공개 키로 비대칭 암호화를 사용하면 키의 개인 부분 소유자에게만 액세스 할 수 있습니다. 충분한 키를 가지는 RSA 암호화 알고리즘으로 이것을 보증 할 수 있습니다. 나머지 네트워크는 정보 추가의 사실만 볼 수는 있지만 정보에 액세스 할 수는 없습니다.

플랫폼은 분산 레지스트리의 보호 및 동기화를 저장하는 기술 메커니즘을 사용자에게 제공합니다. 정보 저장 장치의 구조와 형식에는 제한이 없습니다. RSA는 데이터 레지스트리에 배치 된 연결성과 신뢰성에 대한 정보만 보완합니다.

3. 정의

- 1. 시스템** - 트랜잭션을 처리, 저장 및 전송하고 스마트 계약 조건을 충족 및 확인하는 분산 네트워크 노드 세트입니다. 타사 시스템의 요청을 처리하고 요청에 대한 정보를 제공하십시오.
- 2. 네트워크 노드** - 전체 네트워크 클라이언트가 설치된 하나의 컴퓨터는 공통 시스템에 연결되어 트랜잭션을 수행하고 데이터를 저장 및 전송합니다.
- 3. 레지스트리** - 네트워크 노드 트랜잭션에서 확인되고 저장됩니다.
- 4. 트랜잭션** - 스마트 계약의 하나 또는 다른 방법을 실행하라는 요청. 블록 체인에 추가 기록.
- 5. 스마트 계약** - 상호 작용 조건을 확인하고 준수하는지 확인하는 컴퓨터 프로토콜. 대부분의 경우 스마트 계약은 신뢰 관계의 논리를 모방합니다. 중앙 집중식 소스로부터의 분권화와 독립성은 현명한 계약의 핵심 속성입니다.
- 6. 스마트 계약 메소드**는 스마트 계약의 작업을 계산하고 수신 된 데이터를 레지스트리에 쓰는 것을 담당하는 프로그램 코드입니다.
- 7. 계약 당사자는** 시스템의 사용자 및 네트워크의 최종 참가자입니다.

3.1 네트워크 노드

노드의 무료 액세스 및 연결을 기반으로 독립적이고 완전 분산 된 네트워크를 구축하기 위해 목적에 따라 한 번에 여러 유형의 노드를 사용합니다.

1. 일방 노드 - 유효성에 대한 트랜잭션 검증에 참여하지만, 최소 신뢰도를 가지며, 네트워크에서 노드의 역할을 선택하는 다음주기 동안 현재 처리를 위한 신뢰 노드 또는 노드의 역할을 얻는 후보입니다.
2. 트러스티드 노드 - 트랜잭션 검증에 참여하며, 주된 신뢰 요인을 가지고 있습니다.
3. 네트워크의 주 노드 - 정상 노드에서 전송 된 트랜잭션을 누적하고 이를 신뢰할 수 있는 노드에 분배합니다.
4. 기록 네트워크 노드 - 블록을 형성하는 라운드 중에 선택된 신뢰할 수 있는 노드 중 하나가 데이터베이스에 기록하고 해시 백을 수신하는 네트워크의 모든 참가자에게 전송하고 새로운 라운드를 형성합니다.

3.2 마지막으로 저장된 블록

GBL (블록의 일반 레지스터)는 시스템의 모든 노드에 있는 블록의 일반 레지스터의 완전히 동기화된 상태입니다.

레지스트리 블록의 내용은 여기에 저장된 정보 단위이며, 이 정보에는 이 레지스트리와 관련된 데이터 목록뿐만 아니라 이전 블록의 해시 코드가 포함됩니다. 다른 노드로부터 블록을 수신 한 후, 번호에 따라 블록의 일반 레지스터에서 그 자리를 차지합니다. 따라서 네트워크 대역폭을 크게 절약 할 수 있습니다.

동기화 중에 블록 번호 만 먼저 확인됩니다. 이 노드의 블록이 누락 된 경우 하드 디스크에 다운로드되어 저장됩니다.

이 방법 덕분에 시스템의 모든 유닛에는 항상 관련 정보 만 포함됩니다. 우리는 그것을 LL이라고 부릅니다 - 즉, 마지막 레지스트리. LL은 컨센서스를 찾은 후 레지스트리를 형성하는 노드에 의해 자동으로 생성됩니다. 이 블록은 시스템의 모든 노드로 전송되므로 결과적으로 시스템의 모든 노드가 최신 레지스트리를 유지 관리합니다. 네트워크의 모든 노드는 상호 연결되어 있으며, 그 사이에는 일정하고 불용 할 수 없는 정보 교환 (블록 및 트랜잭션)이 있습니다. 따라서 특정 트랜잭션 집합이 블록으로 구성되어 레지스트리에 추가됩니다. 이 때 각 서버는 다른 서버에 대해 제안 된 제안 집합을 구성합니다. 확인 후에 레지스트리에 추가할지 여부를 결정합니다.

결국 많은 수의 서버 (시스템의 개별 노드)에서 레지스트리 데이터를 여러 번 저장하는 것이 가능하지만 모든 정보는 신뢰할 수 있는 보호를 받습니다. 그리고 시스템의 노드가 많을수록 노드의 신뢰성이 높아집니다.

4. 네트워크 합의

RS합의는 그룹 의사 결정 기법으로, 네트워크의 모든 노드에 대해 최종적으로 수용 할 수 있는 목적을 개발하는 데 목적이 있습니다.

4.1 레지스트리비교

서로 다른 유형의 합의를 비교하려면 먼저 RS 분산 레지스트리의 원칙을 정의해야합니다.

레지스트리 접근성. 모든 노드는 언제든지 레지스트리에 데이터를 쓰고 레지스트리에서 읽을 수 있습니다.

수정 가능성. 네트워크의 모든 노드가 변경 될 수 있습니다.

일관성. 예외없이 시스템의 모든 노드는 서로 일관성이 있습니다. 즉, 동일한 버전의 레지스트리를 보게됩니다.

분리에 대한 저항. 하나 이상의 노드에 장애가 발생해도 시스템의 성능에는 영향을 미치지 않습니다.

4.2 기본 및 작가 네트워크 노드의개념

네트워크의 모든 노드는 분산되어 있으며 다른 노드보다 장점이 없습니다.

적절한 수준의 데이터 스토리지 보안을 보장하고 트랜잭션 처리 속도 및 추가 정보 전송을 향상시키기 위해 Rasmart 플랫폼은 독점적인 결합 프로토콜을 사용합니다.

작성 노드는 생성 된 블록을 모든 시스템 참가자에게 전송 한 후 생성 된 트랜잭션 블록에 서명하고 이를 레지스트리에 저장합니다. 각 시스템 참가자는 들어오는 블록을 레지스트리에 기록하고 기록 된 블록의 해시를 다시 쓰기 노드로 보냅니다. 작성 노드는 주 노드와 신뢰 노드의 새 목록을 작성하고 생성 된 목록을 모든 네트워크 참여자에게 전송하여 Rasmart 플랫폼의 새로운 라운드를 만듭니다.

4.3 네트워크 노드 장비

우리의 목표는 트랜잭션 처리 속도가 가장 빠른 플랫폼을 만드는 것으로, 강력한 서버 하드웨어 및 고 대역폭 인터넷을 희생하면서 최상의 조건에서 네트워크 노드를 유지하기 위해 중요한 인센티브를 사용할 것을 제안합니다.

물질 보상으로 각 노드 소유자는 처리 된 레지스트리의 거래 비용에서 RS 통화로 보상을 받습니다. 균형은 신뢰할 수 있는 노드 (BFT 합의에 참여한 노드)간에 분산됩니다. 백분율은 적어도 3 년 동안 동전을 처음으로 배치 한 후 동전을 처음으로 배치 한 후에 개별 네트워크 노드 사이의 연방 투표를 통해 비율을 형성하는 시스템으로 변경할 수 있습니다.

따라서 서버 소유자는 처리 속도가 빠를수록 특정 노드의 소유자가 더 많은 수익을 얻을 수 있기 때문에 훨씬 더 강력한 장비에서 이 레지스트리를 유지하는 것이 좋습니다.

4.4 합의 도달

시스템의 주 노드는 트랜잭션 배터리의 역할과 시스템의 신뢰 노드에 대한 트랜잭션 송신자를 수행하고, 신뢰 노드는 생성 된 트랜잭션 블록에 서명하는 쓰기 노드를 트랜잭션을 검증하고 생성 된 블록을 모든 시스템 참가자에게 보낸 후 레지스트리에 저장합니다. 접수 된 정보의 프로세스 및 처리와 최신 실제 레지스트리를 기반으로 한 형성 - 이것이 합의의 성취입니다. 마지막 실제 레지스트리의 형성 결과는 합의의 결정입니다.

이 프로세스는 여러 단계로 나눌 수 있습니다:

- 주 노드를 검색합니다;
- 신뢰할 수 있는 사이트 목록 작성하기;
- 거래 목록을 가져 와서 레지스트리에 추가로 추가 할 수 있는 후보자 목록을 작성하기;
- 후보자 목록 처리하기 (노드간에 투표를 통과 하기);
- 시험에 합격하지 않았거나 부정적인 확인을 받은 미확인 거래의 후보자 목록에서 제거하기;
- 레지스트리에 추가 할 확인 된 트랜잭션 목록 구성하기;
- 트랜잭션을 포함하는 블록의 해시 코드를 사용하여 레지스트리에 트랜잭션을 추가합니다;
- 네트워크에 참여하는 모든 노드에 트랜잭션이 있는 완료된 블록을 전송하면, 이 블록은 자동으로 모든 장치의 레지스터에 추가합니다;

4.5 레지스트리 형성 및 초기화

이 프로세스는 다음 순서로 설명 할 수 있습니다:

- 최종 네트워크 구성원은 트랜잭션을 형성합니다;
- 모든 조건이 충족되면 사용자는 특수 플랫폼 소프트웨어에서 원하는 방법을 호출하여 트랜잭션을 시작합니다;
- 유효성 검사기의 커널은 동기화의 모든 단계와 현재 레지스트리 버전의 불변성을 모니터링합니다;
- 합의에 도달하면 모든 거래가 하나의 단위로 수집됩니다;
- 각 블록은 해시 코드로 변환 된 레이블과 노드 식별자로 구성된 자체 번호를 수신합니다. 다음으로, 블록은 합의 발견 모듈에 위치합니다;
- 후보 목록이 컴파일 된 후 트랜잭션 및 블록의 해시가 레지스트리에 기록되며, 소스의 신뢰성을 항상 확인할 수 있습니다;
- 이 해시는 일종의 고유 한 블록 서명이며, 트랜잭션이 생성 된이 블록의 방법과 시기에 대한 자세한 정보를 포함합니다;
- 통합 검색 알고리즘을 사용한 후에도 합의가 이루어지며 블록에 포함 된 트랜잭션은 검토를 위해 유효성 검사기의 코어에 제출 된 다음 레지스트리에 기록됩니다.

4.6 레지스트리에 없는 트랜잭션

완료된 트랜잭션의 목록에서 끝나지 않은 모든 트랜잭션은 거부 된 것으로 표시됩니다. 이 이벤트에 대한 정보는 거래 개시자에게 즉시 표시됩니다. 레지스트리에 포함되지 않은 트랜잭션은 거부됩니다.

5. 거래 처리

5.1 거래

트랜잭션이란 계약 방식의 성능에 대한 정보를 전달하거나 스마트 계약을 생성 할 필요없이 자산간에 직접 전송하는 시스템의 최소 단위를 의미합니다. 미래에는 결과가 피어 - 투 - 피어 네트워크에 배치됩니다.

5.2 합의 발견

저희 시스템은 신뢰할 수 있는 노드를 투표하여 합의에 대한 해결책을 찾는 데 페더레이션 모델을 사용합니다. 또한 합의에 대한 해답을 찾는 알고리즘 - 유한 상태 머신의 도착 알고리즘이 적용됩니다. 컨센서스 자체는 주기적으로 작동합니다. 각주기 트랜잭션은 네트워크에서 추출되어 풀에 저장됩니다. 또한 모든 트랜잭션은 반환 응답을 받기 위해 신뢰할 수 있는 사이트로 전송됩니다. 응답이 수신되는 경우 레지스트리에이 유효성 검사기를 추가하라는 요청이 트랜잭션에 대해 이루어집니다. 합의의 형태로 체인의 종점에 도달하고 이전의 적법성에 대한 완전한 확인이 이루어지면 트랜잭션은 레지스트리에 추가 입력을 위해 레이블이있는 유효성 검사로 추가로 전송됩니다.

5.3 트랜잭션 처리

분산 된 시스템의 성격을 이루기 위해 각 서버는 레지스트리를 위한 대용량 스토리지가 있어야 할뿐만 아니라 모든 트랜잭션을 위한 본격적인 생산적인 처리기가되어야합니다.

우리 시스템에서는 이러한 개념을 "시스템 코어"로 사용합니다. 시스템의 핵심은 시스템의 다른 참가자의 성능에 관계없이 할당 된 작업을 수행하는 데이터 프로세서를 의미합니다. 입력시, 각 커널은 태스크를 수행하기위한 변수 목록을 수신합니다. 종료시 커널은 결과를 음수, 양수, 오류 또는 기타로 제공합니다. 주 데이터 세트 외에도 응답 코드는 항상 시스템 코어에 포함됩니다. 이러한 구조는 각 프로세스의 최대 속도에 필요할 수 있으며 서로 독립적으로 작동해야합니다.

5.4 레지스트리 입력 구조

보안을 손상시키지 않으면 서 적절한 수준의 시스템 성능을 얻으려면 이전 블록의 해시 코드와 트랜잭션 결과로부터 Merkle 트리보기를 생성 할 필요없이 레지스트리 데이터베이스를 사용하는 것이 좋습니다.

Merkle 트리는 체인의 고유 식별자를 얻고 올바른 블록 순서를 복원하기 위해 데이터의 무결성을 검사하는 데 사용되는 해시 유형입니다. 들어오는 정보는 별도의 블록으로 나누며 리프 타이거 해시를 사용하여 개별 해시됩니다. 그런 다음 내부 타이거 해시가 해시 쌍마다 계산됩니다. 해시에 쌍이 없으면 변경되지 않고 더 이상 전달되지 않고 이미 새 체인으로 전송됩니다. 이 절차는 하나의 해시가 최종적으로 표시 될 때까지 반복됩니다.

Merkle 트리를 사용하면 레지스트리 작업이 크게 느려지므로 트랜잭션 처리 속도가 느려서 편안하다고 할 수 없습니다. 동시에 낮은 최적화로 인해 컴퓨팅 리소스에 매우 큰 부하가 발생합니다. 따라서 고속을 유지하려면 나중에 비용을 지불하지 않는 강력하고 값 비싼 많은 노드가 필요합니다. 우리는 이것이 매우 비합리적인 데이터 저장 장치의 사용이라고 생각합니다.

5.5 RS 레지스트리 구조

우리는 RS 시스템에서 트랜잭션 레지스트리를 사용하는 것을 제안합니다. 즉 Merkle 트리를 완전히 포기합니다. RS 레지스트리의 각 항목은 레지스트리에 대한 후보 목록에 추가 할 수 있도록 승인 된 트랜잭션 블록 해시 코드로 구성됩니다. 또한, 그러한 기록에는 형성 시간뿐만 아니라 노드 식별자도 지정됩니다. 레지스트리 항목에는 거래의 방향뿐만 아니라 초기 및 최종 계정, 차변 단위 수, 차변 유형, 차입 유형 및 마지막에 적립 될 단위 수가 포함됩니다. 이 접근 방식을 사용하면 트랜잭션 처리 속도를 크게 향상시키고 레지스트리의 불법적 인 변경의 복잡성을 증가 시키며 레지스트리의 항목을 마지막 번호로 변경하는 가능성을 없앨 수 있습니다. 즉, 정보 처리 프로세스는 가능한 한 신속하고 정확하게 발생합니다.

5.6 블록 크기

i메인 노드와 트러스트 된 노드의 검색주기는 시간의 단위로 작동하고 사이클 시간은 네트워크 자체의 복잡성을 기반으로 계산됩니다. 단위 시간당, 이전 처리주기가 종료 된 순간부터 다음 처리주기가 시작될 때까지 네트워크에 추가 처리를 위해 형성되고 전송 된 N 개의 트랜잭션이 있습니다. 네트워크에서 선택된 N 개의 트랜잭션이 블록으로 이동합니다. 블록 크기는 그 안에있는 트랜잭션의 총 수에 따라 다릅니다.

5.7 거래 참여자 검색

RS 피어 - 투 - 피어 시스템은 정점이 사용자 계정 인 그래프로 표현 될 수 있으며, 지시 된 에지는 계정의 두 정점을 연결하는 시스템에서 만들어진 트랜잭션입니다. 그리고 모든 모서리에는 시작과 끝이 있기 때문에 이를 기반으로 유향 그래프를 만들 수 있습니다.

신원 확인을 위해 다음 조건을 충족하는 경우 :

- 완료된 각 트랜잭션에는 보낸 사람과 받는 사람이 모두 있습니다;
- 계정의 모든 꼭지점은 항상 방향 엣지 (즉, 거래)를 사용하여 다른 꼭짓점과 연결됩니다;
- 각 정점에는 제한된 수의 지향 에지가 있습니다.

유한 쇠사슬을 건설한다는 목적으로 거래를 수행하는 데 필요한 모든 조건을 충족하기 위해 원하는 경로가 열에 포함되어 있다고 결론 지을 수 있습니다.

단순 체인은 반복되는 정점이 없는 다이 그래프의 경로입니다. 그래프 자체는 이 시점까지 알려지지 않았기 때문에 그래프 이론부터 시작하여 알려지지 않은 그래프를 따라 경로를 만들어야 합니다. 그래프 데이터 클래스에서 탐색의 길이는 $\mathcal{O}(nm)$ 과 같으며, n은 정점의 수이고, m은 그래프의 가장자리 수임이 알려져 있습니다. 임의의 그래프에 대해, 길이 $\mathcal{O}(nm)$ 의 탐색이 존재하며, 가능한 최소 탐색 길이 $-Q(nm)$ 를 갖는 그래프도 있다.

모르는 그래프를 가로 지르는 것은 처음에는 그 토플로지가 알려지지 않았기 때문에 그래프를 따라 이동하는 과정에서만 이를 알 수 있습니다. 각 꼭지점에서 정확히 어떤 엣지가 나오는지 눈에 띄게 알 수 있습니다. 처음부터 끝까지 횡단하여 어떤 꼭지점이 엣지로가는지 알아낼 수 있습니다. 사실, 이것은 미로 내부에 위치한 로봇에 의해 미로를 우회하는 작업과 유사하지만 빠져 나갈 계획이 없습니다. 상태 수가 제한적이라면 로봇은 유한 상태 기계입니다. 이러한 로봇은 테이프가 그래프로 대체 된 이른바 튜링 기계의 완전한 아날로그이며, 셀은 꼭지점뿐만 아니라 그래프의 가장자리에도 직접 부착됩니다.

5.8 데이터 링크

적절한 수준의 네트워크 보안을 보장하기 위해 유효성 검사기 노드 간의 데이터는 암호화 된 형식으로 전송되며 노드 간의 각 연결에는 네트워크 라이브러리를 기반으로하는 낮은 수준의 연결이 제공됩니다. 데이터 전송 중에 고장이나 오류가 발생하면 스트림이 자동으로 중단되고 오류 레코드가 먼저 로깅 시스템에 기록 된 다음 로그 파일에 기록됩니다. 모든 데이터는 유형이 지정된 변수를 통해 전송됩니다. 모든 트랜잭션 데이터는 RC4 대칭 알고리즘을 사용하여 암호화됩니다. 그리고 이 알고리즘의 작업은 공유 비밀 키를 통해 이루어지며 키 자체는 노드 간의 연결을 생성하는 동안 암호화 된 형태로 전송됩니다.

5.9 시스템 동작

시스템에서의 조치는 하나의 계정에서 다른 계정으로의 가치 이동 또는 전송이 가장 쉬운 거래를 의미합니다. 시스템에서 컨센서스를 더 검색하기 위한 데이터 전송 검사기.

동일한 ID를 가진 동일한 블록에서 트랜잭션의 중복을 피하기 위해 시스템은 정당하게 올바른 식별자 만 처리되도록 처리하기로 결정했습니다. 유효성 검사기 시스템에는 이미 계정에 대해 트랜잭션이 이루어 졌으므로 합의를 찾을 수 없다는 항목이 이미 있습니다. 따라서 이중 낭비의 문제를 해결할 수 있습니다.

트랜잭션이 완료되고 유효성 검사기가 확인에 대한 정보를 받으면 레지스트리의 상태 변경에 대한 데이터가 목록의 모든 노드에 자동으로 전송되고 레지스트리는 결국 동기화를 시작합니다. 트랜잭션을 항상 활성 상태로 유지하려면 매번 모든 노드의 레지스트리에서 들어오는 트랜잭션을 다시 동기화해야 합니다. 그리고 이 문제를 성공적으로 해결하려면 동기화를 위해 별도의 포트를 사용해야 합니다 (그러한 기회가 있는 경우). 이 모든 것은 포트에서의 작업 부하의 효과적인 분배로 인해 유효성 검사기의 코어에서 들어오는 정보의 처리 속도를 상당히 증가시킵니다. 동기화 흐름은 연속적이고 주기적입니다.

이 경우, RAM 분배의 우선 순위와 CPU의 로드는 평균 이하입니다. 따라서 1000 개의 작업을 메모리에 저장할 수 있을 뿐 아니라 계정 상태도 저장할 수 있습니다. 이 모든 것이 계산 프로세스를 최적화 할 수 있게 해 주며 결과적으로 요청에 대한 응답 속도를 최적화합니다.

5.10 유효성 검사에 트랜잭션 추가

레지스트리에 트랜잭션을 추가하는 것은 유효성 검사를 통과 한 후에만 가능합니다. 즉, 합의가 발견되고 레지스트리에 있는 트랜잭션 결과의 흰색 목록이 컴파일 된 후에 가능합니다. 보안을 강화하기 위해 타사 시스템의 호출이 차단됩니다.

들어오는 매개 변수 - 진행중인 트랜잭션을 특징 짓는 개체입니다. 결과 매개 변수-`<0-ResultValue / 0 <ResultValue` - 함수가 오류 없이 실행되었으므로 결과는 레지스트리의 항목 번호입니다.

수신 매개 변수는 트랜잭션의 고유 레이블, 송신자와 수신자에 대한 데이터, 비용의 일치, 전송 된 값의 양, 전송 된 시스템 정보의 양을 포함하는 오브젝트입니다.

5.11 거래 비용

시스템은 자체 화폐 단위 RS를 사용합니다:

- 시스템을 사용하기 위해 청구되는 내부 지불 수단;
- 시스템 내에서 환전을 수행합니다;
- 시스템 자체 내에서 가치를 교환합니다;
- 현명한 계약 하에 거래를 생성하고 추가 처리합니다;
- 타사 리소스에서 데이터를 구입합니다;
- 시스템 자체 내에서 서비스 비용을 지불합니다.

시스템의로드 수준에 따라 트랜잭션 비용이 다를 수 있습니다. 재료 방법을 레버로 사용하여 네트워크의 부하를 관리하는 방법을 제안합니다.

시스템의 첫 3 년 동안 거래 비용은 운영 유형 및 거래량에 따라 달라집니다. 이미 가까운 장래에 위의 요인에 기초하여 완전 자동 모드에서 거래 비용을 형성하는 알고리즘이 개발 될 것입니다.

6. Rasmart의 해부학

개념적으로, Rasmart는 노드의 구성 요소 인 노드 (플랫폼의 주요 요소)와 최종 사용자 (지갑, 모니터, 오라클)를 위해 만들어진 다수의 마이너 모듈을 노드가 제공하는 api로 작업합니다.

6.1 네트워크 구조

네트워크는 UDP 프로토콜 (OSI 모델의 네 번째 수준)을 통해 메시지를 교환 할 수 있는 노드로 구성됩니다.

위상 적으로 네트워크는 어드레싱에 고유 한 노드 식별자를 사용하는 오버레이 네트워크 (이더넷을 통한)이며 라우팅은 노드 식별자에서 수집 된 정렬 된 방향 그래프를 따라 수행됩니다.

오버레이 네트워크와 실제 네트워크 인프라의 매팅 및 라우팅은 노드 소프트웨어의 전송 레이어에 의해 완전히 수행됩니다.

6.2 네트워크의 노드 식별 및 주소 지정

각 네트워크 참여자는 받는 사람의 공개 키로 암호화하고 보낸 사람의 개인 키로 서명하기 위해 한 쌍의 비대칭 암호화 키를 가지고 있습니다.

노드 및 네트워크 트래픽의 식별은 공개 키의 해시 기능을 기반으로합니다.

Diffie-Hellman 알고리즘을 사용하는 암호화 및 전자 서명의 경우 RSA (Rivest-Shamir-Adleman)는 소수의 키 길이 인 1024 비트 또는 ECDHE (임시 키가있는 타원형 곡선의 알고리즘 유형)를 통해 사용할 수 있습니다.

두 번째 경우에는 256 비트의 길이를 갖는 키가 유사한 암호화 강도에 충분합니다.

6.3 네트워킹

네트워크는 양방향 순환 그래프입니다. 노드 식별자는 각 노드에 대해 "왼쪽에 이웃" 및 "오른쪽에 이웃"과 같은 방식으로 링에 정렬되고 고정됩니다.

노드는 메시지를 생성하여 패킷을 모든 이웃에게 보냅니다. 그들은 이웃 사이에서 패키지 수신자를 찾고 있습니다. 메시지를 찾으면 직접 메시지를 처리합니다. 그렇지 않다면 왼쪽 패킷을 받은 사람들은 그것을 오른쪽의 모든 이웃과 왼쪽의 모든 이웃에게 보냅니다.

트래픽 리던더링은 UDP 손실, 메시지 경로상의 노드 분리 및 패킷 위조 시도를 방지합니다.

이 방식은 모든 노드가 언제든지 UDP 패킷을 수신 할 수 있는 경우에만 작동합니다. 이를 위한 주요 요구 사항은 라우터가 없는 플랫 네트워크와 계층 적으로 중첩 된 사설 네트워크이며 소위 "회색 주소"가 있습니다. NAT 뒤의 노드는 자신의 요청에 응답 할 때만 UDP 트래픽을 수신 할 수 있습니다. 이 요청은 이 요청이 처리 된 사람에게만 전달되며 라우터의 경로는 라우터 ARP 테이블 항목의 유효 기간입니다. 노드가 이러한 조건을 고려하고 동작을 조정하면 네트워크에 참여할 수 있습니다. 들어오고 나가는 트래픽을 분석함으로써 노드는 따라야 할 행동 모델을 이해할 수 있습니다.

이 분석의 기준은 호스트가 이러한 메시지보다 먼저 보낸 적이 없는 사람으로부터 메시지를 수신하는지 여부입니다. 그렇지 않은 경우, 노드는 유지되어야 하는 "열린 창"의 간격으로 정보를 수신 할 수 있도록 작업을 정렬합니다.

6.4 DHT

링 조직은 해시의 균일 한 배포를 사용하여 주소 목록에서 이웃을 찾고 이들 사이에 새로 생성 된 노드를 동적으로 웨지 할 수 있습니다.

토렌트 네트워크에서 사용되는 DHT 알고리즘은 이웃 선택을 위한 메커니즘을 보완 할 수 있는 "메트릭" 개념으로 작동하므로 현실에서는 서로 다른 대륙에 있는 서로 다른 데이터 센터에 있습니다. 이것은 "Sibyl의 공격"에 대한 방어이며, 결정 블록 전체를 포착하고 네트워크를 제어 할 수 있습니다.

반대로, 노드의 응답 속도가 주어지면 메트릭은 상호 작용 속도를 최적화 할 수 있으므로 네트워크의 속도는 실제로 가까운 노드를 희생시키면서 증가합니다.

실제로 보안의 관점에서 충분히 신뢰할 수 있고 트랜잭션 용량의 관점에서 충분히 빠른 "황금색 평균"에 대한 검색이 필요합니다.

6.5 PEX

노드 교환 메커니즘 (피어 교환)을 사용하면 새 노드가 연결될 때 네트워크를 재구성 할 수 있습니다. 순서대로 그래프에 삽입하면, 노드는 그 위치를 찾고, 이웃 노드는 자신의 이웃 노드의 서브 세트를 재분배하여 UDP의 "파장"이 거의 동일하게 유지합니다.

6.6 트래픽

네트워크는 UDP 수준에서 작동합니다. 노드에 의해 생성 된 패킷에는 명령의 송신자, 수신자, 명령 및 임의의 페이로드 (페이로드)가 있습니다. 브로드 캐스트 메시지를 만들려면 존재하지 않는 받는 사람 주소 (예 : 0)를 지정하면 됩니다.

많은 명령이 노드에 네트워크에 등록하고, 레지스트리를 동기화하기 위해 누락 된 정보를 요청하고, 트랜잭션을 작성하고, 결정 블록에 참여할 응용 프로그램을 보내며, 결정 블록에 참여하는 데 필요한 명령을 제공합니다.

메시지를 생성하는 노드는 수명이 짧은 서명으로 서명합니다. 이 시그니처의 특별한 특징은 자체 수명 (TTL)을 가지고 있으며 더 작고 UDP 패킷에서 많은 공간을 차지하지 않는다는 것입니다. 짧은 TTL을 감안할 때 이는 보안상의 문제는 아니지만 그러한 패킷의 확인 및 처리 속도를 높이기 때문에 무단 배포 또는 "외부" 패킷 전송이 불가능합니다.

단명 서명의 확인은 네트워크 수준에서 이루어지며 거부 된 패킷은 첫 번째 리디렉션보다 진행되지 않으며 노드 소프트웨어의 상위 수준에서 계산되지 않습니다.

대형 패키지는 여러 개의 작은 패키지로 나누어져 있으며, 개별 패키지는 개별 메시지로 전송되며 수신자의 노드에서만 함께 수집됩니다. 이는 차단에 대한 추가 보호 조치입니다.

6.7 레지스트리 구조

레지스트리는 임의의 데이터의 데이터베이스이며, 인스턴스는 각 노드에 저장되고 복제됩니다. 이 데이터 위에는 정보의 무결성, 신뢰성 및 관련성에 대한 책임이 있는 추가 구조가 구축됩니다. 기술적으로 이베이스는 임의의 값에 대한 키가 해시 인 "키 - 값" 저장소 (key-value)로 구성됩니다. 로컬 데이터베이스 스토리지는 Google의 임베디드 레벨 DB 데이터베이스입니다.

6.8 저장소와 머클 나무

데이터 코드 위에 이진 트리가 만들어지며, 각 노드는 하위 수준 해시의 쌍으로 연결된 해시와 같습니다.

따라서 전체 데이터 트리가 전체 데이터베이스의 상태를 표시하는 하나의 "루트" 해시로 "축소"됩니다. 전체 트리가 검사되고 승인되지 않은 변경을 수행하는 경우 해시가 변경되어 더 이상 다른 노드의 동일한 해시와 일치하지 않으므로 데이터베이스의 무결성을 위반하게 됩니다. 이것은 수정 된 머클트리 알고리즘입니다.

6.9 트랜잭션 체인 및 블록 체인

트랜잭션 체인은 데이터베이스에 대한 변경 사항을 설명하고 데이터베이스의 현재 상태 (머클 트리의 루트 해시)를 나타내는 레코드 목록입니다. 레코드는 블록 작성자의 전자 서명에 의해 서명된 블록으로 결합되고 이전 블록의 해시를 포함하여 소위 블록 체인 ("블록 체인"- 영어 "블록의 체인"에서)이라는 단순한 링크 된 목록을 형성합니다. 이 메커니즘은 이미 입력 된 데이터를 변경으로부터 보호하고 이러한 변경을 합법적으로 만든 사람을 인증합니다.

6.10 동기화 및 무결성 제어

데이터베이스는 key-value 세트이므로 노드 간의 단일 레코드 교환으로 데이터베이스 동기화가 감소됩니다. 트랜잭션 체인 또는 Merkle 해시가 "수렴 (converge)"을 중단 할 때마다 변경되거나 누락 된 레코드의 위치가 결정되고 네트워크에서 요청됩니다. 이렇게하면 최신 변경 사항을 업데이트 할 수 있을 뿐 아니라 네트워크에 노드가 오래 있지 않거나 로컬 데이터베이스에서 되돌릴 수 없는 데이터가 손실 된 후 데이터베이스 상태의 초기 동기화 및 복원을 보장 할 수 있습니다.

7. 데이터베이스 엔티티

7.1 지갑

지갑은 네트워크 회원의 개인 계정입니다. 참여자는 돈을 송수신하기 위해 실행 노드와 한 쌍의 전자 서명 키가 필요합니다. 지갑 ID와 노드 ID는 공개 키 해시입니다.

7.2 거래

전송을 수행 할 때 네트워크 참여자는 트랜잭션을 생성하고 개인 키로 서명 한 다음 네트워크로 전송합니다. 금융 거래는 발신자, 수신자, 이전 금액, 운영 통화 및 임의의 추가 데이터에 대한 정보가 있는 엔터티입니다. 이러한 트랜잭션은 블록 체인에 속하며 데이터베이스의 정확성을 모니터링하기 위한 메커니즘의 일부입니다.

무결성 모니터링은 노드가 시작될 때마다 수행되고 데이터베이스의 로컬 복사본의 계산 된 상태가 네트워크에서 받은 최신 레지스트리 상태와 일치하지 않는 경우에도 수행됩니다. 모든 위반이 동기화주기를 트리거합니다.

7.3 임의의 개체

전송 정보 외에 레지스트리에는 다른 형식의 정보도 포함될 수 있습니다. 그러나 당신은 그것을 직접 저장할 수 없습니다.

그러한 각 엔티티에는 스마트 계약을 맺고 레지스트리에서 데이터를 작성, 변경 및 삭제할 수 있는 소유자가 있습니다.

7.4 스마트 계약

스마트 계약은 저장 프로시저로 데이터베이스에 저장됩니다. 이것은 식별자로 액세스 할 수 있는 프로그램 코드입니다. 스마트 계약 ID는 소스 코드의 해시입니다.

7.5 거래

참가자가 생성 한 트랜잭션은 아직 블록 체인의 일부가 아니며 트랜잭션 체인에 포함되지 않습니다. 그 전에 그들은 정확성을 검사해야합니다. 이 수표는 두 가지로 나옵니다. 송금자가 송금 할 금액보다 많은 자금을 보유해야하며 송금자의 개인 키로 거래가 서명되어야합니다. 비 전달 트랜잭션은 삭제되고 확인 된 블록에서 생성 된 블록이 유효성 검사기에서 서명하고 블록 체인에 포함됩니다.

7.6 «포크» 문제 해결

블록 체인은 본질적으로 "포크"(영어의 "fork"에서 "포크")에 취약합니다. 단일 링크 목록이 있는 곳에서는 나무가 있을 수 있습니다. 여러 블록이 하나의 부모를 참조하지 못하게하는 것은 없습니다. 레지스트리에 쓰고 난 후 블록을 변경할 수 없으므로 이중 링크 된 목록에서와 같이 후속 블록에 대한 링크를 "닫을" 수 없습니다.

다른 블록 체인처럼, 우리는 언제든지 네트워크에 하나의 레코딩 블록 만있을 수 있다는 사실로 이 문제를 해결합니다. 유효성 검사기 중 누가 검증 라운드에서 해결 될 것입니다.

7.7 결정 블록

트랜잭션의 검증 및 새로운 블록의 보존은 결정 블록에 의해 수행됩니다. 이것은 서로 독립적으로 검사를 수행하고 솔루션의 결과를 서로 교환하는 노드의 하위 집합입니다. 신뢰할 수 없는 환경, 즉 최종 사용자가 제어하는 노드가 있는 분산 된 네트워크에서는 처음에는 신뢰할 수 없었습니다. 이러한 메커니즘을 통해 손상된 노드를 탐지하고 올바른 결과를 얻을 수 있습니다.

이 방법의 기본은 "비잔틴 장군의 임무"의 결정입니다. 이 명령은 부분적으로 신뢰할만한 데이터를 기반으로 올바른 결정을 내릴 수 있도록 합니다.

7.8 합의의 실현

모든 노드가 제출 된 트랜잭션이 유효한지 여부에 대한 정보를 교환 한 후에 컨센서스가 트랜잭션의 유효성을 결정합니다. 기록 할 준비가 된 모든 트랜잭션에 대한 결정을 한 후에, 블록을 기록하는 노드가 선택되고, 그 후에 결정 블록은 새로운 노드에서 새로 형성되고 모든 것이 반복됩니다. 이를 통해 분기에 대한 보호가 이루어집니다. 네트워크에 항상 새로운 블록을 추가하는 노드가 하나만 있습니다.

7.9 의사 결정 블록의 병렬화

의사 결정 블록은 둘 이상일 수 있습니다.

수집 된 모든 트랜잭션이 하위 집합으로 나뉘어 질 경우 모든 준비 작업을 동시에 수행 할 수 있으며 주기에서 의사 결정주기까지 레코드 권한은 한 노드에서 다른 노드로 릴레이를 통해 전송됩니다. 이러한 계획에서 기록은 거의 연속적이며 네트워크 트랜잭션 용량은 여러 번 증가합니다.

7.10 트랜잭션 그래프를 부분으로 나누기

노드가 전송자와 수신자인 연결된 그래프의 형태로 트랜잭션 세트를 표현하고 이들 사이의 연결이 전송량과 동일한 가격을 가지면 그래프를 최적으로 하위 세트로 나눌 수 있습니다.

이를 위해 각 부분 집합은 각 노드의 결과 균형이 음이 아닌 그래프를 형성해야 합니다. 작업은 "배낭 포장 문제" 해결로 줄었습니다. 이 기능의 목적은 트랜잭션 그래프의 일반적인 비영성을 위반하는 최소 트랜잭션 수를 버리는 방식으로 준비된 트랜잭션 목록을 "완료"하는 것입니다. 이 단계에서 삭제된 트랜잭션은 재확인 및 "누적"을 위해 다음 라운드로 전송됩니다.

7.11 수수료

수수료는 네트워크 참여자가 네트워크 기능을 지원하는 데 컴퓨팅 리소스를 사용하도록 유도하는 방법입니다. 노드가 생산적인 장비에서 작동하고 고품질의 인터넷 채널을 가지고 있다는 것은 흥미롭습니다. 신속하게 대응할 수 있는 노드는 의사 결정 블록에 들어가고 완료된 작업에 대한 커미션을 받을 가능성이 큽니다.

7.12 수수료 계산 방법

이 테이블에는 거래 발급을 위한 수수료 금액(송금 자로부터 시스템이 부과하는 가격)이 형성되는 이유 목록이 포함되어 있습니다.

라운드때문에 거래 수수료의 의존성

- | | | |
|------|--|---|
| I. | <ul style="list-style-type: none">• 라운드에서 거래수• 라운드에서 거래수가 많을수록 수수료는 작습니다 | <ul style="list-style-type: none">• 최소 라운드에서 거래수 1입니다• 명목 가치 = 10,000 라운드에서 거래• 한 라운드에서 처리되는 트랜잭션 수의 상한은 65 536으로 설정됩니다 |
| II. | <ul style="list-style-type: none">• 라운드에서 신뢰할 수 있는 노드수• 라운드에서 신뢰할 수 있는 노드수가 많을수록 수수료는 더 큽니다 | <ul style="list-style-type: none">• 명목 가치 – 101 |
| III. | <ul style="list-style-type: none">• 물리적 거래 규모• 블럭 체인 디스크에 더 많은 공간이 필요하면 수수료는 더 큽니다 | <ul style="list-style-type: none">• 물리적 거래 규모 부터 ...까지...• 명목 가치. 제한이 없다. |
| IV. | <ul style="list-style-type: none">• 구매자 거래 활동• 보낸 사람이 생성하는 트랜잭션이 많을수록 커미션이 커집니다. | <ul style="list-style-type: none">• 트랜잭션 활동의 하한선0입니다.• 명목 가치 – 5 000 – 10 000• 트랜잭션 활동의 상한선 초당 1,310,720 입니다. |

8. 스마트 계약

스마트 계약은 수신자가 식별자를 지정하는 트랜잭션이 만들어 질 때 실행되는 프로그램 코드입니다. 스마트 계약 ID는 소스 코드의 해시입니다.

스마트계약을 발행하면 소스 코드가 포함 된 트랜잭션이 생성됩니다. 스마트 계약서는 데이터베이스에 저장되며 식별자가 수신자 인 네트워크에 거래가 나타날 때 실행됩니다. 실행되면 스마트 계약서는 데이터베이스의 일부 항목을 생성, 수정 또는 삭제하므로 호출간에 상태를 저장합니다.

또한 스마트 계약은 블록 체인에서 정보를 읽고 트랜잭션을 생성 할 수 있습니다. 예를 들어 스포츠 경기의 베팅을 수집하는 스마트 계약서로이 이벤트에 대한 정보가 데이터베이스에 나타나면 형성된 "은행"의 총 분포에 비례하여 상금을 분배합니다. 그가 결정을 내리게 될 기초에 관한 정보는 다른 스마트 계약에 의해 기지에 공급된다. 따라서 스마트 계약의 본질은 외부 세계의 모든 사실, 스포츠 경기의 결과, 공증인에 의한 문서의 법적 인증 사실 등이 될 수 있습니다.

스마트 계약은 Lua 어로 작성되었으며 고립 된 가상 머신 내에 모든 기능을 갖추고 있습니다. 표준 Lua 라이브러리는 사용할 수 없으며 스마트 계약은 외부 세계와 상호 작용할 수 없습니다.

스마트 계약은 특별히 작성된 SDK의 틀 내에서만 필요한 모든 기능을 제공합니다.

수신자가 스마트 계약 식별자 인 트랜잭션을 수신하면 결정 블록의 참가자는 코드를 실행하고 완료되면 모든 변수 및 생성 된 엔터티를 포함하여 가상 시스템의 상태에 대한 해시를 교환합니다. 의사 결정 블록의 모든 참가자에 대한 스마트 계약 실행 결과가 수렴되면 트랜잭션이 블록 체인에 저장되고 모든 수정 된 레지스트리 엔터티는 네트워크를 통해 동기화됩니다.

스마트 계약은 알고리즘 적으로 제한되어 있지 않으며 기생 또는 오류 코드로 인해 네트워크가 중단되어 영원한 주기로 이어질 위험이 있습니다. 이를 피하기 위해 스마트 계약에는 엄격하게 제한된 실행 시간이 주어지며 그 이후에는 트랜잭션이 중지되고 트랜잭션은 유효하지 않은 것으로 간주됩니다. 그러나 실행 비용은 결정 블록의 참가자들에게 청구되고 분배됩니다.

스마트 계약이 변경할 수 있는 엔티티는 관리자에 의해 초기화되어야 합니다. 다른 모든 개체를 그는 읽을 수 있지만 쓸 수 없다.

스마트 계약에는 입력 및 출력이 없으므로 네트워크 요청을 수행 할 수 없습니다. 그들이 할 수 있는 모든 것은 엔티티의 레지스트리 사본을 읽고 엔티티를 생성, 수정 및 삭제하는 것뿐입니다.

8.1 가상 실행 머신

루아 인터프리터는 프로그램이 순차적으로 실행되는 고립 된 레지스터 기반 가상 머신을 제공합니다. 가상 시스템 내부에서는 주변 운영 체제의 리소스를 사용할 수 없으므로 모든 라이브러리 (표준 입출력 라이브러리 포함)를 명시 적으로 제공해야합니다. 따라서 가상 시스템은 특별히 작성된 SDK에만 액세스 할 수 있으며, 예를 들어 파일 또는 네트워크 연결을 설정할 수 있습니다.

8.2 비용 표현

Rasmart 암호화는 계약의 한 단위 비용이며, 이후의 합의를 위해 완전히 다른 두 단위의 비용을 비교할 수 있습니다. RS 암호화는 근본적으로 값 간 전송을 위한 번들의 역할을 합니다. 이 접근법은 모든 값이 RS 암호화와 관련하여 액체라는 사실로 인해 가능했습니다.

8.3 데이터 소스

RS 시스템은 가장 정확한 검증과 추가 정보를 명확히하기 위해 타사 데이터 제공업체를 사용하여 적절하고 완전한 작업을 수행합니다. 이것은 계약 당사자 중 한 사람에 관한 비공개 정보 때문입니다. 예를 들어, 차용인의 신용 상태를 파악하여 신용 기금 부여 결정을 내릴 수 있습니다.

이 시스템은 통합 버스를 호출 할 수 있는 능력을 갖추고 있으며, 실제로 100 개의 타사 정보 시스템을 사용하여 작업 할 수 있습니다. 이 버스는 물론 다른 참가자의 데이터를 유료로 제공하는 형태로 제 3 자 시스템에 대한 요청을 독립적으로 생성합니다. 암호화 RS는 지불로 사용됩니다.

8.4 API

각 노드는 네트워크에 요청하기 위한 REST API를 제공합니다. 쿼리는 트랜잭션 생성, 스마트 계약 서게시, 트랜잭션 목록 읽기, 레지스트리에서 값 검색 등의 작업을 수행 할 수 있습니다. API는 로컬에서만 요청을 받아들이며 네트워크와 분산 레지스트리와 상호 작용할 소프트웨어는 사용자 자신의 로컬 노드가 있어야합니다. 전송을 수행하는 응용 프로그램 인 지갑은 동일한 워크 스테이션의 바로 옆에 설치된 노드의 API를 통해 네트워크와 통신하는 타사 소프트웨어의 예입니다.

9. 보안 및 가능한 위협

요청은 정보 시스템이 이전에 제공 한 주소로 암호화 된 형태로만 전송됩니다. 결정을 내리는 데 필요한 최소의 정보를 포함하는 모든 서비스 응답은 효과적인 것으로 간주됩니다.

9.1 중재자 공격

보낸 사람을 인증하는 트랜잭션의 전자 서명으로 해결됩니다.

9.2 공격 51

네트워크가 커짐에 따라 유효성 검사 라운드를 치는 확률은 무시해도 됩니다.

9.3 RSA 키 해독

2048 비트 (256 바이트)의 RSA 키의 신뢰성은 수학적으로 입증되었습니다. Shor의 양자 알고리즘에 의한 인수 분해 공격은 양자 컴퓨터가 없기 때문에 무의미합니다.

9.4 시빌의 공격

손상된 노드가 있는 100 % "환경"의 경우에만 가능합니다. 신뢰할 수 있는 노드 하나만으로 공격을 탐지 할 수 있습니다. 이 프로세스는 또한 신뢰할 수 있는 유효성 검사기 노드의 원을 선택하는 과정에서 엔트로피 요소가 복잡합니다.

9.5 이중 폐기물의 문제점

한 번에 한 노드 만 블록을 쓰게되면 네트워크가 분기로부터 보호되므로 동일한 돈을 두 번 사용할 수 없게됩니다.

9.6 교통 차단 및 대체.

동일한 패키지가 반복적으로 복제되어 다른 경로로 이동합니다. 트래픽을 차단하려면 전체 네트워크 환경을 제어해야합니다. 인터넷의 경우, 이것은 있을 법하지 않습니다. 패키지에는 짧은 수명의 패킷을 인증하고 시간이 지남에 따라 "팽창"하기에 충분한 전자 서명의 경량 버전도 제공됩니다.

9.7 로컬 저장소 변경.

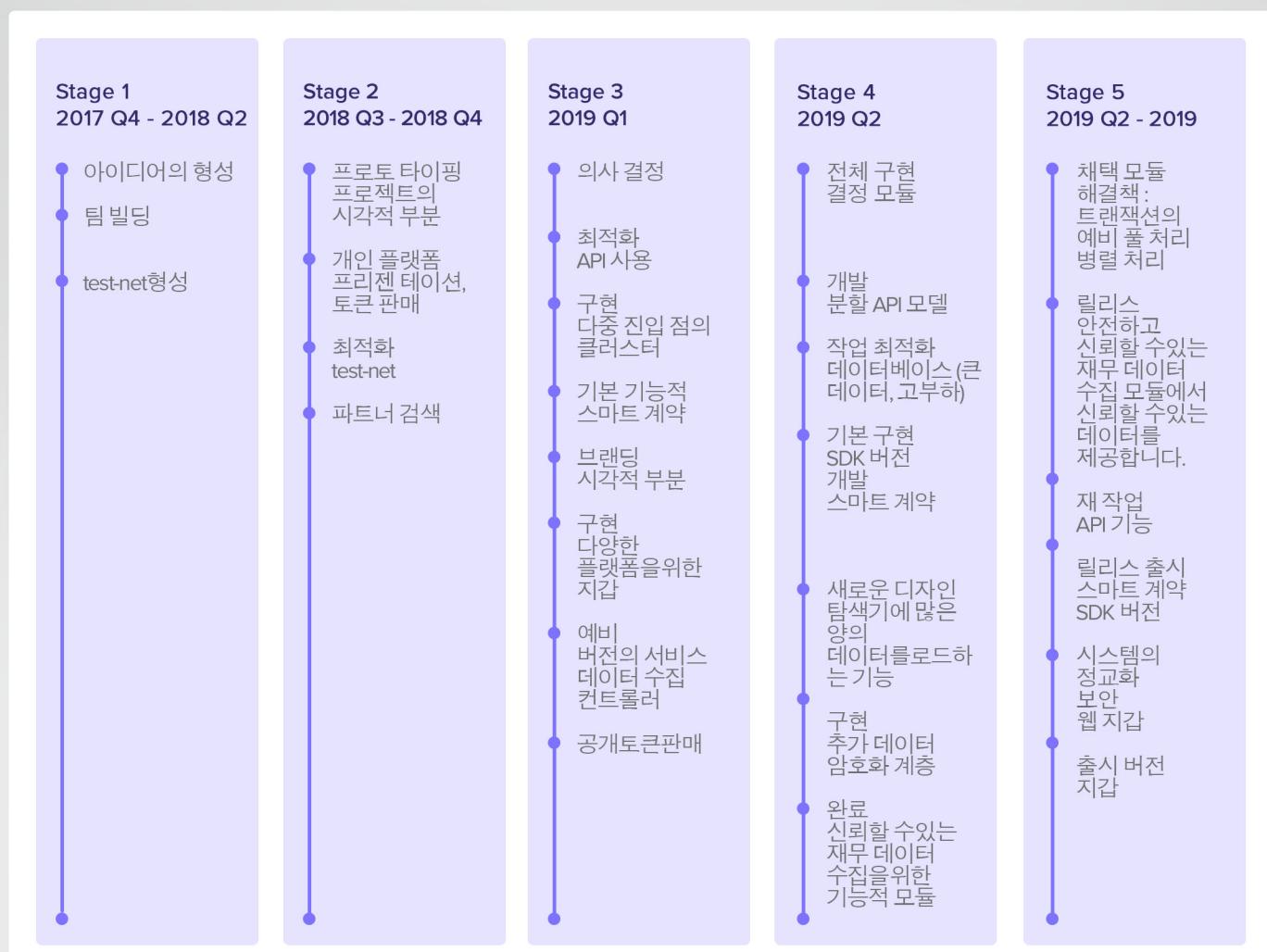
로컬 스토리지에 개입하면 데이터베이스의 상태가 다시 계산됩니다 (Merkle 트리 및 마지막 블록의 루트 해시). 이러한 데이터가 현실과 일치하지 않으면 네트워크가 복구 될 때까지 손상된 노드와의 상호 작용을 거부합니다.

10. 이행 계획

10.1 기술 프로젝트 이행 계획

10.2 ICO

로드맵



Rasmart 초기 코인 제공

토큰 세부 정보:

토큰 : RAS 토큰 유형 :
유틸리티

토큰의 특성 :

토큰 가격 : \$ 0.23
하드 캡 : \$ 50,000,000
토큰 최대: 500,000,000 RAS
최소 구매 거래: \$
가능한 화폐 : ETH

공개 판매 :

500,000,000 RAS - 토큰의 기본 릴리스
225 000 000 RAS - ICO
67 000 000 RAS - 마케팅 및 자문
90 000 000 RAS - 플랫폼 지원 5
59 000 000 RAS - 플랫폼 지원
59 000 000 RAS - 팀 및 파트너

ICO 다이어그램

단계	기간, 일	최소 구매, USD	보너스, %
비공개 판매	60	10 000	30
사전 판매	20	5 000	20
주요 판매 단계	10	500	5-10

* – 모든 단계의 대규모 구매의 경우 보너스를 받을 수 있습니다. 토큰은 회사 기금 또는 팀 비율에서 할당됩니다.

10.3 Rasmart 암호화

시스템의 작동 버전을 시작한 후 고정 된 수의 토큰이 500,000,000 RS 통화 단위로 발행됩니다. 앞으로는 초기 판매시 발행 된 ERC20 토큰과 교환 할 수 있습니다. 교환은 고정 비율, 즉 1ERC20 표준 토큰 = 1RS 통화로 할 수 있습니다.

ICO 이후의 계획 - 거래 속도를 초당 400,000으로 높입니다.

스마트 계약의 현재 메커니즘은 쉽지만 동시에 충분한 생산 언어인 Lua를 기반으로합니다. 그의 안전은 의심의 여지가 없었다. 팀과 관련하여 금융 거래를 위한 언어에 대한 심층적인 보안 감사를 수행하고 가상 시스템을 보호하는 방법을 연구합니다.

차 수준의 라우팅 메커니즘을 개발하여 네트워크 속도를 크게 높일 계획입니다. 이렇게하면 각 노드에 대해 가능한 가장 빠른 경로를 빠르게 검색하는 데 도움이됩니다.

