Bitcoin 2.0 : A Peer-to-Peer Electronic Cash System

"비트코인이라는 유령이 전 세계를 거닐고 있다."

https://wiki.p2pfoundation.net/Nakamoto_Satoshi
Nakamotosatoshi610@gmail.com
www.microbitcoin.org

1. 서문

2009년 비트코인의 첫 번째 블록을 생성했다. 우리가 거둔 성과는 첫 번째, 이중 지불 문제를 제3자의 개입 없이 해결했다는 것 두 번째, 디지털에도 하나의 원본만이 존재하게 할 수 있다는 것 세 번째, P2P로 전자화폐 거래를 할 수 있다는 가능성 네 번째, 기존 컴퓨터의 데이터베이스 시스템이 해결하지 못한 소수점 이하의 숫자 계산을 염려하지 않아도 된다는 것을 전 세계 시민들에게 보여주었다.

지난 1971년 닉슨쇼크로 시작된 신용화폐 시스템의 역사는 고작 50년을 넘기고 있을 뿐이다. 성공적이라고 말하는 이도 있지만 대부분 학자는 그렇게 바라보지 않는다. 신용화폐 자본주의 시스템은 신용화폐 즉 부채를 기반으로 인쇄된 채권에 불과하다는 것이 정설이고 신용화폐를 뒷받침하는 것이 강력한 국가 폭력이라는 것을 우리는 잘 알고 있다.
영원히 상환되지 않는 국채는 팬데믹과 같은 전 세계 전염병 확산으로 위축된 소비를 늘린다는 목적으로 이전 유통량의 3배에 해당하는 화폐를 발행했다.
지금의 실물에서 보이는 화폐는 이미 교환 가치가 계속 떨어지는 인쇄된 종이일 뿐이다. 너무나 한꺼번에 팽창된 전 세계 통화량은 부의 불평등을 심화시키고 국가간의 불평등을 만들게 되며 하이퍼-인플레이션을 당연히 동반하게 된다.
지금 당신이 가진 1달러의 화폐가 오늘 가장 큰 교환 가치를 가진다. 즉 무제한 찍어내는 신용화폐는 결국 신용화폐를 가지고 있는 동안 계속 교환 가치가 줄어드는 역사적 교훈을 우리 인류는 경험적으로 가지고 있다.

비트코인의 전자화폐 시스템이 논문과 코드로 구현된 이후 한 세력으로부터 시험에 들었다. 가장 먼저 미국을 중심으로 한 '오타쿠' 자유 지상주의자들이 비트코인 재단을 장악하고 모든 상품에 대한 P2P 거래와 개인의 문제를 국가로부터 통제받는 것을 저항하는 무기로 비트코인을 사용했다.
그들은 대표적으로 '실크로드'를 비롯한 토르(TOR) 브라우저 기반의 다크웹에서

거래 수단으로 사용되는 정당성을 부여했다. 또한 '사토시 다이스'같은 사행성 게임을 통해 다른 사람이 채굴한 비트코인을 모았고 결국 그 게임에 대해 잘 모르는 파키스탄 부자에게 지분을 팔아 자본을 축적했으며 이 게임으로 인해 비트코인 시스템은 DDos공격을 받는 것과 같은 네트워크의 과부하를 유발시켰다. 이들은 이 자본을 통해 초반에 생성된 극소수 커뮤니티를 기반으로 자본에 대한 통제에 국가가 개입하는 것도 부정하며 전세계 시민들의 접근을 철저히 차단했다.

이것의 아이러니는 중앙 통제를 받지 않겠다고 하며 스스로는 중앙 통제를 강화하여 자본을 기하급수 적으로 늘리는 방식을 취했다. 자유 지상주의자 자신들이 그 중앙통제자가 된 것이다. 대표적으로 그들이 만든 거래소가 그렇다.
이들은 비트코인의 1 CPU 1 VOTE 원칙에 통제받지 않는 자본 증식을 위해서 과감히 포기하였고, 고성능 주문형 반도체를 만들고 누구나 채굴에 참여하는 것을 제한시켰고, 심지어는 '하드포크'를 통해 '화폐 복사'를 하여 자본을 급격히 증식시켰다.

또한 비트코인과 같은 코드를 기반으로 이미지와 채굴량만 조절하여 새로운 코인이라며 만들었다. 이는 비트코인의 코드 신뢰를 더욱 떨어트리는 것으로 작용했고 지금도 여전히 이들의 행위는 지속되고 있다.
그 결과, 현재의 비트코인 시스템은 초기 설계와는 너무나도 동떨어진 중앙 집중형 시스템으로 변했다. 지급 결제 시스템이 아니라 많은 전력 소비와 강력한 반도체에 의해 시스템이 유지되며, 몇몇 자유 지상주의자에 의해 투기적 가치 저장 수단으로 자본주의 금융 산업에 포섭되어 버렸다. 소수의 채굴기 제조업체, 강력한 자본으로 만들어진 채굴업체, 자유 지상주의자가 자기들 만의 리그로 현재 비트코인을 자유를 대표하는 신화로 만들었다.

두 번째는 아나키즘과 공공선을 얘기하며 여러 가지 실험을 진행하는 부류이다. 물론 자유 지상주의자들보다 전 세계 시민을 생각한다는 면에서는 도덕적인 명분을 가지고 정의를 실현하려는 선의에서 블록체인 프로젝트에 참여한다. 아주 작은 프로젝트지만 커먼즈(commons)를 얘기하거나 협동조합 운동과 P2P의 본류를 회복하려고 한다. 하지만 전체 암호화폐 시장은 이미 자유 지상주의자의 자본에 포섭되어 자본의 힘을 막기에는 역부족이며, 인수·합병당하거나 기술을 탈취당하기도 해서 자본주의 사회에 대한 영향력과 변화를 만들기에는 미약한 수준이다.

하지만 모든 변화와 혁명은 항상 주변부와 약한 고리에서 시작된다.
단지 가치를 전달하고 저장하는 수단으로 제3세계의 가난한 인민을 도울 수 있는지, 그리고 전 세계 인류에게 공통의 교환 가치를 원활하고 저렴한 방식으로 제공하는 P2P 플랫폼은 가능한지가 지금의 현실에서 가장 중요한 논쟁거리가 될 것이다.

비트코인과 우리가 선택한 비트코인 2.0 버전인 마이크로비트코인은 각자 내포한 숫자에 여러 의미를 담고 있다. 단순히 2,100만 개만 채굴한다고 무작위로 설정된 것이 아니며 소수점 이하 8자리가 무슨 의미인지, 4년의 반감기도 우연히 만들어진 숫자가 아니다. 마찬가지로 마이크로비트코인의 채굴량이 610억 개라는 것과 마이크로비트코인 레이어-2에서 만들어지는 토큰의 수수료도 다 우리 인류 역사에서 중요한 숫자의 의미를 담고 있다. 마이크로비트코인이 낸 수수께끼의 정답을 많은 개발자와 사용자들이 알아내길 바란다.

2. 기술적 설명에 앞서

비트코인 2.0이라고 불릴 수 있는 마이크로비트코인의 레이어-2의 의미는 첫 번째, 메인넷인 마이크로비트코인은 교환 가치를 가지지 않는다는 것이다. 그 많은 발행량에도 불구하고 단지 레이어-2의 토큰, ST 등 거래의 수수료로만 사용될 뿐이다. 마이크로비트코인 자체로는 그것으로 결제하거나 교환 가치로써 사용될 수 없고 단지 커뮤니티의 의사 결정을 위한 가치를 가지게 될 것이다.
두 번째 특징은 프로그래밍 언어를 모르더라도 누구나 마이크로비트코인 플랫폼 위에서 몇 번의 클릭만으로 새로운 토큰 혹은 ST 등을 생성할 수 있다. 그리고 한 번 발행한 티커(Ticker)를 모방하여 같은 이름을 가진 토큰을 만들 수 없고, 토큰 생성 시 토큰 관리자가 발행한 토큰을 더 발행하거나 소각할 수 있도록 옵션을 설정할 수 있어서 많은 응용 분야에서 활용할 수 있도록 개방성과 가용성을 넓혔다.

우리는 마이크로비트코인을 통해 진정한 의미의 P2P 전자화폐 결제 시스템을 세계 최초로 레이어-2로 구현함으로써 본래의 비트코인 탄생의 목적에 도달하려고 한다.
혹자는 '비트코인과 별개로 처음부터 이것을 구현하는 플랫폼을 만들면 되는 것이

아니냐'는 질문을 할 수 있을 것이다. 하지만 우리는 지금까지 만들어진 암호화폐 중 비트코인 철학과 코드를 계승하면서 동시에 가장 유연하고 창의적인 기술력을 갖춘 '비트코인 2.0' 프로젝트를 구현하고자 한다.

전 세계의 많은 개발자가 함께하여 새로운 제안과 창의적인 아이디어를 녹여 낼 수 있기를 진정으로 바란다. 물질적 자원은 제한되어 있으니 공동체에 의해 관리되어야 하며 비물질적 자원은 제한 없이 넓게 퍼트려서 새로운 지평을 열어가는 계기로 삼아야 한다. 우리는 커먼즈 정신에 입각하여 세계 공동체의 더욱 큰 발전을 이뤄낼 수 있는 도구로서 이 프로젝트를 확장시켜 갈 것이다.

> **기술요약**: 마이크로비트코인은 현실의 금융 환경에서의 결제를 위한 탈중앙화된 블록체인이다. 비트코인 UTXO를 상속하며 하드포크로 구현되었다. 프로젝트를 시작한 지 약 1년 후 비트코인 네트워크에서 상속된 광범위한 크기의 블록체인과 블록 유효성 검사 중 PoW 알고리즘의 성능 저하 등 몇 가지 개선할 사항이 발생하였다. 2019년 10월 9일 커뮤니티는 이러한 문제를 해결하기 위해 기존 네트워크를 근본적으로 전환하는 새로운 네트워크를 구현하였다. 새로운 마이크로비트코인 네트워크는 UTXO(Unspent Transaction Output) 스냅샷, 더 작은 블록 사이즈, 새로운 블록 보상 공식 및 CPU 기반의 Pow 알고리즘을 특징으로 한다.

3. 새로운 네트워크 론칭의 전제 조건

최초의 마이크로비트코인 네트워크는 2018년 7월 11일 비트코인 네트워크의 하드포크로 시작되었다. ASIC[1] 저항성과 소액 결제에 적합한 빠른 블록 시간에 초점을 맞추었다. 실제 통화 단위와의 상호 작용이 용이하도록 1BTC를 10,000 MBC로 만들었다.

최초의 마이크로비트코인 블록은 2018년 7월 11일에 채굴되었으며, 당시에는 ASIC 구현이 없었기 때문에 ASIC 저항성을 가진 것으로 알려진 SHA256d 해시

[1] https://en.bitcoin.it/wiki/ASIC

함수를 NIST SHA-3 Groestl[2] 알고리즘으로 대체하여 하드포크를 일으켰다. 하지만 Baikal[3]이 2018년 10월 26일 Groestl을 지원하는 BK-G28을 발표하였고, BK-G28 채굴자들은 마이크로비트코인 네트워크 해시 파워의 주요 공급원이 되어 탈중앙화의 가치를 근본적으로 손상시켰고 이로 인해 우리의 가정을 수정되어야만 했다. 광범위한 연구를 거듭한 결과 우리는 슈나이더(Bill Schneider)의 Rainforest[4](RFv2라고도 한다) PoW 알고리즘을 통해 마이크로비트코인 네트워크를 2019년 3월 6일 1차 하드포크에 이어 2019년 5월 7일 2차 버전으로 강화하였다.

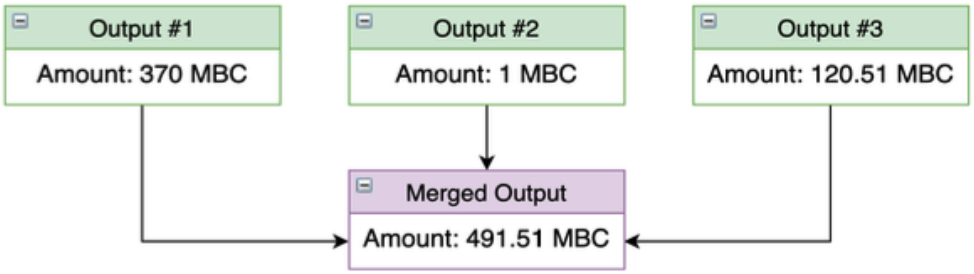
그리고 얼마 후 Rainforest v2 알고리즘은 PoW 유효성 검증 단계에서 속도가 너무 느려지고 200GB가 넘는 블록체인 사이즈 문제가 결합되어 마이크로비트코인의 전체 노드를 동기화/유지하는 것을 어렵게 만들었고 본질적으로 네트워크의 탈중앙화를 저해한다는 것이 드러났다. 이것은 우리는 새로운 네트워크를 향한 도전을 시작한 이유이다.

4. 스냅샷 (Snapshot)

마이크로비트코인 네트워크는 최종 주소 잔액이 기본적으로 모든 미사용 출력값의 합인 UTXO[5] 모델에서 작동하기 때문에 한 네트워크에서 다른 네트워크로 잔액을 이동하는 것은 간단하다.

우리는 525,000 블록(첫 번째 MBC 블록)부터 1,137,200 블록까지 모든 UTXO를 가져와 복사하고 병합했다. 예를 들어 이전 네트워크 주소에 3개의 미사용 출력값이 있는 경우, 금액의 합계를 사용하여 하나의 출력값으로 병합하였다.

예시:



[2] https://www.groestl.info

[3] https://bitcointalk.org/index.php?topic=5057818.0

[4] https://www.slideshare.net/bschn2/the-rainforest-algorithm

[5] https://www.investopedia.com/terms/u/utxo.asp

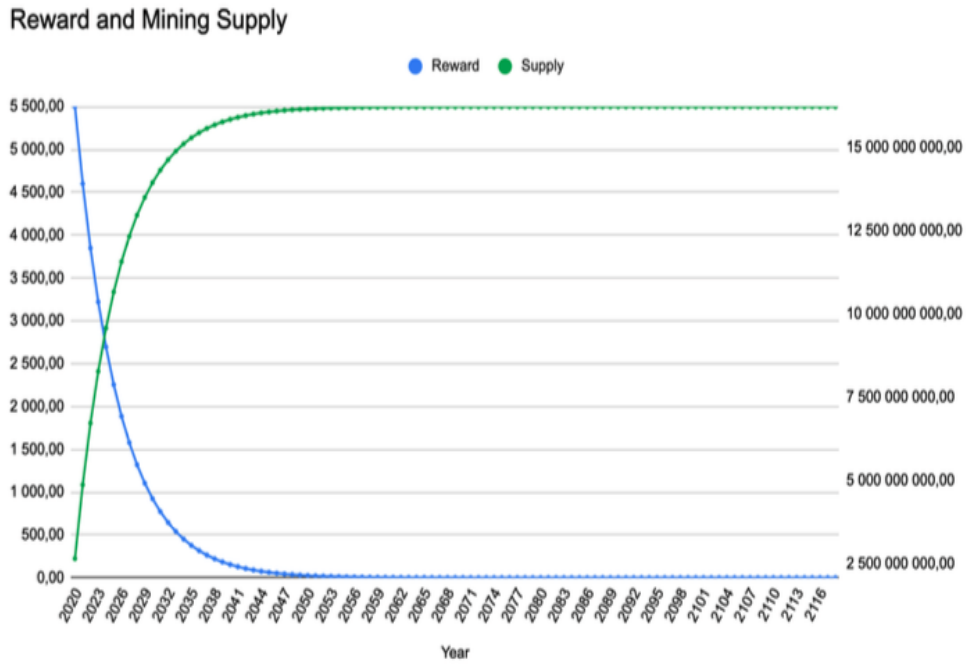
모든 스냅샷 출력값은 새로운 마이크로비트코인 네트워크의 제네시스(genesis)[6] 블록에 위치하며 익스플로러에서 확인할 수 있다.

5. 공급과 보상 (Supply and emission)

새로운 네트워크 출시 시점에서 최초 네트워크(525,000 블록) 론칭 이후 이동되지 않은 코인을 스냅샷하지 않고 소각함으로써 현 사용자 대비 총 공급이 과도하게 책정되어 있는 문제를 해결하였다. 총 44,386,397,362.4252 MBC가 활성화 되었고, 약 2,700,000 BTC가 하드포크 이후 이동했다.

새로운 코인의 더 나은 분배를 위해 블록 보상 일정이 조정되었다. 블록 보상을 4년마다 50%씩 감소시키는 반감기(Halving)[7] 대신에 새로운 블록 보상을 자연스럽게 감소시킨다. 기본 보상은 약 2년의 에포크(epoch)마다 30%씩 감소하고 있다.

보상 및 채굴 공급량에 대한 그래프:



C++에서 보상 공식 구현.

---

[6] https://en.bitcoin.it/wiki/Genesis_block

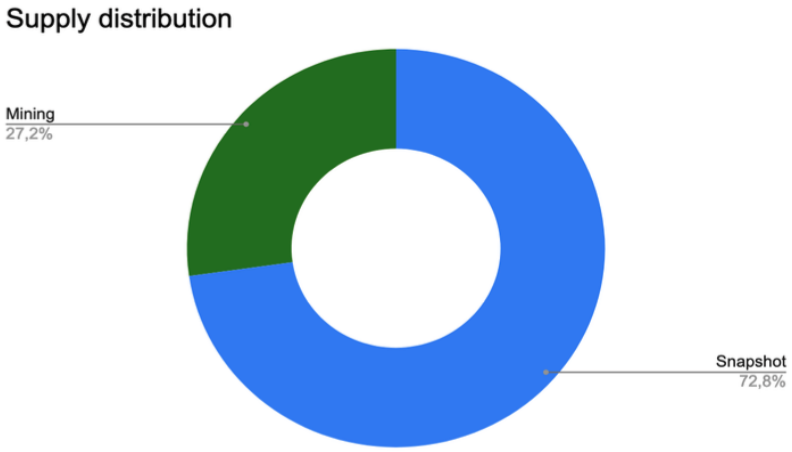
[7] https://en.bitcoin.it/wiki/Controlled_supply

```cpp
#include <iostream>
#include <cmath>

// Amounts of satoshit per coin
const int64_t COIN = 10000;

int64_t reward(int height) {
  // Initial reward per block
  const int64_t reward = 5500 * COIN;
  // Reward decreasing epoch (2 years)
  const int epoch = 525960 * 2;
  // Decrease amount by 30% each epoch
  const long double r = 1 + (std::log(1 - 0.3) / epoch);
  return reward * std::pow(r, height);
}
```

총 공급량은 610억 MBC로 한정되며, 이 중 44,386,397,362.4252 MBC는 이전 네트워크에서 스냅샷을 통해 활성화된 양이다. 나머지 16,613,602,638 MBC는 약 100년 동안 채굴될 것이다.



6. 블록 크기 (Block size)

네트워크의 안정성을 높이기 위해 블록 스패밍(block spamming)을 방지하고 분당 1블록의 관점에서 공정하고 매력적인 수수료 시장을 만들기 위해 블록 크기를 300kb로 줄였다. 이것의 구현은 비트코인 코어 개발자 Luke Dashjr 제안[8]에서 영감을 얻었다.

[8] https://github.com/bitcoin/bitcoin/compare/v0.17.1...luke-jr:example_300k-0.17

7. Power2B 작업 증명 알고리즘 (Power2B Proof-of-Work algorithm)

나카모토 사토시가 비트코인 원본 백서에서 제안한 "1-CPU-1-Vote"[9] 아이디어와 탈중앙화의 가치를 살리기 위해 CPU 친화적이고 반-GPU적인 FPGA 및 ASIC-nuetral로 설계된 YesPower[10] 해시 함수를 수정하여 만든 파워2B(Power2B)[11]를 사용했다. 이것은 계산 속도가 빠르고, 순차적 메모리 하드 해싱(sequential memory-hard hashing)을 통해 GPU 속도를 CPU와 같은 속도로 낮추고 FPGA 및 ASIC의 잠재적 이점을 제한하는 문제를 해결한다. 지금까지 YesPower는 수십 개의 다양한 암호화폐에 보안을 제공함으로써 효과적인 CPU 중심의 알고리즘임이 입증되었다.

우리의 Power2B 수정은 SHA256 기반의 PBKDF2 및 HMAC를 Blake2B[12] 기반으로 대체하여 YesPower의 독창적인 디자인을 그대로 유지한다. 이것은 원래 YesPower 용 FPGA와 ASIC을 Power2B와 호환되지 않도록 구현하기 위해서 수행했다. 이를 위해서는 개발자들이 별도의 마이크로비트코인 소프트웨어/하드웨어를 구현하여야 하며, 결과적으로 네트워크 보안을 전반적으로 강화해야 한다.

8. 난이도 조정 알고리즘 (Difficulty adjustment algorithm)

마이크로비트코인 네트워크는 Zawy12가 작성한 LWMA3[13] 난이도 조정 알고리즘을 사용한다. 가장 최근의 난이도와 계산 시간을 추정하여 현재 해시레이트 난이도를 설정한다. 평균 난이도를 계산 시간의 선형 가중 이동 평균(LWMA)으로 나눈다. 이것은 가장 최근의 해결 시간에 더 비중을 둔다. 타임스탬프 조작과 해시 공격으로부터 작은 규모의 코인을 보호하기 위해 설계되었다. 기본 공식은 다음과 같다:

next_difficulty = average(Difficulties) * target_solvetime / LWMA(solvetimes)

9. 다른 Bitcoin 하드포크와의 비교

---

[9] https://bitcoin.org/bitcoin.pdf

[10] https://www.openwall.com/yespower/

[11] https://github.com/MicroBitcoinOrg/Power2B

[12] https://blake2.net

[13] https://github.com/zawy12/difficulty-algorithms/issues/

마이크로비트코인을 다른 비트코인 하드포크와 비교한 표이다.

	Bitcoin	BitcoinCash	BitcoinGold	MicroBitcoin
Total Supply	21M	21M	21M	61B *
PoW Algorithm	SHA256	SHA256	Equihash	Power2B
Mining Hardware	ASIC	ASIC	CPU/GPU	CPU
Block Creation Interval	10min	10min	10min	1min
Difficulty Adjustment	Bitcoin DAA	SMA	LWMA2	LWMA3
SegWit	Yes	No	Yes	Yes
Block Size	1mb	32mb	1mb	300kb

비트코인인 소숫점 8자리인 반면 마이크로비트코인은 소숫점 4 자리를 갖는다. 따라서 Satoshi 단위로 볼 때 마이크로비트코인의 공급량은 비트코인보다 3 배 더 크다.

10. 토큰 레이어(Token Layer)

블록체인 기술은 그 본래의 설계로 인해 새로운 기능을 추가할 때 합의를 유지하기 위해서는 모든 네트워크 피어가 새로운 규칙 집합인 하드포크에 합의해야 한다. 마이크로비트코인을 위한 토큰 레이어(Token Layer)를 구축하는 동안 하드포크에 대한 이전의 경험을 바탕으로 '블록체인 데이터 임베딩(blockchain data embedding) 기반의 하위 네트워크'라는 또 다른 접근 방식을 사용하기로 결정했다. 이는 기존 컨센서스에 하드포킹이나 새로운 획기적인 변화를 도입하지 않고도 기존 생태계 내에서 토큰과 같은 새로운 기능을 도입할 수 있는 완벽한 방법이다.

10-1. 개요

네트워크가 성장함에 따라 새로운 기능과 개선에 대한 아이디어가 등장할 것이다. 이것은 일반적으로 기본 합의 규칙을 수정하고 네트워크 피어의 대다수가 소프트웨어를 업데이트해야 하기 때문에 네트워크 구성원들에게 유지보수에 시간과 노력을 투자하게 하고 불편을 초래한다. 이런 이유로 커뮤니티는 새롭게 제안된 변경사항들을 받아들이지 않을 수 있으며, 이는 네트워크 분할을 초래하고 커뮤니티를 와해시키기도 한다.

소프트 포크는 이 문제에 대한 하나의 해결책이며, 비트코인 개발자들에 의해 도입되었다. 그 요지는 합의에 새로운 규칙을 추가하는 것인데, 예를 들어 일정 비율의 네트워크가 새로운 규칙을 수락한 후에 블록 보상을 제한함으로써 이전에 유효했던 블록을 무효로 만드는 것이다. 프리 소프트 포크 소프트웨어는 여전히 합의의 일부이기 때문에 업데이트 노드에 의해 생성된 블록을 처리할 수 있지만, 새로운 노드는 새로 수립되고 합의된 규칙을 깨기 때문에 이전 소프트웨어에 의해 생성된 블록을 받아들이지 않을 것이다. 다른 측면에서 하드 포크는 예를 들어 네트워크에 토큰 기능을 추가하고 본질적으로 고유의 구조[14]를 가진 새로운 종류의 거래를 도입함으로써 이전에 유효하지 않았던 블록이 유효하게 되는 방식으로 규칙을 수정하는 것이다.

레이어 2 네트워크의 출현은 기본 블록체인 네트워크에 부과되는 제약과 소프트 포크와 하드 포크 모두에 의해 야기되는 장애로부터 비롯되었다. 이들은 서로 다른 규칙에 의해 통제되고 합의에 의해 통제되는 별개의 네트워크에서 새로운 특징을 도입하며, 이는 기본 네트워크와 독립적으로 작동한다. 이러한 접근 방식의 좋은 예는 비트코인 블록체인 위에 구축된 라이트닝 네트워크[15]이다. 이 네트워크는 오프-체인에서 즉각적인 결제를 용이하게 하고 기본 네트워크를 사용하여 거래를 제공하고 결제 채널을 폐쇄함으로써 해당 결제를 완료한다. Omni Layer[16] 역시 합의를 깨지 않고 기본 네트워크에 새로운 기능을 도입하는 또 다른 좋은 예이다. 자체 페이로드(payload)를 OP_RETURN 출력에 내장하여, 누구나 비트코인 블록체인을 거치며 인코딩된 페이로드(payload)를 처리하여 Omni Layer의 현재 상태를 재구축할 수 있다.

10-2. 디자인(Design)

우리는 새로운 기능을 도입했던 이전의 접근 방식을 고려하여 OP_RETURN 출력을 통한 블록체인 데이터 임베딩을 토큰 레이어(Token Layer)의 기반으로 사용하기로 결정했다. OP_RETURN opcode는 트랜잭션에 추가 데이터를 첨부하는 표준 방식으로, 다음[17]과 같이 구성된 script PubKey로 0-값 출력을 추가하는 것이다.

---

[14] https://petertodd.org/2016/forced-soft-forks

[15] https://lightning.network/

[16] https://www.omnilayer.org/

[17] https://en.bitcoin.it/wiki/Script

이 방식은 단일 트랜잭션에서 최대 83바이트의 인코딩된 페이로드(payload)를 첨부할 수 있다. 우리는 트랜잭션당 스토리지 용량이 다소 제한적이기 때문에 페이로드(payload) 인코딩을 위해 데이터를 직렬화하고 전송하기 위한 콤팩트하고 효율적인 방법을 제공하는 메시지 팩(Message Pack)[18]을 사용했다.

프로토콜은 마이크로비트코인 네트워크를 통해 데이터가 포맷되고 전송되고 처리되는 방법을 정의하는 규칙의 집합이다. 이를 통해 모든 토큰 레이어의 클라이언트가 교환하는 정보를 이해하고 해석할 수 있어 이들 간의 원활하고 표준화된 통신이 가능하다.

토큰 레이어는 마이크로비트코인 블록체인을 스캔하여 작동한다. 이 과정에서 클라이언트는 각각의 블록을 거치며 트랜잭션에서 OP_RETURN 출력을 찾는다. 이러한 출력이 발견되면 토큰 레이어 클라이언트는 첫 번째 바이트를 확인하고 서로 다른 하위 네트워크를 구분하기 위해 사용되는 고유 식별자인 현재 체인 ID와 비교한다. 체인 ID 바이트 뒤에 메시지 팩(Message Pack)을 사용하여 인코딩된 프로토콜 페이로드(protocol payload)가 이어진다.

토큰 레이어는 비트코인-스타일의 사토시를 사용하여 토큰 생성자가 지정한 0에서 8 사이의 모든 금액을 표현할 수 있다. 예를 들어 소숫점 2자리의 10,000 TEST 토큰은 1,000,000 사토시로 표현된다. 메시지 팩(Message Pack)에 사토시 값을 저장하는 가장 쉽고 명확한 데이터 유형은 이진법으로, 값 필드에 필요한 공간을 지정할 수 있기 때문에 바이트 배열[19]을 나타낸다. 우리는 정수 값을 빅 엔디언 10 바이트(big-endian 10-byte) 배열로 변환할 것인데, 이는 토큰 레이어 대부분의 요구를 감당하기에 충분하다.

토큰 티커(token ticker)는 대문자, 라틴 문자, 숫자 및 .(콤마)와 -(하이픈) 기호만 사용할 수 있다. 토큰 티커의 길이는 3-32자 범위 내에 있어야 한다. 또한 다른 유사한 솔루션에 존재하는 몇 가지 다른 문제를 해결한다. 예를 들어, 루트(root), 서브(sub), 유니크 및 소유자(unique and owner)와 같은 다른 토큰 유형을 나타내는데 사용될 수 있다.

루트 토큰(Root token)은 기본 토큰 유형으로 제한 없이 사용할 수 있다. 재발급 필드(reissuable field)를 true Token Layer로 설정한 루트 토큰이 생성되면 자동으로 owner Token이 생성되며, 이는 ! 기호로 표시된다.

---

[18] https://msgpack.org/index.html

[19] https://github.com/msgpack/msgpack/blob/master/spec.md#type-system

at the end of ticker: TEST (root) and TEST! (owner).

오너 토큰(Owner token)은 루트 토큰에 대한 소유권을 나타낼 뿐만 아니라 루트 이름 위에 추가적인 공급 및 발행, 서브/유니크 토큰의 생성 등을 승인하는 데 사용된다. 서브 토큰을 발행할 때에도 동일한 규칙이 적용된다.

서브 토큰(Sub token)은 그 소유자가 기존 루트 토큰 위에서만 발행될 수 있으며 / 로 표시된다. 예를 들면 'COMPANY' 토큰을 발행하고 해당 토큰의 진위를 증명할 수 있는 COMPANY/STOCK를 발행할 수 있다.

유니크 토큰(unique token)은 대체 불가능한 것들을 나타내기 위해 사용되며 #기호로 표시되는 소수점 없이 1단위로만 생성될 수 있다. 하위 토큰뿐만 아니라 그 위에 루트 토큰(Root token)을 발행해야 한다. 예를 들면 'COLLECTION' 루트 토큰과 그 위에 'COLLECTION#ART'를 발행하는 방식이다.

마지막으로 이러한 복잡한 시스템을 관리하기 위해 토큰 레이어(Token Layer)는 거버넌스 체계를 갖추고 있다. 토큰 생성 또는 발행 비용을 금지, 금지 해제, 변경 등 특수한 형태의 거래를 발행할 수 있고 토큰 발행으로 자금을 받을 수수료 주소를 업데이트할 수 있는 관리자 주소를 보유하고 있다. 관리자 주소는 네트워크 시작 시 설정되며 하드포크를 통해서만 업데이트가 가능하다. 이 시스템은 해킹이나 기초 통화 가격 변동의 변화와 같은 사건을 신속하게 완화할 수 있다.

10-3. 프로토콜(Protocol)

각각의 프로토콜 메시지는 버전(version) 및 카테고리(category)의 두 개의 불변의 필드를 갖는다. 버전 필드는 토큰 레이어 프로토콜에 대한 수정이 필요한 경우 사용된다. 카테고리 필드는 각각의 주어진 메시지가 하는 일, 예를 들어 토큰 발행, 전송, 소각 등을 정의한다. 메시지 페이로드(message payload)는 카테고리에 따라 토큰 메타데이터 또는 전송량과 같은 추가 필드를 포함할 수 있다. 데이터 구조가 주어진 카테고리에 대한 모든 요건과 일치하지 않으면 메시지는 삭제된다.

토큰 메시지 만들기(Create token message)

이 메시지는 새로운 토큰을 생성하는 역할을 한다. 생성 비용을 지불하기 위해서는 스팸 및 토큰 티커 불법 점유(token ticker squatting)를 방지하기 위해 거버넌스 주소 추가 출력이 필요하다. 각각의 토큰 티커(token ticker)는 Unique하며 한 번만 사용할 수 있다. 토큰에는 'true'로 설정하면 추가 오너 토큰(owner token)을 생성하는 재발급 필드가 있다. 예를 들어, 사용자가 'TEST' 토큰을 생성하면 오너 토큰

(owner token)도 받을 수 있다.

`!TEST`.

Category fields:

- `reissuable`: bool - defines whether owner can increase supply of this token
- `value`: bytes - token supply value encoded in bytes
- `decimals`: int - token decimal places
- `ticker`: str - token ticker

Example raw data:

```
{
    "v": b"\x00\x00\x00\x00\x00\x00\x0fB@", # Value - 1000000 satoshis
    "r": False, # Reissuable - false
    "t": "TEST", # Ticker - TEST
    "d": 2, # Decimals - 2
    "c": 1, # Category - create
    "m": 1, # Version - 1
}
```

Example encoded message:
`86a176c40a000000000000000f4240a172c2a174a454455354a16402a16301a16d01`

토큰 메시지 발행(Issue token message)

메시지는 토큰 공급을 증가시키는 역할을 한다. 토큰 재발급이 'true'로 설정되어 있고 사용자가 오너 토큰(owner token)을 가지고 있는 경우에만 사용할 수 있다. 토큰 메시지 발행은 생성 수수료를 지불하기 위해 거버넌스 주소에 대한 추가적인 출력이 필요하다.

Category fields:

- `value`: bytes - token additional supply value encoded in bytes
- `ticker`: str - token ticker

Example raw data:

```
{
    "v": b"\x00\x00\x00\x00\x00\x00\x0fB@", # Value - 1000000 satoshis
    "t": "TEST", # Ticker - TEST
    "c": 2, # Category - issue
    "m": 1, # Version - 1
}
```

xample encoded message:
`4a176c40a000000000000000f4240a174a454455354a16302a16d01`

토큰 메시지 전송(Transfer token message)

주소 잔액 간에 토큰을 전송하는 역할을 한다. 적은 마커 양으로 수신자 주소로 추

가 출력을 요구하므로 토큰 레이어 클라이언트가 전송 수신자를 식별하는 데 도움이 된다. 잠금 필드를 'int' 값으로 설정하면 수신자는 지정된 높이까지 자신이 받은 토큰을 사용할 수 없다.

Category fields:

- `lock`: int or null - optional block height until which transfer will be locked and unspendable
- `value`: bytes - token transfer value encoded in bytes
- `ticker`: str - token ticker

Example raw data:

```
{
    "v": b"\x00\x00\x00\x00\x00\x00\x00\x0fB@", # Value - 1000000 satoshis
    "t": "TEST", # Ticker - TEST
    "l": 100, # Lock - block #100
    "c": 3, # Category - transfer
    "m": 1, # Version - 1
}
```

Example encoded message:
`85a176c40a000000000000000f4240a174a454455354a16c64a16303a16d01`

메시지 소각(Burn message)

이 메시지는 토큰을 소각하는 역할을 한다. 추가적인 출력 없이 토큰 보유자라면 누구나 수행할 수 있다.

Category fields:

- `value`: bytes - token burn value encoded in bytes
- `ticker`: str - token ticker

Example raw data:

```
{
    "v": b"\x00\x00\x00\x00\x00\x00\x00\x0fB@", # Value - 1000000 satoshis
    "t": "TEST", # Ticker - TEST
    "c": 6, # Category - burn
    "m": 1, # Version - 1
}
```

Example encoded message:
`84a176c40a000000000000000f4240a174a454455354a16306a16d01`

비용 메시지(Cost message)

이 관리자 메시지는 토큰 생성/발급의 마이크로비트코인 비용 변경 역할을 한다.

토큰 레이어 거버넌스 주소에서만 사용할 수 있다.

Category fields:

- `value`: bytes - MBC cost value encoded in bytes
- `category`: str - cost category (create/issue)
- `type`: str - token type (root/sub/unique)

Example raw data:

```
{
    "v": b"\x00\x00\x00\x00\x00\x00\x0fB@", # Value - 1000000 MBC satoshis
    "a": "create", # Action - create
    "t": "root", # Type - root
    "c": 8, # Category - cost
    "m": 1, # Version - 1
}
```

Example encoded message:

`85a176c40a00000000000000f4240a161a6637265617465a174a4726f6f74a16308a16d01`

금지 메시지(Ban message)

이 관리자 메시지는 지정된 주소에서 토큰 잔액을 금지하는 역할을 한다. 적은 마커 양으로 금지 주소에 추가 출력을 요구하므로 토큰 레이어 클라이언트가 금지할 주소를 식별할 수 있다. 추가 필드는 필요하지 않다.

Example raw data:

```
{
    "c": 4, # Category - ban
    "m": 1, # Version - 1
}
```

Example encoded message:

`82a16304a16d01`

금지 해제 메시지(Unban message)

이 관리자 메시지는 지정된 주소에서 토큰 잔액을 금지 해제하는 역할을 한다. 적은 마커 양으로 금지 해제 주소에 대한 추가 출력을 요구하므로 토큰 레이어 클라이언트가 금지 해제할 주소를 식별하는 데 도움이 될 것이다. 추가 필드는 필요하지 않다.

Example raw data:

```
{
    "c": 5, # Category - unban
    "m": 1, # Version - 1
}
```

Example encoded message:
`82a16305a16d01`

수수료 주소 메시지(Fee address message)

이 관리자 메시지는 토큰 생성 및 발행에 대한 마이크로비트코인을 지불받을 수 있는 수수료 주소를 업데이트하는 역할을 한다. 이는 적은 마커 금액으로 새로운 수수료 주소에 대한 추가 출력을 요구하므로 토큰 레이어 고객들이 새로운 수수료 주소를 식별하는 데 도움이 될 것이다. 추가적인 필드는 필요하지 않다.

Example raw data:

```
{
    "c": 7, # Category - fee address
    "m": 1, # Version - 1
}
```

Example encoded message:
`82a16307a16d01`

References

[1] https://en.bitcoin.it/wiki/ASIC

[2] https://www.groestl.info

[3] https://bitcointalk.org/index.php?topic=5057818.0

[4] https://www.slideshare.net/bschn2/the-rainforest-algorithm

[5] https://www.investopedia.com/terms/u/utxo.asp

[6] https://en.bitcoin.it/wiki/Genesis_block

[7] https://en.bitcoin.it/wiki/Controlled_supply

[8]https://github.com/bitcoin/bitcoin/compare/v0.17.1...luke-jr:example_300k-0.17

[9] https://bitcoin.org/bitcoin.pdf

[10] https://github.com/MicroBitcoinOrg/Power2B

[11] https://www.openwall.com/yespower/

[12] https://blake2.net/

[13] https://github.com/zawy12/difficulty-algorithms/issues/3

[14] https://en.bitcoin.it/wiki/Satoshi_(unit)

[15] https://petertodd.org/2016/forced-soft-forks

[16] https://lightning.network/

[17] https://www.omnilayer.org/

[18] https://en.bitcoin.it/wiki/Script

[19] https://msgpack.org/index.html

[20] https://github.com/msgpack/msgpack/blob/master/spec.md#type-system

Links

Official Website: https://microbitcoin.org

GitHub: https://github.com/MicroBitcoinOrg/

Explorer: https://microbitcoinorg.github.io/explorer/#/

Web Wallet: https://microbitcoinorg.github.io/wallet/#/

API: https://api.mbc.wiki/

Discord: https://discord.gg/8zg2nTV

Telegram: https://t.me/microbitcoinorg

Twitter: https://twitter.com/MicroBitcoinOrg

Forum: https://mbc.wiki

BitcoinTalk: https://bitcointalk.org/index.php?topic=3982489.msg37769108

Reddit: https://www.reddit.com/r/MicroBitcoinOrg/

Token Layer: https://github.com/MicroBitcoinOrg/Tokens