자체 블록체인 네트워크 구축을 위한 블록체인 코어 프레임워크 구현

The Implement of Blockchain Core Framework for Construct Own Blockchain Network

요 약

블록체인 기술은 일종의 분산 원장 데이터베이스로써, 중앙 서버가 존재하지 않고, 모든 노드가 Peer로 참여하는 퓨어 P2P 방식의 네트워크이다. 모든 네트워크 참여자들이 Consensus 에 참여하여 원장의 위·변조를 사전에 방지하고, 네트워크의 신뢰성을 유지할 수 있기 때문에 Decentralized Service라는 특성을 가지고 있다.

이러한 이유 때문에, Decentralize를 필요로 하는 서비스에서 블록체인을 접목시키려는 시도가 늘고 있다. 하지만 서비스와 잘 호환될 수 있는 자체 블록체인 코어를 Zero-Base에서 설계 및 구축하는 것은 많은 시간이 소요되기 때문에 대다수의 업체는 호환성, 속도 등의 다양한 문제점을 감안하고, 이미 존재하는 블록체인 프로젝트를 Fork하거나 이더리움, EOS와 같은 플랫폼을 이용한 Dapp을 구축한다.

본 논문에서는 자체 블록체인 코어를 보다 쉽고 효율적으로 구현 할 수 있도록 하는 블록체인 코어 프레임워크를 오픈 소스를 이용하여 구현 및 개발한 내용을 다룬다.

1. 서 론

Decentralize를 필요로 하는 다양한 서비스에서 블록체인을 이용하려는 시도가 늘어나고 실제 제공되는 서비스도 빠른 속도로 증가하고 있다. 하지만 자신이 제공하고자 하는 서비스와 잘 호환될 수 있는 자체 블록체인 코어를 Zero-Base에서 설계 및 구축하는 것은 많은 시간과 노력이 필요하기 때문에 대다수의 업체는 호환성, 속도 등의 다양한 문제점을 감안하고, Bitcoin과 같은 이미 존재하는 블록체인 프로젝트를 Fork하여 사용하거나 별도의 코어가 필요한 서비스라 할지라도 Ethereum, EOS와 같은 플랫폼을 이용한 Dapp을 구축하는 경우가 많다.

타 블록체인 프로젝트를 사용하면 수많은 문제가 발생하는데도 불구하고 자체 코어를 개발하지 않는 데에는 여러가지 이유가 있다. 가장 일반적인 코어 개발 방식은 기존의 블록체인 코드를 Fork한 후 구현하는 것이다. 하지만, 기존의 블록체인 프로젝트는 대다수가 하드코딩 되어 있기 때문에 코드 분석이 어렵고 블록체인 코어의 특성상 하드코딩되어 있는 대부분이 해싱 값이기 때문에 더욱 코드 분석의 어려움이 가중된다.

또한 기존 블록체인 프로젝트들은 자체 블록체인 프로젝트에 특화된 코드설계를 갖고 있다는 점이다. 즉 오픈소스로서의 범용성과 활용성이 떨어진다. 이러한 프로젝트를 Fork하여 사용할 경우 간단한 코드 수정에도 수많은 Code Dependency를 체크해야 하는 문제가 발생하게 된다.

뿐만 아니라 기존 블록체인 프로젝트는 익명성 프로토콜이나 자체 프로젝트에 종속적인 기능이 운영 및 유지 보수 과정에서 추가 구현된다. 하지만 Fork를 통해 자체 블록체인 네트워크를 구축하고자 할 경우 기존 블록체인의 추가 구현된 내용의 대다수가 불필요하다. 이러한 불필요한 기능들은 코드 분석을 어렵게 만들고 Dependency문제를 가중시킨다. 또한 필요하지 않은 기능은 성능의 저하를 야기하게 된다.

위와 같은 문제들로 인해 블록체인 네트워크를 분리 시키는 것은 많은 비용이 요구된다. 본 논문에서는 이러한 기존의 문제를 해결하기 위한 프레임워크를 구현 및 개발한 내용을 다룬다.

2. 관련 연구

2.1 Bitcoin

Blockchain Technology는 2008년 Satoshi Nakamoto의 논문 Bitcoin : A Peer to Peer Electronic Cash System[1]에 최초 기술되었으며 Bitcoin은 2009년 개발된 최초의 블록체인 기술을 이용한 Decentralized 암호화폐이다.

Open Source로 공개되어 있지만 대다수의 상수와 변수 값이 하드코딩되어 있고 코드를 변경하였을 경우 Assert에 의해 컴파일시에 에러가 발생하는 등 Open Source의 특성인 범용성과 확장성을 갖추고 있지 않다.

2.2 Litecoin

Litecoin은 MIT의 Charlie Lee가 개발한 비트코인 포크 기반의 암호화폐로 해싱방식을 SHA-256에서 Scrypt로 바꾸고,블록체인 블록사이즈를 4MB로 늘렸으며 최대 발행 수량을 늘린 블록체인 프로젝트이다. 자체 블록체인 서비스 구현을 위해 비트코인의 Assert 문장과 비트코인 종속적인 코드를 모두 제거하여 비교적 코드 수정이 간단하다.

Open Source로 공개되어 있고, 비트코인 종속성을 제거하였기 때문에 코드 수정이 비교적 간편해 가장 많은 블록체인 프로젝트의 모체가 되었다.

비록 Assert문을 제거하여 최대 발행량, 네트워크 주소, 해싱 방식, 블록사이즈, 블록 생성 시간등의 하드코딩된 블록체인 인자를 바꿀 수는 있게 되었지만, 큰 틀에서의 변화를 위해서는 기존 코드를 새롭게 고치고 Dependency를 체크하며 수정해야 한다는 점에서 여전히 Bitcoin Fork 방식의 한계점이 존재한다.

2.3 Peercoin

Peercoin[2]은 Bitcoin을 모체로 하며 기존 Bitcoin의 합의 알고리즘인 작업증명(proof-of-work)을 지분증명(proof-of-

stake)으로 변경하였다. 지분증명 방식 기반의 블록체인 프로젝트들의 모체가 되고 있으며 Bitcoin Fork 프로젝트 중 최초로 가장 많은 부분에서 변화가 나타난 프로젝트이다.

Bitcoin에서 합의 알고리즘을 Dependency를 모두 고려하면서 Bitcoin의 코드를 수정하여 구현하였지만 결국 Bitcoin이 아닐 뿐, Peercoin에 종속되어있기 때문에 Bitcoin의 문제점으로 언급했던 오픈소스로서의 확장성과 범용성 문제를 여전히 해결하지 못했다.

2.4 Bytecoin

Bytecoin[3]은 CryptoNote technology 기반의 최초 익명성 코인으로 cryptonight 알고리즘을 사용하였으며 Ring signature을 이용하여 익명성과 정보 보호를 강조하며 등장하였다. Bitcoin과 Bitcoin Forked 프로젝트와 전혀 관련이 없는 최초의 독립 블록체인이기도 하다. CryptoNote라고 하는 자체 블록체인 코어 기술을 통해 만들어졌으며, 기존 하드코딩방식이 아닌 확장성과 범용성을 고려한 프로젝트다. 비교적 쉽게 인자를 수정할 수 있지만 소스코드 길이가 비트코인보다 훨씬 방대하며, 자체 블록체인 네트워크를 분리해내는 것이 어렵다는 단점이 있다.

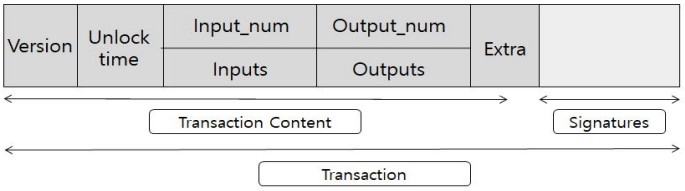
2.5 Ethereum

Ethereum[4]은 2015년 Vitalik Buterin이 개발한 Blockchain based Decentralized Application을 개발할 수 있는 플랫폼이다.

Bitcoin을 Fork하는 방식이 아닌 Go 언어를 이용하여 직접 블록체인 코어를 비롯한 모든 기능을 직접 구현하였으며 Bitcoin의 화폐로서의 기능에만 적용가능한 단점을 극복하여 거래나 결제뿐 아니라 다양한 Application을 운영할 수 있는 확장성을 제공한다. 하지만 비교적 개발자들에게 친숙하지 않은 Go언어를 통해 작성되어 접근성이 떨어지고, 플랫폼으로서의 기능을 하기 때문에 자체 블록체인 구현을 위해 사용하기에는 불필요한 기능이 과도하게 구현되어 있어 소스코드 자체가 방대하고 분석이 어려울 뿐더러, 불필요한 기능을 분리해내기 어렵다는 특징이 있어 자체 블록체인을 구현하는데에 사용되지는 않는다.

3. 구현 내용

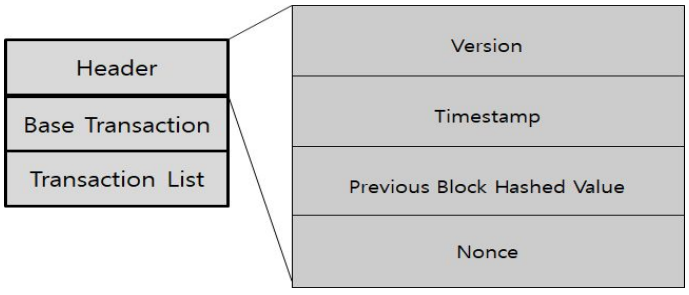
3.1 Transaction Structure



<그림 1>

블록체인의 트랜잭션 구조는 **<그림 1>**과 같으며, 필요에 따라서 타입이나 용도는 이용자가 수정할 수 있도록 구현하였다. UnlockTime에는 UNIX 기반의 Timestamp를 기록하며, Inputs에는 트랜잭션 소유권에 대한 키 정보와 Block Height가, Outputs에는 전송될 코인의 수량과 목적 주소정보가 저장된다.

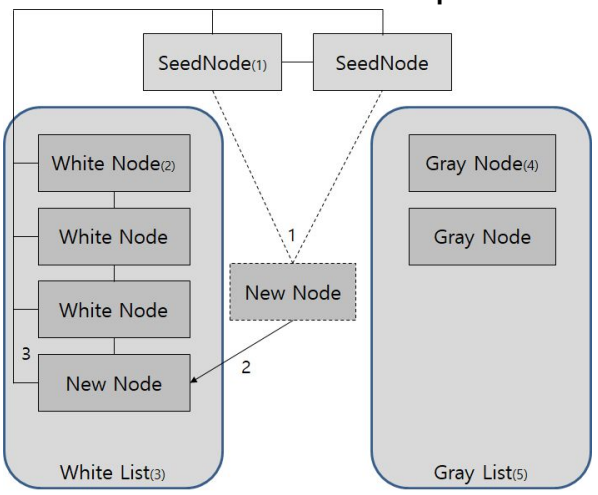
3.2 Block Structure



<그림 2>

블록은 상기 **<그림 2>**와 같이 블록 헤더 + Base Transaction + Transaction List로 구성하였다. 블록헤더는 Version과 UNIX 기반의 Timestamp,이전 블록의 해쉬값, POW를 위한 Nonce로 구성되어 있으며 Base Transaction은 해당 블록에 생성된 최초의 트랜잭션, 즉 Block Creation Transaction 정보를 저장한다. Transaction List에는 해당 블록에 포함된 트랜잭션들의 TxHash 값이 리스트형태로 저장된다.

3.3 P2P Network Structure and Development



<그림 3>

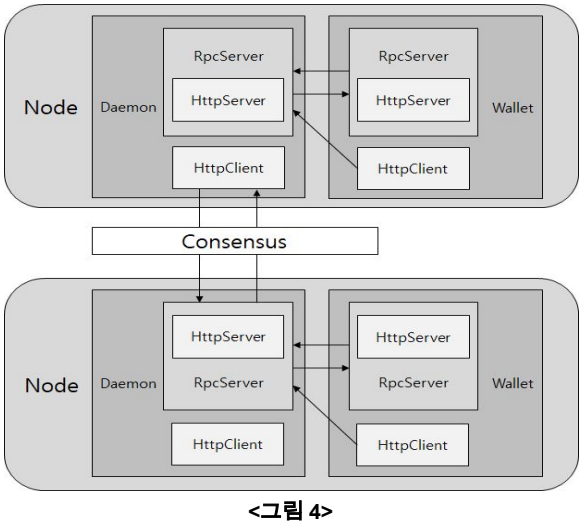
<그림 3>은 본 프레임워크의 P2P 네트워크 개발 구조이다.

(1)SeedNode로 네트워크의 첫번째 노드이며 다른 노드들은 처음 연결시 해당 노드에 먼저 연결되어 현재 네트워크에 참여하고 있는 노드들의 항목을 받아 항목의 노드들과 연결된다. (2)White Node는 현재 네트워크에 참여 P2P연결되어 있는 노드를 의미하며 (3)White List는 White Node들의 집합이며 새로운 노드가 연결될 때 SeedNode에서 전달되는 요소이다. (4)Gray Node는 Blockchain Network에 참여하였지만 현재는 연결이 종료된 노드를 의미하며 각 Gray Node는 종료 당시의 White List를 갖고 있다. (5) Gray List는 연결이 종료된 노드인 Gray Node들의 집합이다.

새로운 노드가 Blockchain Network에 참여하고자 할 때 1번 모든 SeedNode와 연결되어 현재 참여자 노드 항목인 White List를 받아온다. 다음으로 2번 White List에 새로운 노드가 추가된다. 마지막 3번 White List에 있는 모든 노드들과 연결된다

이러한 방식을 통해, SeedNode의 IP값만 지정해준다면 손쉽게 분리된 블록체인 네트워크를 구성할 수 있다.

3.4 Blockchain Network Model



<그림 4>

<그림 4>는 본 블록체인 프레임워크 P2P노드의 구조와 통신

방식 모델이다.

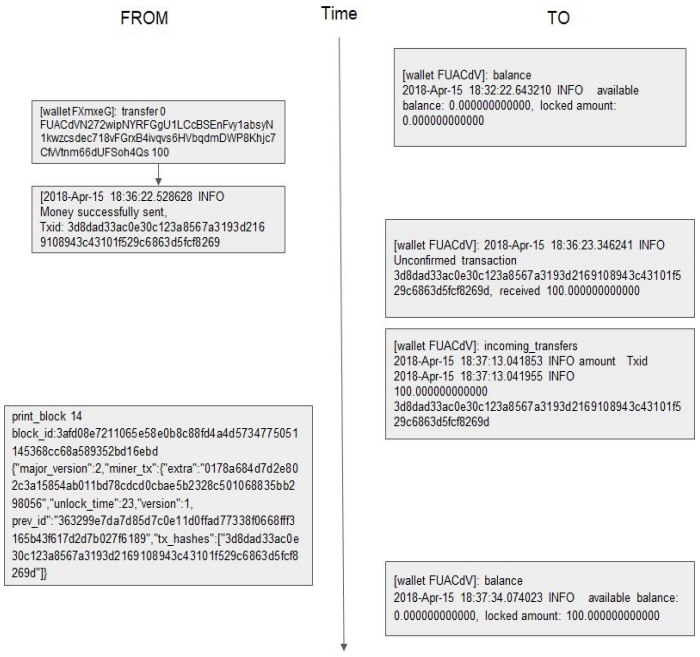
노드 내에서는 Daemon과 Wallet이 동작하고 있으며 통신을 위해서 RPCServer와 HttpServer가 실행된다. RPCServer는 HttpServer를 동작시켜 외부 노드 및 프로세스와 통신을 진행한다. RPC method는 오픈소스로 공개되어 있는 Bytecoin의 JSON_API[5][6]를 이용하여 구현하였다. RPCServer는 JSON RPC 방식으로 통신하게끔 구현하였다. Wallet과 Daemon은 RPCServer와 HttpClient간의 통신을 하게 되는데, 요청측에서 JSON방식으로 Wrapping된 Signal과 Method 실행 요청을 전달하고 그에 따른 state 또는 반환 JSON 값을 리턴하는 방식으로 구현되었다.

다음으로 Consensus는 작업증명(Proof of Work)를 사용한다. 작업증명 방식은 컴퓨터의 연산력을 바탕으로 합의를 진행하는 방식으로 새로운 블록을 블록체인 네트워크에 추가하기 위해서는 Difficulty에 따른 Target 규격을 만족시키는 nonce값과 그 해시값을 구해야 한다. 본 프레임워크에서는 임의의 nonce값을 해시함수에 넣고 Difficulty에 따른 숫자보다 작은 값인지 확인한다. 만약 Difficulty에 따른 인자 값보다 작은 값의 Nonce를 구하게 된다면 블록을 생성하고 채굴 보상을 받고, 아니라면 또다른 임의의 Nonce값을 대입하며 블록해시 값을 찾도록 구현되었다.

마지막으로 노드들 사이의 통신으로는 동기화과정, Consensus 에 의한 Mining Block submit 등에 사용되는데 이는 Daemon에 의해 실행된 HttpClient가 요청을 보내오면 HttpServer에서 요청을 수신하고 RPCServer 에서 요청에 해당하는 기능을 수행한다. 수행한 결과는 다시 요청 받은 노드의 HttpServer에 JSON형태로 전달되며 전달받은 JSON이 요청한 HttpClient에 전달되게 된다.

3.5 Own Blockchain Implementation By Using this Framework.

3.5.1 Transfer Process



<그림 5>

<그림 5>는 실제 본 논문에서 개발한 Framework를 이용하여 구현된 자체 블록체인에서 자체 Coin을 Transfer하는 프로세스를 나타내고 있다.
처음엔 Balance가 0이었던 지갑에 Transaction을 통해 성공적으로 Coin이 송금되어 100코인이 증가한 것을 확인할 수 있다.

3.5.2 Transfer Output

```
{
        'jsonrpc' : '2.0',
        'id' : 'transfer' ,
  'result' : {
    'transaction' : {
        'fee' : 10,
        'extra' : '0178a684d7d2e802c3a15854ab011bd78...',
        'timestamp' : 0,
        'blockIndex' : 13,
        'state' : 0,
        'transactionHash' : '3d8dad33ac0...',
         'amount' : -110,
        'unlockTime' : 0,
        'transfers' :
        {
            'amount' : 100,
            'type' : 0,
            'address' : 'FUACdVN272wipNYR..."
        }
        'paymentId' :,
        'isBase' : False
    }
  }
}
```

<그림 6>

<그림 6>은 실제 Transaction의 Output 결과로 나온 JSON 결과이다. 자체 블록체인에서 성공적으로 RPC를 통해 Transfer가 진행되었고 자체 네트워크가 분리되었다는 것을 확인할 수 있다.

4. 결론 및 향후 연구

관련연구에 기술된 다양한 오픈소스 프로젝트를 이용하여 기존 하드코딩된 방식에서 탈피한 블록체인 코어 프레임워크를 개발 하고 이를 포크함으로서 독립적인 블록체인 네트워크를 손쉽게 구현할 수 있었다.

향후 연구로는 네트워크 분리 뿐만 아니라 전체적인 로직과 컨센서스 수정을 간단하게 하고 코드 자체의 Dependency 문제를 보다 효과적으로 해결하고, 블록체인 코어 자체의 성능을 기존 프로젝트보다 향상시키는 연구를 진행할 것이다.

5. 참고 문헌

[1] Satoshi Nakamoto. "Bitcoin A Peer-to-Peer Electronic Cash System" bitcoin.org, 2009

[2] Sunny King. "PPCoin: Peer-to-Peer Crypto-Currency with Proof-of-Stake", August, 2012

[3] Nicolas van Saberhagen. "CryptoNote v 2.0" (White Paper), October, 2013

[4] Vitalik buterin. "Ethereum White Paper", 2013

[5] Bytecoin. "Bytecoin RPC Wallet JSON RPC API" https://wiki.bytecoin.org/wiki/Bytecoin_RPC_Wallet_JSON_RPC_API, 2012

[6] Bytecoin. "Daemon JSON RPC API" https://wiki.bytecoin.org/wiki/Daemon_JSON_RPC_API , 2012