Owlchain(BOScoin) Technical Specification

Yezune Choi, Jake Hyunduk Choi, Myungsan Jun

0. Adventures of Ideas

마지막 형이상학자로 불렸던 Alfred North Whitehead[1]는 “인간의 과업은 모험하는 것"이라고 단언했다. 블록체인에 대한 새로운 상상력과 실행들이 우리들의 세계를 풍요롭게 만들고 있다. 이 생각의 모험에 동참하기 위해 우리의 생각을 세상에 내 놓는다. 그리고 많은 이들이 우리의 새로운 생각의 모험에 동참하기를 희망한다.

모든 생각들은 그 생각의 기반이 되는 생각의 뿌리들이 있다. 블록체인 역시 인터넷 상에 기반없이 떠 있는 부유물이 아니라 기존의 생각들 위에 뿌리를 내리고 있는 결과물이다. 우리의 블록체인 프로젝트가 인터넷 세상에 확고하게 뿌리를 내리기 위해서는 그 뿌리들을 정확히 알아볼 필요가 있다. 인터넷 그리고 웹이 성장할 수 있었던 생각들은 몇 가지 원칙들로 정리해 볼 수 있겠다.

0.1 레이어 구조

TCP/IP는 인터넷 프로토콜을 대표하지만, TCP는 IP의 대표적인 응용 프로토콜의 하나일 뿐이다. 세션 연결이 필요하지 않은 게임이나 스트리밍 등의 응용의 경우 UDP가 TCP의 매력적인 대안으로 사용되고 있다. 인터넷 세상에서 프로토콜들은 자신의 영역에서 명확한 기능(스펙)을 정의하고 그 기능만을 수행하게 된다. 레이어 구조의 장점은 대체가능성을 높여 새로운 프로토콜이 등장하는 것을 방해하지 않는 것이다. IPv4가 IPv6으로 대체가능한 것도 이러한 레이어 구조 덕을 보고 있는 것이라고 할 수 있다. 이런 관점을 블록체인 기술 발전 과정에 대입해 보자면, 현재의 블록체인 기술은 아직 초기 단계로 레이어 구조 설계 단계에는 접근하지 못하고 있다고 볼 수 있다. 블록체인이 프로토콜의 지위를 획득하려면 다양한 기술들이 결합가능하고 경쟁할 수 있는 레이어 구조 접근이 필수적이다.

0.2 프로토콜의 개방성

TCP/IP가 인터넷의 표준으로 자리잡은 것은 동시대 경쟁자들에 대한 속도, 성능 등 스펙 상의 우위도 있었겠지만 그 경쟁력의 핵심은 개방성이 있었다. TCP/IP 프로토콜은 특허 등의 사용에 대한 제한이 없이 개방되어 많은 연구와 개발이 이루어 질 수 있는 토양으로 작용하였다. 프로토콜의 표준화에서 특정 기술에 대한 특허는, 단순히 그 프로토콜을 사용하는 것을 넘어 프로토콜의 발전에 기여하려는 기업이나 개인들이 참여를 망설이게

[1] 관념의 모험(Adventures of Ideas, 1933년)은 화이트헤드의 저서, 저자 자신이 "가장 쓰고 싶었다"고 말한 문명 비판서로 《과학과 근대 세계》 다음으로 널리 읽혀진다.

만드는 민감한 문제이다. 생각의 소유권(특허)는 새로운 생각들이 피어나는 것을 억제하는 요소로 작용한다. 인터넷 프로토콜을 구현한 프로그래밍 코드들이 오픈소스 라이센스 정책을 취함으로써 그 스스로 지배적인 프로토콜이 됨과 동시에 많은 개선과 혁신이 일어났던 것을 고려해 보면, 인터넷 세상에서 생각의 소유권을 주장하는 것이 얼마나 쓰잘데기 없는 생각인지 알 수 있다.

퍼블릭 블록체인의 경우 오픈소스 정책은 옵션이 아니라 필수 조건이다. 소스코드를 개방을 하지 않는것 자체가 퍼블릭 블록체인으로서 결격사유에 해당한다고 할 수 있다. 다행히 대다수의 퍼블릭 블록체인들이 소스코드를 개방함으로써, 오픈소스를 통한 동반성장 전략을 실행하고 있다. 우리 역시 선배들이 만들어놓은 훌륭한 문화를 따라 오픈소스 전략을 충실히 따를 것이다.

0.3 거버넌스 구조

한가지 아쉬운 것은 비트코인과 같은 성공적인 퍼블릭 블록체인 네트워크조차 커뮤니티의 방향을 결정하는 과정에 최대한의 다수가 참여하는 구조의 거버넌스를 구축하지 못하고 있는 것이다. 많은 기대를 모았던 이더리움 역시 다오(DAO) 사태[2]를 해결하는 과정에서 거버넌스 문제를 해결하지 못했다는 것을 증명하고 말았다. 진정한 프로토콜의 개방성을 획득하려면 초기 블록체인 설계자들이 향후에 참여할 커뮤니티 이해관계자까지 고려한 거버넌스 구조를 고민해야 할 필요가 있다.

0.4 단대단(E2E) 원칙

단대단 원칙은 일반인들에게는 “망중립성의 원칙"으로 더 잘 알려진 것이다. 인터넷 프로토콜 특성(features)이 완성되기 위해서는 상위 응용에 대해 간섭하지 않는 단대단(End-to-end) 원칙이 적용될 필요가 있다. 단대단 원칙이란 프로토콜은 어떠한 경우에도 자신(프로토콜)이 서비스를 제공하는 즉 자신(프로토콜)의 서비스를 통과하는 컨텐츠에 대해 차별하지 않는 원칙이다. 이것은 고속도로에서 요금을 징수하는데 화물의 종류와 관계없이 동일한 요금을 징수하는 것과 동일하다고 볼 수 있겠다. 이 원칙을 블록체인에 적용해 보면, 선발 DApp에 의해 후발 DApp들이 차별 받지 않을 권리가 보장되어야 한다는 것을 의미한다. 다시 이 원칙을 Owlchain의 관점으로 해석하면 구축된 온톨로지(OWL)간 비간섭 원칙을 보장한다는 것을 의미한다. 즉 선행하는 온톨로지가 후행하는 온톨로지에 하위 프로토콜 레이어에 영향력이나 지배력을 행사하지 않는다는 것을 의미한다.

우리는 이러한 4가지 원칙을 중심으로 Owlchain의 프로토콜 특성을 구현하고자 한다.

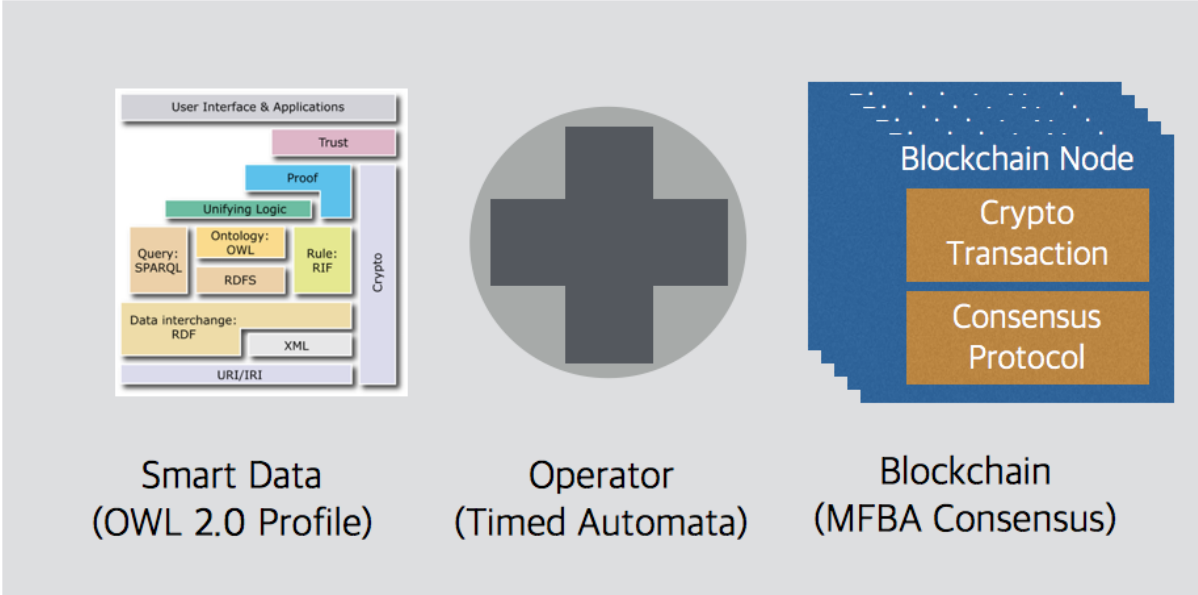
[2] DAO Hack - 이더리움 DApp형태로 개발된 P2P 기반의 펀드 서비스로 개발되었으나 서비스 시작전에 펀딩금액의 1/3이 해킹되는 사태가 발생하였다. 이더리움 DApp 안전성에 의문을 제기하는 대표적인 사건으로 언급되고 있다. 자세한 정보는 다음 사이트를 참고
http://www.coindesk.com/understanding-dao-hack-journalists/

1. Introduction

Owlchain은 Web Ontology Language(OWL)와 Timed Automata Language(TAL)을 적용한 Trust Contract Blockchain이다. W3C의 시맨틱웹 표준인 OWL의 목적은 기존의 웹표준 기술들의 조합을 통해 인터넷 상에서 정보의 신뢰성(trust)을 제공하는 것이다. OWL은 2.0버전에서 기존의 1.0버전에서는 해결하지 못했던 데이터 처리의 시간 지연 문제를 Profile 구분 방식으로 전환하여 상당부분 해결하였다. BOSCoin은 Trust Contract를 작성하기 위해 사용하는 “링크드(linked) 데이터"를 OWL 2 Profile 기반으로 제공한다. 링크드

또한 웹 브라우저에서 화면을 보여주기 위해서 HTML, CSS와 같은 정적인 요소와 자바스크립트를 이용한 동적인 처리방식을 사용하듯, Trust Contract의 작성에도 동적인 처리에 Timed Automata Language(TAL)가 사용된다. TAL은 timed automata modeling과 pure function의 2가지 제약사항이 있는 프로그래밍 환경이다. timed automata modeling을 통해 개발자가 발견하지 못한 프로그램 코드상의 비정의 영역(Undefined Area - reachability problem)를 탐지할 수 있으며 pure function을 사용함으로써 개발 시에 발생할 수 있는 side effect를 제한(get rid of) 할 수 있다. 이런 제약으로 인해 Trust Contract 실행환경은 메모리 안전성과 결정 가능성을 확보할 수 있다.

- Timed Automata - assure decidability
- Pure Function - remove side effect



[Picture 0] Trust Contract Blockchain

OWL언어는 프로그래머 입장에서는 일종의 잘 정의된 링크드 데이터(Linked Data)로 볼 수 있다. 데이터 정의와 제약조건 그리고 데이터의 관계(link) 등이 OWL 언어로 정의되어 있는 상태에서 프로그래밍을 할 수 있기 때문이다. TAL은 블록체인(OWL-chain)의 데이터를 처리하는 언어다. TAL은 블록체인 상에 새로운 트랜잭션을 등록하기 위한 일종의 연산자(operator)로 추상화 될 수 있다. c++ 언어의 간단한 수식으로 설명해보면 “c = a+b;” 수식이 주어 졌을 때, “+” 연산자의 경우 두 변수a, b를 더하고 그 값을 반환하게 된다. Operator 그 자체로는 연산(computing)에 사용되는 두 변수에 어떠한 변화도 만들어 내지 않는 순수함수(pure function)로 구현(implementation)된다. 순수함수는 메모리

안정성(memory safe)을 제공하여 프로그램의 버그를 예방할 수 있게 해준다. 이와 같이 TAL언어로 작성된 오퍼레이터는 처리하려는 링크드 데이터와 블록체인의 정보를 참조하여 새로운 트랜잭션을 반환한다. TAL은 순수함수의 조건과 Timed Automata의 특징을 모두 만족시켜야 한다. 이렇게 TAL 프로그래밍 환경이 엄격한 이유는 TAL의 운영환경이 모든 사용자에게 오픈되는 permissionless blockchain이기 때문이다. 오픈 환경에서 임의의 사용자에 의해 작성된 Trust Contract(Linked Data + TAL operator)의 경우 엄격한 보안 모델을 정의하지 않는다면 블록체인 네트워크에 예측할 수 없는 ~~되지 않는 상황~~ 문제를 발생시킬 수 있다.

1.1 Comparison of Blockchains

아래 표는 OWL-chain의 특징을 비트코인 및 이더리움의 특징과 비교한 것이다.

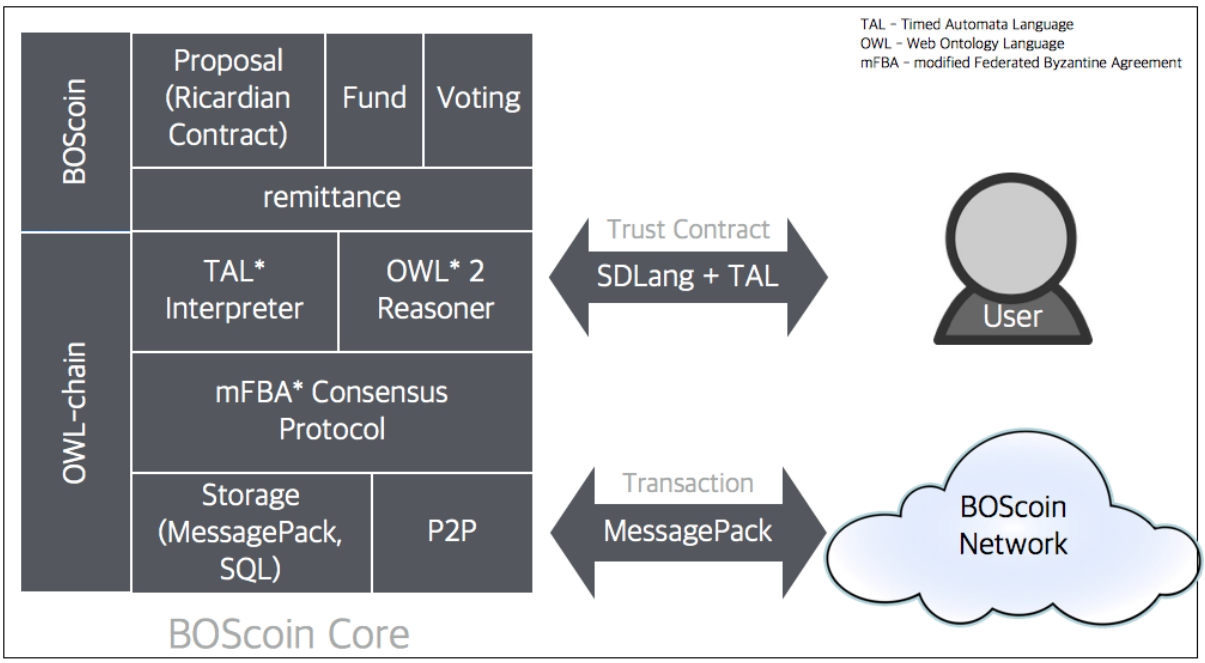
Feature	Bitcoin	Ethereum	OWL-chain(BOScoin)
Consensus	Proof of work	Current: Proof of work. Future: Casper(?)	Modified FBA(Federated Byzantine Agreement)
- History revision mechanism	Soft and hard forks	Current: Soft and hard forks. Future: Block revisions in case of temporary network isolation.	Block revisions in case of temporary network isolation.
- Membership	Open	Open	Open with constraint (min 10,000 BOS)
Block Confirmation Time	10 minutes	Current: 15 seconds	5 seconds (target)
Block Size	1 MB	Dynamic	Dynamic
Maximum Transaction	100 KB	Dynamic based on gas limit	Dynamic
Transaction Throughput	7 tx/sec	25 tx/sec	Target is 1,000 tx/sec
Coins	Bitcoin, plus tokens such as provided by Omni Layer	Ether, plus tokens that can be issued by contracts.	BOScoin
Governance system	None	None	Commons budget, Proposal, Voting
Smart Contracts: - Computational Power	Stack-based language with few instructions	Turing complete	Timed Automata Model
- Decidability	Decidable	Not Decidable, using	Decidable

		instruction fee(gas)	
- Runtime Architecture	Script runs on Bitcoin Core, Libbitcoin, and other native implementations	Ethereum Virtual Machine implemented on multiple platforms	OWL inference Engine on multiple platforms
- Programming Language	Bitcoin Script	Solidity, Serpent, LLL and any other languages that get implemented on the EVM.	OWL + TAL

[표 0] Comparison of Blockchains

2. Architecture Overview

OWL-chain의 계층 구조는 여러 온톨로지 모델을 수용할 수 있도록 범용 블록체인 모델로 설계되었다. OWL-chain은 추론엔진(inference engine)과 합의(consensus) 프로토콜의 기능을 결합하여 Trust Contract를 처리하게 된다. 추론엔진의 기본 타입(primitives data types)에 추가적으로 송금, 투표 등에 관련된 클래스 타입들이 블록체인을 통해 구현된다. 블록체인 기반의 TAL(Timed Automata Language)은 온톨로지 언어가 지니는 프로그래밍 언어로서 표현성 및 계산의 한계(limit)를 확장하는데 사용된다.



[Picture 1] BOScoin Architecture Diagram

2.1 Trust Contract

Contract Types Features	Smart Contract (Ethereum)	Ricardian Contract (R3CEV CORDA)[1,2]	Trust Contract (BOScoin)

프로그래밍 방식	통합언어 (LLL, Serpent, Solidity)	선언과 처리 분리 (Ricardian Contract + pure function)	선언과 처리 분리 (OWL* + TAL*)
Contract 결정성(decidability)	undecidable with gas(fee)	undecidable (3rd party evaluation)	decidable(TAL)
blockchain type	Permission-less	Permission	Permission-less
Consensus	PoW*	various	mFBA*
Contract Inference	None	None	OWL Reasoning
OWL*: Web Ontology Language TAL*: Timed Automata Language PoW*: Proof of Work mFBA*: modified Federated Byzantine Agreement			

[표 1] Contract Comparison

2.2 OWL(Web Ontology Language)

BOScoin의 트랜잭션은 OWL 2 Profile을 기반으로 사용자 정의가 가능한 Trust Contract를 지원한다. 기존의 블록체인 Contract는 가상 기계(Virtual Machine)를 사용해서 의미해석 기능을 제공하지 못 하였다. BOScoin의 Trust Contract는 시만텍 웹 기술에서 출발한 추론 엔진(inference engine)을 통해 트랜잭션의 문맥(semantic)을 블록체인 노드가 이해할 수 있다. 따라서 추론엔진은 입력된 요청을 해석하여 잘못된 Contract가 생성되는 것을 원천적으로 방지한다.

웹 환경에서 정확한 지식을 표현하는 방식은 W3C 표준화의 중요한 작업 중 하나로 진행되고 있다. OWL은 웹의 의미론(Semantics)를 지원하기 위해 개발된 표준언어이다. OWL 2 표준화는 이전 버전에서 문제가 되었던 성능 문제를 해결하기 위한 시도로 새롭게 프로파일 개념을 도입하여 용도에 맞게 OWL을 사용할 수 있도록 하였다. OWL 2 Profile은 3가지 규격으로 구성된다. 표현 타입이 복잡한 온톨로지의 경우(TBOX 중심) OWL 2 EL 프로파일을 선택하여 구성하면 최적화된 온톨로지 결과물을 얻을 수 있도록 하고 있다. 반면에 표현해야 할 개체(Individual)들이 많은 경우 OWL 2 QL 프로파일을 사용하여 좋은 온톨로지 데이터베이스를 얻을 수 있다. OWL-chain은 사용자가 OWL 2 프로파일을 선택하여 블록체인을 구성할 수 있도록 지원한다. 사용자는 구성하려는 온톨로지 데이터베이스 형태에 따라 프로파일을 결정하여 구성할 수 있도록 지원한다.

2.3 TAL(Timed Automata Language)

Owlchain(t`, OWLdata) = OWLdata @ Owlchain(t)

Owlchain(t`) : 새로운 블록체인 상태
Owlchain(t): 현 블록체인 상태
OWLdata: OWL 정의된 individual data set
@ : TAL (Timed Automata Language) Operator 저장소에 정의된 Timed Automata operator 이다.

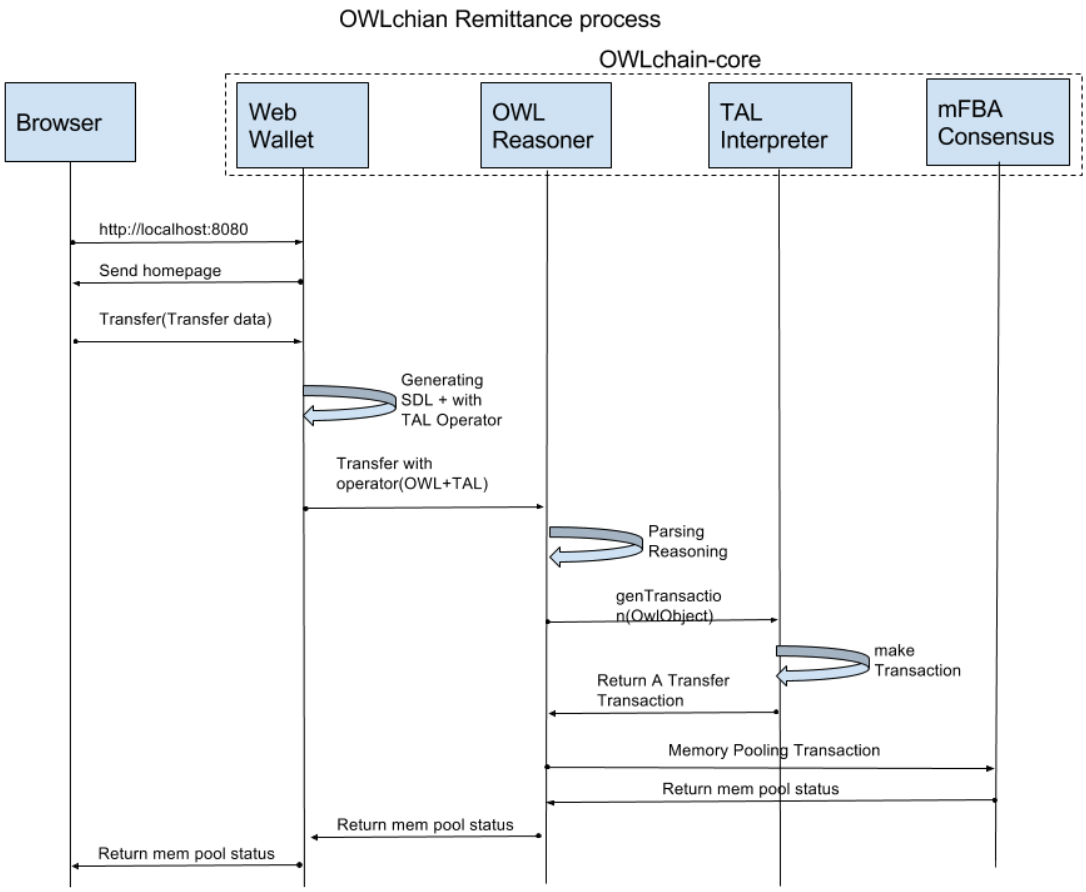
3. BOScoin

3.1 Remittance

BOScoin의 송금은 비트코인의 UTXO 패턴과 동일한 형식을 따른다. 다만 트랜잭션을 구성하는 방식이 bitcoin script와는 다르게 OWL 2 EL 규격의 문서로 작성된다. 이렇게 생성된 문서는 추론엔진에 의해 검증이 이루어지고 컨센서스 알고리즘을 통해 블록체인 네트워크로 확장된다.

```
Ontology {
    “http://blockchainos.org/remittance”
    Import "http://blockchainos.org/ontologies/remittance-v1.owl"
    Individual type="remittance" {
        Sender addr="1KrGTeQs55sf1zyTWWR4Y5qhe9Zxg2ftpy"
        Receiver addr="1FZNMuL8HUmf9TLdac62K4cGGpD2JEwnax" balance=1000 unit=BOS
        Receiver addr="1F1tAaz5x1HUXrCNLbtMDqcw6o5GNn4xqX" balance=500 unit=BOS
    }
    operator name="remitance" addr=“http://blockchainos.org/tal-repo/remittance-v1.tal”
}
```

[예제 1] 송금 - SDLang format



[그림 1] 송금 처리 다이어그램

3.2 Proposal - Trust Contract

투표의 대상이 되는 제안 proposal은 Ricardian Contract 형식으로 구성된다. [Picture 2]는 Ricardian Contract 형태로 제안 예제를 보여주고 있다. 제공되는 컨트랙트의 포맷은 SDLang 형태로 직렬화(serialization) 된다. Proposal는 1개의 트랜잭션으로 기록되고 해석이 완료되면 고유 트랜잭션 번호(number)를 부여받아 투표가 가능해 진다. Proposal은 보증금이 있는 형태와 없는 형태 2가지를 지원한다.

```
// Sample Proposal using SDLang format
Ontology {
    "http://blockchainos.org/proposal"
    Import "http://blockchainos.org/ontologies/proposal-v1.owl"
    Individual type="proposal" {
        Title "BOS Across The World"
        Owner "BOS-in-USA"
        Monthly-amount  BOS=180
        Completed-payments "no payments occurred yet (3 month remaining)"
        Payment-start-end   start=04-01-2017 end=19-04-2017 added-on=08-12-2016
        Please-vote-within  days=19
        Final-voting-deadline in-month=1
        Will-be-funded No // This proposal needs additional 232 Yes votes to become funded.

        Proposal-description {

            Description "BOS Across the World -- Weekly Show Interviewing Businesses and People"

            Overview "This is a 3-month pilot proposal to seek out real business owners, both
                conventional and unconventional, and conduct face-to-face interviews with them
                regarding the use of BOS and how it could be used."

            Scope "The scope of this project is not only to communicate the value of BOS to real
                people in real businesses, but it allows BOS developers and community to follow
                along and watch first-hand, how average people interact with BOS. Real-world
                interviews will be an invaluable feedback-loop to help eliminate or lessen the
                barriers to entry, while promoting BOS in creative and fun ways."

            Deliverables {
                "1. One show per week for 12 weeks. Tuesdays, delivered to various social media
                    channels like Youtube, and shared on Twitter."
                "2. Weekly frequent updates on the BOS.org proposal forum."
            }

            Schedule {
                "Each week, filming Wednesday to Friday (A+B footage)"
                "Each week, Saturday to Monday (Post, editing)"
                "Each week, Tuesday (Upload to social media channels)"
                "12 episodes in total"
            }

            Note "All audio-video, lighting and editing equipment is owned by me, and provided at
                no charge."
        }
    }
    Operator name="proposal" addr="http://blockchainos.org/tal-repo/proposal-v1.tal"
}
```

[Picture 2] BOScoin Proposal

3.3 Voting - decision making system

BOScoin의 투표 기능은 블록확인작업(block confirmation)에서 생성되는 Commons budget을 분배하는 메커니즘이다. 투표는 제안된 proposal에 3가지 형태로 가능하다. 선택 방법은 3가지 형태로 {positive, negative, neutral}의 선택이 가능하다. 중립 neutral은 무효표 기능을 가진다.

```
// Sample voting using SDLang format
Ontology {
    "http://blockchainos.org/vote"
    Import "http://blockchainos.org/ontologies/vote-v1.owl"
    Individual type="vote" {
        Voter addr="1KrGTeQs55sf1zyTWWR4Y5qhe9Zxg2ftpy"
        Proposal title="BOS Across The World" id="2C499D3D-F5E2-40D8-BFDF-C0A612E8E2AB"
        balot "positive" //{positive, negative, neutral}
    }
    Operator name="vote" addr="http://blockchainos.org/tal-repo/vote-v1.tal"
}

```

[Picture 3] Voting

3.4 Commons budget

제안된 Proposal에 대한 예산은 Commons budget에서 충당된다. Commons budget은 2가지 형태로 공급된다.

1. 제네시스부터 5년간 발행 - 블록 컨펌하는 노드에서 컨펌시마다 특수 트랜잭션 UTXO 형태로 발행하여 Commons budget 계좌에 적립된다.

2. 트랜잭션 수수료에 일부를 Commons budget 계좌에 적립한다.

4. Owlchain

OWL-chain은 추론엔진을 통해 데이터가 스스로를 설명(self-descriptive) 할 수 있게 설계되었다. 추론엔진은 OWL 2 표준언어를 해석할 수 있도록 제작되며, 검증이 완료된 계약을 operator를 통해 실행한다.

4.1 Semantic Reasoner

추론엔진은 OWL 2 규격으로 작성된 문서의 정합성(validation)을 검증한다. OWL문서는 class, property, individual 세 가지 구성요소로 이루어진다. class는 자료의 type을 정의하고 property는 속성을 기술하며, individual은 클래스 타입의 개별 인스턴스(instance)를 생성하기 위한 구문(statement)이다. Semantic Reasoner는 W3C OWL 2 profile 규격을 해석할 수 있도록 구현된다. OWL 2 profile은 Description Logic 문법을 따라 일정한 해석 성능을 보장하도록 작성되었다. [표 1]은 OWL 2 profile의 해석 시간을 표시한다. OWL-chain은 OWL 2 profile의 EL, QL 및 RL의 제약사항 내에서 구현된다. Profile 내에서 구현된 추론엔진은 [표 1]과 같은 시간 복잡도를 보증(guarantee)한다. 이는 추론엔진이 계산 결정성(decidability)을 확보하는 메커니즘으로 작동한다.

Language	Reasoning Problems	Taxonomic Complexity	Data Complexity	Query Complexity	Combined Complexity
OWL 2 EL	Ontology Consistency, Class Expression Satisfiability, Class Expression Subsumption, Instance Checking	PTIME-complete	PTIME-complete	Not Applicable	PTIME-complete

	Conjunctive Query Answering	in EXPTIME	PTIME-complete	NP-complete	in EXPTIME
OWL 2 QL	Ontology Consistency, Class Expression Satisfiability, Class Expression Subsumption, Instance Checking	NLogSpace-complete	In AC0	Not Applicable	NLogSpace-complete
	Conjunctive Query Answering	NLogSpace-complete	In AC0	NP-complete	NP-complete
OWL 2 RL	Ontology Consistency	PTIME-complete	PTIME-complete	Not Applicable	PTIME-complete
	Class Expression Satisfiability, Class Expression Subsumption, Instance Checking	PTIME-complete	PTIME-complete	Not Applicable	co-NP-complete
	Conjunctive Query Answering	PTIME-complete	PTIME-complete	NP-complete	NP-complete

[표 1] OWL 2 Profile Time Complexity

4.1.1 IRI Namespace Restriction

원래 OWL 2 profile은 인터넷상에서 참조하는 온톨로지를 확장하기 위해 IRI (Internationalized Resource Identifiers)를 사용한다. [picture 3]의 Prefix는 표현확장을 위해 사용하는 IRI 표현 예제이다. blockchain 기반의 온톨로지에서는 외부 노드에 보관된 온톨로지들을 사용하게 되면 온톨로지 영역이 블록체인 외부에 존재하게 되며, 외부참조 온톨로지의 경우 블록체인의 합의 알고리즘의 무결성 보장을 받을 수 없게 된다. 블록체인 네트워크 외부의 데이터에 대해서 블록체인이 무결성을 보장 할 수는 없다.

```
Prefix(:=<http://example.com/owl/families/>)
Prefix(otherOnt:=<http://example.org/otherOntologies/families/>)
Prefix(xsd:=<http://www.w3.org/2001/XMLSchema#>)
Prefix(owl:=<http://www.w3.org/2002/07/owl#>)
Ontology(<http://example.com/owl/families>
  Import( <http://example.org/otherOntologies/families.owl> )

  Declaration( NamedIndividual( :John ) )
  Declaration( NamedIndividual( :Mary ) )
  Declaration( NamedIndividual( :Jim ) )
  Declaration( NamedIndividual( :James ) )
  Declaration( NamedIndividual( :Jack ) )
  Declaration( NamedIndividual( :Bill ) )
  Declaration( NamedIndividual( :Susan ) )
  Declaration( Class( :Person ) )
  AnnotationAssertion( rdfs:comment :Person "Represents the set of all people." )
  Declaration( Class( :Woman ) )
  Declaration( Class( :Parent ) )
```

[Picture 4] OWL 2 IRI

BOScoin의 IRI 참조영역(reference namespace)은 블록체인의 합의 알고리즘의 무결성을 보장할 수 있는 https://blockchianos.org IRI 네임스페이스(namespace)로 영역이 제한된다. 이 영역은 블록체인 내부에 존재해서 블록체인의 트랜잭션으로 기록하는 것이 승인된 영역이다. 향후 고려사항으로 다른 블록체인과의 온톨로지 공유 문제가 제기 될 수 있다. 온톨로지 공유는 블록체인간 상호 거래에서 트랜잭션의 의미를 합의하는데 중요한 이슈이지만 이번 스펙에서는 다루지 않는다. 추후에 온톨로지 기반의 블록체인간에 Trust Contract의 교환/거래가 필요한 상황에서 논의할 수 있을 것이다.

```
## BOS IRI based on http://blockchainos.org
Prefix(:=<http://blockchainos.org/owl/families/>)
Prefix(otherOnt:=<http://blockchainos.org/otherOntologies/families/>)
Prefix(xsd:=<http://blockchainos.org/w3c/2001/XMLSchema#>)
Prefix(owl:=<http://blockchainos.org/w3c/2002/07/owl#>)

Ontology(<http://blockchainos.org/owl/families>
  Import( <http://blockchainos.org/otherOntologies/families.owl> )

  Declaration( NamedIndividual( :John ) )
  Declaration( NamedIndividual( :Mary ) )
```

[Picture 5] BOScoin IRI Restriction

4.2 Development Components

4.2.1 SDLang

블록체인의 사용자 계층을 나누어 보면, 개발자와 지갑사용자(최종 사용자), 그리고 컨트랙트 작성자로 구분할 수 있다. 이 사용자들간에 공통관심사는 Trust Contract의 내용이다. SDLang은 세 그룹의 사용자간의 사용 편리성과 표현력의 균형을 위해 선택되었다. SDLang 언어 형식 자체는 개발자에게 친숙한 개발언어의 자료 표현형에서 기반하며 구조화된 데이터의 표현도 간단하게 수행할 수 있다. 자세한 사항은 SDLang 언어 소개를 참조한다.

```
// This is a node with a single string value
title "Hello, World"
// Multiple values are supported, too
bookmarks 12 15 188 1234
// Nodes can have attributes
author "Peter Parker" email="peter@example.org" active=true
// Nodes can be arbitrarily nested
contents {
    section "First section" {
        paragraph "This is the first paragraph"
        paragraph "This is the second paragraph"
    }
}
// Anonymous nodes are supported
"This text is the value of an anonymous node!"
// This makes things like matrix definitions very convenient
matrix {
    1 0 0
    0 1 0
    0 0 1
}
```

[Picture 5] SDLang format

4.2.2 MessagePack Serialization

MessagePack은 구글 protocolBuffer와 같은 파일, 자료구조 등의 직렬화(serialization)에 사용되는 규격이다. MessagePack은 직렬화를 위해 IDL파일 등을 요구하지 않아 프로그래머의 직렬화 작업을 쉽게 할 수 있다. 또한 Zero-copy 직렬화를 보장해서 블록체인처럼 대규모 네트워크 상의 프로토콜 패킷 교환에도 성능상의 이점이 있다.

```
import std.file;
import msgpack;

struct S { int x; float y; string z; }

void main()
{
    S input = S(10, 25.5, "message");

    // serialize data
    ubyte[] inData = pack(input);

    // write data to a file
    write("file.dat", inData);

    // read data from a file
    ubyte[] outData = cast(ubyte[])read("file.dat");

    // unserialize the data
    S target = outData.unpack!S();

    // verify data is the same
    assert(target.x == input.x);
    assert(target.y == input.y);
    assert(target.z == input.z);
}
```

[Picture 6] MessagePack Serialization/Unserialization

4.3 Modified FBA Consensus

global blockchain network의 건전성(integrity)을 유지하는 것은 어려운 일이다. 비트코인의 p2p 비잔틴 장군 문제는 비트코인과 같은 블록체인 기술이 등장하기 이전부터 암호학의 한 분야로 연구 되었다. 연구에 따르면, 충분한 수의 정상 동작하는 노드가 존재한다면 시스템은 다른 노드들의 비정상적인 동작 혹은 공격으로부터 저항성(resistance)을 가질 수 있지만, PBFT(Practical Byzantine Fault Tolerance)와 같은 기존의 비잔틴 장군 문제의 솔루션들은 관리자에 의한 노드 통제가 전제된다.

비트코인은 작업 증명(PoW)이라는 차별화된 접근 방식을 통해 네트워크를 개방(open membership)하는데 성공했지만, 기존의 비잔틴 장군 문제의 솔루션들이 가지고 있던 속도(low latency), 유연한 신뢰(flexible trust), 이론적 보안성(asymptotic security)은 보장되지 않는다.

최근에는 다시 학문으로 돌아가서 전통적인 비잔틴 장군 솔루션의 장점들을 유지하면서 네트워크를 개방할 수 있는 방안에 대한 연구가 관심을 끄는데, Federated Byzantine Agreement가 가장 완성도 높은 결과 중 하나로 평가 받고 있다.

4.3.1 Quorum and Quorum slice topology for Federation

* 인터넷 토폴로지와 유사한 Quorum 운영 (분산성 및 속도에 대한 효율성)
* 지능 알고리즘을 이용한 최적의 쿼럼 자동 업데이트 기능(아직 아이디어 수준)

4.3.2 Decentralized Global Consensus

mFBA의 핵심 가치는 네트워크의 개방성이다. 기존의 비잔틴 장군 솔루션들은 비정상 동작을

알아내기 위해 시스템 전체의 합의가 필요했기 때문에 신뢰할 수 있는 폐쇄적 네트워크가 전제 되었다. FBAS는 전체 시스템을 정족수 단위(quorum slice)로 나눈 뒤, 정족수가 합의한 내용을 네트워크 전체가 공유하는 방식으로 설계 되었기 때문에, 네트워크 구성에 있어서 높은 자유도를 가진다.

따라서 보스코인 생태계를 구성하는 노드들은 네트워크에서 요구하는 최소 조건만 만족한다면 자유롭게 네트워크의 구성원이 될 수 있다. 조건에 대한 예를 들면, 블록 생성 속도를 유지하기 위한 최소 네트워크 대역폭 사이즈 등이 요구 될 수 있다.

4.3.3 Stake Features

BOScoin membership을 가지고 블록생성 등과 같은 작업을 하기 위해서는 최소한의 납입증명(Proof of Stake)이 필요하다. 10,000 BOS당 1개의 트랜잭션을 발생시키고 해당 트랜잭션 번호(number)를 통해 소유권을 증명한 노드만이 합의 네트워크에 참가할 자격을 획득할 수 있다. FBAS의 구현체인 Stellar Consensus Protocol(SCP)의 자격 증명이 Quorum 멤버십의 증명으로만 가능한 것과 차이가 있다.

4.3.4 Reward Features

confirm reward - 결과

5. Implementation Roadmap

BOScoin은 컴패니언 서비스와 연계되어 관련 기능들과 ICO일정 등을 고려하여 개발한다.

- P2P
- FBA Consensus
- Remittance
- Web Interface
- Ricardian Contract Proposal & Vote
- SPV(simple payment verification) wallets
- Inference Engine
- Smart Contract Modeler
- RPC & REST API

Milestone → ↓ Module	M1	M2	M3	M4
P2P	Protocol specification & Implementation	Unit & Acceptance Test		
mFBA Consensus	Key design Implementation	mFBA Acceptance Test		
Remittance	Address design UTXO Pattern Send & Receive coin			

Data Store	Store specs & SQLite Store implementation	MessagePack History		Blockchain backup & restore using ISP(AWS,Azure and google)
CLI & Web Interface	CLI design & implementation	Web UX design		
Trust Contract Proposal & Vote		Trust Contract design & specification	Proposal & Vote implementation	
SPV(simple payment verification) wallets		Wallet Formal specification	UX design Application PoC Test	Android & iOS Wallet
Inference Engine	Formal specification and key design elements available	Reasoner integration with Blockchain		
Trust Contract Modeler			Formal specification and key design elements available	App Deployment & Demo Site
RPC & REST API			Blockchain Explorer	

[표 2] Implementation roadmap

A. Reference

A.1 FBA Consensus

https://www.stellar.org/papers/stellar-consensus-protocol.pdf
https://medium.com/a-stellar-journey/on-worldwide-consensus-359e9eb3e949#.izg7ydd08
https://news.ycombinator.com/item?id=9341687
https://www.reddit.com/r/ethereum/comments/338mip/stellar_consensus_protocol_canwill_something_like/

A.2 Web Ontology Language - OWL

http://www.semantic-web-journal.net/sites/default/files/swj120_2.pdf
https://www.w3.org/TR/owl2-primer/
https://www.w3.org/TR/owl2-overview/
https://www.w3.org/TR/owl2-profiles/
http://www.semantic-web-journal.net/sites/default/files/swj120_2.pdf
fact++ owl reasoner - http://owl.man.ac.uk/factplusplus/

A.3 TAL(Timed Automata Language)

A theory of timed automata* - http://www.cis.upenn.edu/~alur/TCS94.pdf

uppaal language reference - http://www.uppaal.com/index.php?sida=217&rubrik=101
Modeling Bitcoin Contracts by Timed Automata - https://arxiv.org/pdf/1405.1861v2

A.4 Serialization & Parsing Tools

Structured Ontology Format - http://webont.org/owled/2007/PapersPDF/submission_18.pdf
SDLang - https://sdlang.org/
MessagePack - http://msgpack.org/
PEG - https://en.wikipedia.org/wiki/Parsing_expression_grammar
Bitcoin Script - https://en.bitcoin.it/wiki/Script
Bitcoin UTXO - https://bitcoin.org/en/glossary/unspent-transaction-output

[1] Jamie Redman, R3CEV Unveils Corda, But 'Is Not Building a Blockchain', April 6, 2016, https://news.bitcoin.com/r3cev-corda-is-not-building-a-blockchain/
[2] Richard Gendal Brown, Introducing R3 Corda™: A Distributed Ledger Designed for Financial Services, April 5, 2016, http://www.r3cev.com/blog/2016/4/4/introducing-r3-corda-a-distributed-ledger-designed-for-financial-services

B. Appendix

B.1 Timed Automata Language PEG(Parsing Expression Grammar)

{
TAL 설계 사양서 소개...
}

Operator repository specification

C. Coding Style Guide

C.1 D Coding Style

D 프로그램을 작성하기위한 코딩 규약을 준수하면 작성한 코드에 대한 가독성이 높아지고 다른 사람과 코드를 통한 협업이 용이해 진다.

* Whitespace(공백)

. 한 줄에 한 문장.
. 하드웨어 탭 대신 space를 사용.
. 각 들여 쓰기 레벨은 4칸

* 일반적인 코드 작성

아래에 달리 열거되지 않는 한, 모든 변수를 포함하여 camelCased 작성을 기본으로 한다.
따라서 여러 단어를 결합하여 형성된 이름은 첫 번째 단어가 대문자가 아닌 다른 단어를 갖는다.
또한 private이 아닌 경우 밑줄 '_'로 시작하지 않는다.

```
ex) int myFunc();
    string myLocalVar;
```

* Modules(모듈)

모듈 및 패키지 이름은 모두 소문자여야하며 문자 [a..z] [0..9] [_] 만 포함한다.
이렇게하면 대소 문자를 구별하지 않는 파일 시스템을 다룰 때 문제가 발생하지 않는다.

```
ex) import std.algorithm;
```

* Classes, Interfaces, Structs, Unions, Enums, Non-Eponymous Templates

사용자 정의 유형의 이름은 PascalCased로 작성한다.
(첫 번째 문자가 대문자인 것을 제외하면 camelCased와 동일)
Enum member는 camelCased로 작성한다.

```
ex) class Foo;
    struct FooAndBar;
    enum Direction { bwd, fwd, both }
```

*Eponymous Templates

해당 템플릿 내의 심볼과 동일한 이름을 가진 템플릿은 대문자로 표시한다.
(따라서 해당 템플릿의 인스턴스화는 해당 심볼로 대체됨)

```
ex) template GetSomeType(T) { alias GetSomeType = T; }
    template isSomeType(T) { enum isSomeType = is(T == SomeType); }
    template MyType(T) { struct MyType { ... } }
    template map(fun...) { auto map(Range r) { ... } }
```

* Functions

함수 이름은 camelCased로 작성하며, 속성 및 멤버 함수도 포함된다.

```
ex) int done();
    int doneProcessing();
```

* Constants(상수)

상수의 이름은 일반 변수와 마찬가지로 camelCased로 작성한다.

```
ex) enum secondsPerMinute = 60;
    immutable hexDigits = "0123456789ABCDEF";
```

* Keywords

이름이 키워드와 충돌 할 경우 다른 이름을 선택하는 대신 키워드를 사용하는 것이 바람직하고 이 경우 밑줄 '_'를 하나 추가해야 한다.
키워드와의 충돌을 피하기 위해 이름을 대문자로 사용해서는 안된다.

```
ex) enum Attribute { nothrow_, pure_, safe }
```

C.2 HTML5 Coding Style

owlchain 프로젝트의 HTML5 코딩 규약은 아래 구글 스타일 가이드를 기반으로 수정 보완한다.
(참고 - https://google.github.io/styleguide/htmlcssguide.xml#HTML_Formatting_Rules)