퀀텀 백서

분산 모바일 애플리케이션 플랫폼에서의
스마트 계약 가치 전송 프로토콜

Patrick Dai, Neil Mahi, Jordan Earls, Alex Norta

서론 : 트랜잭션을 위하여 지분 증명 검증을 채택한 스마트 계약을 지원하는 Blockchain 은 작업 증명 솔루션에 비해 상당한 성능 이점을 보장합니다. 광범위한 산업에 적용하기 위해서는 다른 중요한 요구 사항도 충족시켜야 합니다. 예를 들어, 안정적으로 이전 기종과 호환되는 스마트 계약 시스템은 간단한 지불 증명(simple payment verification, SPV) 기술을 지원하는 라이트 모바일 지갑을 사용하여 조직간 정보 교류 통합을 반드시 자동화해야 합니다. 현재 스마트 계약 솔루션을 이끌고 있는 이더리움은 비용이 많이 드는 작업증명을 사용하여 미래에 하드 포크를 여러 번 예상할 수 있고, 전체 Blockchain을 다운로드 해야 합니다 .따라서 이더리움 스마트 계약은 보안상 문제가 있어 유틸리티를 제한하고 공식적인 의미가 결여되어 있습니다. 이 백서는 사회기술적 응용 프로그램의 적합성, 공식적인 의미의 언어 표현력 적용, 그리고 빠르고 숙련된 산업 적용을 위한 스마트 계약 견본 라이브러리의 공급을 목표로 하는 Qtum 스마트 계약 프레임 워크를 제시함으로써 기술의 상태와 격차를 없앱니다. 우리는 이더리움 대안으로써의 Qtum 유틸리티의 장점을 논의하고 산업 기반 어플리케이션으로써의 미래 Qtum 스마트 계약 개발 계획을 제시합니다.

Key words : smart contract, business network model, DAPP, mobile, information logistics, cross-organizational, peer-to-peer, distributed system, e-governance, Qtum framework

1. 도입

당사자 간의 협상 동의를 의미하는 컴퓨팅을 가능하게 하고 검증하고 실행하는 통합과 실행 프로토콜을 스마트 계약 이라고 합니다. 스마트 계약은 처음에 금융기법[6], 사물인터넷(IoT) 어플리케이션[33], 디지털 서명 솔루션[11] 같은 다양한 영역에서 응용 프로그램이 발견됩니다. 스마트 계약의 필수적인 사항은 거래의 분산 검증입니다. 그것은 초기에는 소위 작업증명(PoW)을 의미합니다. 스마트 계약을 가능하게 하는 핵심 기술은 트랜잭션 이벤트를 신뢰할 수 있는 중앙 기관에 보내지 않고 기록하는 Blockchain이라 불리는 공공 분산 원장입니다. Blockchain 기술은 프로토콜 계층에 대한 제한된 작업 집합으로 구성된 Bitcoin[23], a peer-to-peer(P2P) cryptocurrency and payment 의 시작과 함께 크게 인기를 얻었습니다. 비트 코인은 트랜잭션 검증을 위해 PoW를 사용하여 전기 소모가 많아 많은 비용이 듭니다.

Bitcoin과 대조적으로, 많은 스마트 계약 시스템은 작성에 있어 자바스크립트의 문법과 대상이 비슷한 튜링-완전 언어 솔리디티를 갖추고 있습니다. 예를 들면 이더리움 가상[44] 머신이 있습니다. 이더리움은 사실상 선도적인 스마트 계약 시스템이지만 여러 가지 결함에 가진 채 전파되었습니다. 첫째, 작업 증명 트랜잭션 확인은 확장성을 감소시켜 대부분의 어플리케이션에서 이더리움의 사용이 적합하지 않습니다. 둘째, 최근 crowdfunding 사례를 연구하여 보면, 이더리움 계열 솔리디티 스마트 계약은 확실한 검증[3]을 위한 방법과 관련하여 보안 취약성 때문에 해킹되었습니다. 보안 결함으로 인해 약 5,000만 $의 손실이 발생했습니다. 그 결과, 이더리움은 별도의 이더리움 버전을 허용하고 분열되어 hardfork가 일어나게 되었습니다. 하지만 이더리움 hardfork는 서비스 거부 공격으로 인해 발생했고, proof-of-stake[2] 거래 검증 및 Blockchain 샤딩(분할)을 실현하기 위해 더 많은 hardfork가 반드시 필요합니다.

이 외에도 더 많은 이유로 인하여 이더리움이 관련 산업에 채택되는데 대해 광범위한 제약을 받고 있습니다. 예를 들어 조직간 정보 교류를 자동화하는 능력이 없고 외부와 관련된 내부 전용 계약과 관련하여 프라이비시 보호 차별화가 부족하며, 블록체인을 위해 더 나은 proof-of-stake 거래 검증을 수행하는 안전하고 안정적인 가상 머신, 스마트 컨트랙트 언어의 공식적인 검증 그리고 전체 블록체인을 다운로드하지 않아도 되는 라이트 지갑과 단순 지불 증명(SPV)[14]스마트 컨트랙트를 지원하는 모바일 장치 솔루션 등이 부족합니다. SPV은 클라이언트가 임의의 전체 노드[23]에 연결할 때 단지 블록 헤더만 다운로드 하는 것을 의미합니다.

Qtum은 좀 더 적당한 대안의 현실적인 부족에 따라[19] 사실상 이더리움 가상 머신 (EVM)을 사용 하는 동안 EVM에는 잘못 처리된 예외에 대항하고 트랜잭션 순서 지정, 타임 스탬프 등의 의존도에 대항하여 과거 경험이 있는 공격에 대하여 결함이 있음을 발견하였습니다. 또한, EVM은 Bitcoin 또는 ColoredCoin[36]과 같은 다른 Blockchain 시스템에 대한 호환이 가능함에 따라 산업 적용을 확장하기 위해서는 사이드 체인[10] 및 소비되지 않은 트랜잭션 출력 값(unspent transaction outputs, UTXO)[10]을 사용하는 스마트계약 시스템에 바람직하며, Bitcoin Lightning Network[35] 기능 채택으로 양방향 소액 결제 채널을 통한 확장성을 확보할 수 있습니다.

이더리움의 스마트 계약 시스템은 많은 주목을 끌고 있지만, 위의 같은 이유로 인하여 관련 산업에서의 적용은 많지 않습니다. 이 백서에서는 비용과 시간을 줄이기 위해 조직 간 정보 교류를 가능하게 하고 주요 고객 요구 사항을 충족하기 위한 스마트 계약 솔루션을 개발하는 방법에 대한 질문에 대하여, 그에 대한 대답으로 스마트 계약 시스템용 Qtum 프레임 워크를 제시함으로써 해결하고자 합니다. 우려 사항을 해결하고자, 우리는 다음과 같은 하위 질문을 제기합니다.

Qtum 스마트 계약 솔루션은 차별화된 기술적 성능 이점으로 무엇을 제공 합니까?
Qtum 프레임워크가 추구하는 주요 스마트 계약 요구 사항은 무엇입니까?
Qtum 프레임워크가 지향하는 조직 간 정보 교류 자동화의 고유한 특징은 무엇입니까?

이 백서의 나머지 부분은 다음과 같이 구성되어 있습니다. 첫째, 섹션 2는 관련 솔루션과 비

교하여 기술적으로 성능 향상을 달성하기 위한 Qtum 프레임워크의 구체적인 장점에 초점을 맞춥니다. 섹션 3에서는 사회기술적으로 조직된 스마트 계약 시스템을 위해 관련 이해 관계자들과 함께 기능적 목표 및 품질 목표를 제공합니다. 섹션 4에서는 Qtum 프레임 워크 가치 전송 프로토콜에 의해 지원되는 운용 사례를 보여줍니다. 마지막으로, 섹션 5에서는 이 백서에 대한 제한 사항, 미해결 문제 및 향후 개발 작업에 대해 논의합니다.

2. Qtum Performance Advantage(Qtum 실행 이점)

Qtum의 주요 목표 중 하나는 지분 증명 (PoS)[37] 합의 모델을 사용하여 첫 번째 UTXO 기반 스마트 계약 시스템을 만드는 것입니다. POS는 다음 블록의 생산자가 암호화 화폐의 보유량을 기반으로 선택되는 것을 의미합니다. 따라서 블록은 채굴되는 대신 일반적으로 단조하거나 주조되며, 거기에는 블록 보상으로 거래 수수료뿐만 아니라 그들의 지분 자금에 대한 일정 율의 "이윤" 을 받을 수 있습니다.

Qtum은 비트코인 및 이더리움 생태계와 호환이 되고 이더리움 가상 머신 (EVM) 호환성을 가지고 Bitcoin의 변화가 일어나는 것을 목표로 합니다. 참고로 Qtum EVM은 이더리움과 달리 끊임없이 호환됩니다. 실용적인 디자인 접근법을 추구하는 Qtum은 모바일 장치로 구성된 전략을 가지고 업계에서 사용할 수 있는 케이스를 채택하고 있습니다. 모바일 장치는 사용자의 폭넓은 확대를 통하여 Blockchain 기술을 촉진시킴으로써, Qtum의 PoS 트랜잭션 검증을 분산시킬 수 있습니다.

2장은 다음과 같이 구성 됩니다. 섹션 2.1 Bitcoin UTXO 대 이더리움 계정 모델의 장점을 비교합니다. 섹션 2.2에서는 Qtum 블록체인에 대한 합의 플랫폼에 대해 설명합니다. 섹션 2.3는 EVM과의 Qtum 계약 통합을 보여줍니다. 마지막으로, 2.4 절에서는 Qtum 작업에 대한 지불 모델에 대하여 설명합니다.

2.1 UTXO Versus Account Model(UTXO 대 계정 모델)

UTXO 모델에서 소비되지 않은 비트 코인을 입력값으로 사용한 트랙잭션이 이루어지면 트랜잭션 출력값으로 새로운 UTXO가 생성됩니다. 소비되지 않은 트랜잭션 출력값은 변경되어 spender(소비자)로 반환 됩니다. 이러한 방식으로 비트 코인의 특정 볼륨이 다른 개인키 소유자 간에 전송되고 트랜잭션 체인에서 새로운 UTXO를 소비하고 생성합니다. Bitcoin 트랜잭션의 UTXO는 트랜잭션의 새로운 버전에 서명하는 데 이용되는 개인키를 가지고 잠금이 해제됩니다. Bitcoin 네트워크에서 채굴자는 어떤 입력을 포함 하지 않는 coinbase 거래라는 프로세스를 가지고 비트코인을 생성합니다. Bitcoin은 트랜잭션에 대해 제한된 작동 세트를 가진 스크립팅 언어를 사용 합니다. Bitcoin 네트워크에서 스크립팅 시스템은 데이터를 Last-In, First-Out의 LIFO 원칙에 따른 추상적인 데이터 형식인 스택(메인 스택 및 알트 스택)으로 처리 합니다.

Bitcoin 클라이언트에서 개발자는 스크립팅 형식을 요약하기 위하여 isstandard () 함수를 사

용합니다. Bitcoin 클라이언트는 P2PKH(공개키 해시에 지출), P2PK(공개키에 지출), MultiSignature(다중서명, 15개 미만의 키 서명), P2SH(스크립트 해시에 지출), 그리고 OP_RETURN을 제공합니다. 이 다섯 가지 표준 스크립팅 유형으로 Bitcoin 클라이언트는 복잡한 지불 논리를 처리할 수 있습니다. 게다가, 비표준 스크립트를 만들 수 있으며, 채굴자가 표준이 아닌 거래를 캡슐화에 동의하는 경우 실행됩니다.

예를 들어, 스크립트 생성 및 실행 프로세스에 P2PKH를 사용하여, 가상의 Bitcoin 주소 "Bread Address"를 가지고 빵집에서 빵을 사기 위하여 0.01 btc를 지불한다고 가정합니다. 이 트랜잭션의 출력은 다음과 같습니다.

OP_DUP OP_HASH160 <Bread Public Key Hash> OP_EQUAL OP_CHECKSIG

OP_DUP 작업은 스택의 맨 위 항목을 복제 합니다. OP_HASH160는 비트 코인 주소를 최상위 항목으로 반환 합니다. 비트 코인의 소유권을 설정하려면 디지털 키 및 디지털 서명과 함께 Bitcoin 주소가 필요합니다. OP_EQUAL는 맨 위 두 항목이 정확하게 동일하면 TRUE(1), 아니면 FALSE(0)를 반환합니다. 마지막으로, OP_CHECKSIG는 트랜잭션의 해시된 데이터에 대한 서명과 관련된 유효성 검사와 함께 공개키와 서명을 생성하며, 일치가 발생 하면 TRUE를 반환합니다.

잠금 스크립트에 따른 스크립트의 잠금 해제는:
<Bread Signature> <Bread Public Key>
위 두개의 결합 스크립트는:
<Bread Signature> <Bread Public Key> OP_DUP OP_HASH160
<Bread Public Key Hash> OP_EQUAL OP_CHECKSIG
잠금 해제 스크립트와 잠금 스크립트의 미리 정의 된 조건이 일치하는 경우에만 스크립트 조합이 true로 실행됩니다. 즉, Bread Signature는 유효한 Bread Address 서명의 개인키를 일치시켜 서명해야 하며 그 결과는 true입니다.

불행히도, Bitcoin의 스크립팅 언어는 완전한 튜링이 아닙니다. 예를 들어, 루프 기능이 없습니다. Bitcoin 스크립팅 언어는 일반적으로 사용되는 프로그래밍 언어가 아닙니다. 이러한 제한으로 무한 루프 생성이나 기타 복잡한 논리 허점 같은 복잡한 지불 조건의 발생을 방지하여 보안 위험을 완화합니다.

UTXO 모델에서는 공공 원장을 통해 각 트랜잭션의 히스토리를 투명하게 추적할 수 있습니다. UTXO 모델은 확장성을 나타내는 여러 주소 간에 트랜잭션을 초기화하는 병렬 처리 기능을 가지고 있습니다. 또한 UTXO 모델은 사용자가 UTXO의 출력으로 주소 변경을 사용할 수 있어 개인 정보 보호를 지원합니다. Qtum의 목표는 UTXO 모델의 혁신적인 디자인을 기반으로 스마트 계약을 구현 하는 것입니다.

UTXO 모델에 비해 이더리움은 계정 기반 시스템입니다. 더 정확하게, 각 계정은 상태 변환으로 직접적인 가치 및 정보 전송을 수행합니다. 20바이트의 이더리움 계정 주소는 트랜잭션에

대한 일회성 처리를 보장하기 위한 카운터와 이더 로 불리는 거래 비용을 지불하기 위한 주요 내부 암호화 연료의 잔고 및 선택적 계약 코드 및 채우지 못한 계정 저장소로 구성됩니다.

이더 계정의 2 가지의 유형은 개인키를 외부와 통제하는 것과 계약 코드를 통제하는 것입니다. 이전은 코드가 없는 계정 타입이 메시지 전송을 위한 트랜잭션을 만들고 서명 합니다. 나중은 내부 저장소 읽기 및 쓰기, 계약 만들기 또는 다른 메시지 전송에 대한 위한 메시지를 받은 후에 코드가 활성화 됩니다.

이더리움에서는 잔액 관리가 현실 세계의 은행 계좌를 닮았습니다. 새로 생성된 모든 블록은 다른 계정의 글로벌한 지위에 영향을 줄 수 있습니다. 모든 계정에는 다른 계정이나 주소를 호출하기 위한 스토리지와 코드 공간 베이스를 가지고 있으며 각각의 실행 결과를 저장합니다. 기존의 이더리움 계정 시스템에서 사용자는 클라이언트 원격 절차 호출을 통해 P2P 트랜잭션을 수행합니다. 스마트 계약을 통해 더 많은 계정에 메시지를 보내는 것이 가능하지만 이러한 내부 트랜잭션은 각 계정의 잔액으로만 표시되어 이더리움의 공공 장부에서 이를 추적하는 것은 어려운 일입니다.

위의 설명에 따라 이더리움 계정 모델을 확장성 병목 현상으로 간주하고 Bitcoin 네트워크 UTXO 모델의 명확한 이점을 살펴보았습니다. Bitcoin 네트워크 UTXO 모델은 우리가 제안하고자 하는 네트워크 효과를 향상시키기 때문에, 보류 중인 Qtum 방출을 위한 근본적인 디자인은 UTXO 모델의 적용하는 것으로 결정합니다.

2.2. Consensus Management(합의 관리)

합의에 대하여는 각 프로젝트 요구 사항을 충족하는 플랫폼이 있어 지속적인 논의가 진행중입니다. 가장 널리 논의되는 합의 방식으로는 PoW[41], PoS[2], Dynamic PoS 그리고 HyperLedger에 의해 논의되는 Byzantine Fault Tolerance[7] 등이 있습니다. 합의의 특성은 분산 알고리즘에 데이터 일관성을 달성하는 것입니다. 사용할 수 있는 옵션은, 예를 들어, Fischer Lynch와 Paterson의 정리[5]에 의한, 각 노드 간 100% 합의 없이는 상황 합의에 도달할 수 없다는 것입니다.

Bitcoin 네트워크에서 마이너는 POW를 통해 해시 충돌을 확인하는 과정에 참여하고 있습니다. 마이너의 해시 값이 특정 조건을 계산하고 만족할 때, 마이너는 새로운 블록이 채굴된다는 것을 네트워크에 주장할 수 있습니다.

$$Hash(BlockHeader) \leq M/D$$

마이너 M과 채굴 난이도 D의 계산을 위해 Hash()는 값 범위 [0, M] 및 D가 있는 SHA256 파워를 나타냅니다. 비트 코인에 의해 사용 되는 SHA256 알고리즘은 마이너의 수가 마이닝 난이도에 비해 많을 경우 모든 노드가 신속하게 각 블록을 확인할 수 있습니다.

80바이트의 Blockheader는 각각의 Nonce에 따라 달라집니다. 채굴의 전반적인 난이도 수준

은 Blockchain 네트워크의 총 해시 파워에 따라 역동적으로 조정됩니다. 둘 이상의 마이너가 동시에 블록을 생산하면 네트워크에서 작은 포크가 발생합니다. 이것은 Blockchain이 그것을 받아들여야 하는 블록인지 거부해야 하는지 결정을 내리는 데 중요한 포인트입니다. Bitcoin 네트워크에서 가장 입증된 작업을 첨부한 체인이 적법합니다.

대부분의 PoS Blockchain은 Bitcoin Core의 이전 버전에 근거한 PeerCoin 에 기초한 유산에서 그 기초를 찾을 수 있습니다. 다른 PoW 알고리즘으로는 Scrypt, X11, Groestl1, Equihash[4] 등이 있습니다. 새로운 알고리즘 런칭의 목적은 하나의 독립체에 의해 컴퓨팅 파워의 축적을 방지하고 응용 프로그램 특정 집적 회로 (ASIC)는 도입할 수 없음을 보장하기 위함입니다. Qtum Core는 기초 합의 형성을 위하여 최신 Bitcoin 소스 코드에 기반한 PoS 를 선택 합니다.

전통적인 PoS 트랜잭션에서 새 블록의 생성은 다음 조건을 충족해야 합니다.

$$\text{ProofHash} < \text{coins} \times \text{age} \times \text{target}$$

ProofHash에서 스테이크 수정자[40]는 소비되지 않은 출력값(코인)과 현재 시간을 함께 계산합니다. 이 방법을 사용하면 한 악의적인 공격자가 많은 양의 coin age 을 누적하여 이중 지출 공격을 시작할 수 있습니다. coin age에 기인한 또 다른 문제는 노드가 지속적인 접속에 따른 보상 후에 간헐적으로 접속하는 것입니다. 그러므로 PoS 합의의 개량 버전에서 coin age 조정은 더 많은 노드를 동시에 접속하도록 독려합니다.

원래의 PoS 구현은 coin age 공격 및 기타 유형의 공격 가능성으로 인해 여러 가지 보안 문제를 겪고 있습니다[16]. Qtum는 Blackcoin 팀의 보안 분석[40]에 동의하고 최종 Qtum Core로 PoS 3.0를 채택합니다. PoS 3.0은 이론적으로 지갑을 오프라인으로 둔 코인 소유자에게 인센티브를 주지 않는 대신, 더 오래 코인을 걸어놓은 투자자들에 보상합니다.

2.3 Qtum Contract and EVM Integration(Qtum 계약과 EVM의 통합)

EVM은 256비트 머신 언어를 사용하는 스택 기반입니다. 이더리움에서 실행되는 스마트 계약은 실행을 위해 이 가상 머신을 사용합니다. EVM은 이더리움의 Blockchain을 위해 디자인되고 계정을 기반으로 하는 방법을 이용하여 모든 가치를 이동한다고 가정합니다. Qtum은 Bitcoin의 Blockchain 설계를 기반으로 하며 UTXO 기반 모델을 사용합니다. 따라서 Qtum에는 UTXO 기반 모델을 EVM의 계정 기반 인터페이스로 변환하는 계정 추상화 단계가 있습니다. 참고로 컴퓨팅의 추상화 단계는 상호 운용성과 플랫폼 독립성을 촉진하는 데 대한 우려를 불식시키기 위해 특정 기능의 구현 세부 사항을 숨기는데 중요한 역할을 합니다.

EVM 통합 : Qtum에 있는 모든 거래는 단지 Bitcoin과 같이 Bitcoin 스크립팅 언어를 사용한다. 그러나 Qtum에는 세 가지 새로운 opcode가 존재 합니다.

- OP_EXEC : 이 opcode는 트랜잭션의 특수한 프로세스를 촉발시키며(아래에 설명) 또 한 특정 입력 EVM 바이트코드를 실행 합니다.

- OP_EXEC_ASSIGN : 이 opcode는 역시 OP_EXEC와 같이 특수한 프로세스를 촉발시킵니다. 이 opcode에는 계약 주소와 계약에 대한 데이터가 입력되어 있습니다. 다음에는 지정 된 데이터를 전달하는 동안(EVM에 있는 CALLERDATA 로 제공하여) 계약 바이트코드를 실 행 합니다. 이 opcode는 선택적으로 스마트 계약에 화폐를 전송합니다.

- OP_TXHASH : 이 opcode는 계정 추상화 단계의 특이한 부분을 조정하는 데 사용되 며 현재 실행중인 트랜잭션 ID 해시를 푸시합니다.

일반적으로 스크립트는 출력을 소비하려는 경우에만 실행됩니다. 예를 들어 스크립트가 Blockchain에 있는 동안 표준 공개키 해시 트랜잭션에 대하여 유효성 검사 또는 실행이 수행되지 않습니다. 트랜잭션 입력이 출력을 참조할 때까지 실행 및 유효성 검사가 수행되지 않습니다. 이 시점에서 트랜잭션은 입력 스크립트(scriptsig)가 출력 스크립트에 유효한 데이터를 제공하여 '0'이 아닌 결과를 반환하도록 하는 경우에만 유효합니다.

그러나 Qtum은 Blockchain으로 합병될 때 즉시 실행되는 스마트 계약을 수용해야 합니다. 그림 1에 표시된 것처럼 Qtum은 OP_EXEC 또는 OP_EXEC_ASSIGN를 포함 하는 트랜잭션 출력 스크립트 (ScriptPubKey)의 특수 처리를 통해 이를 달성합니다. 이러한 opcode 중 하나가 스크립트에서 발견되면 트랜잭션이 블록에 배치된 후 네트워크의 모든 노드에 의해 실행됩니다. 이 모드에서 실제 비트코인 스크립트 언어는 스크립팅 언어로 덜 사용되며 대신 EVM에 데이터를 전달합니다. EVM은 opcode 중 하나가 실행될 때 자체 상태 데이터베이스 내에서의 상태를 이더리움 계약과 비슷하게 변경합니다.

Qtum 스마트계약 사용을 쉽게 하기 위해, 우리는 특정한 pubkeyhash 주소에서 어간 추출된 작성자뿐만 아니라 스마트 계약에 보내진 자료를 검증해야 합니다.Qtum blockchain의 UTXO 세트가 너무 크게 되지 않도록 하기 위해, OP_EXEC 및 OP_EXEC_ASSIGN 트랜잭션 출력값도 사용할 수 있습니다. OP_EXEC_ASSIGN 출력값은 그들의 코드가 다른 계약이나 pubkeyhash 주소로 자금을 보낼 때 계약에 의해 소비 됩니다. OP_EXEC 출력값은 계약이 blockchain으로부터 스스로를 제거하기 위한 자기제거(suicide) 작업을 시작하면 언제든지 소비됩니다.

Qtum 계정 추상화 계층 : EVM은 계정 기반 blockchain에서 작동 하도록 설계 되었습니다. 그러나 비트코인을 기반으로 한 Qtum은 UTXO 기반 blockchain을 사용하며 가상 머신과 기존의 이더리움 계약에 커다란 수정 없이 EVM을 Qtum blockchain에서 작동하도록 허용하는 계정 추상화 계층(Account Abstraction Layer, AAL)이 포함 되어 있습니다.

EVM 계정 모델은 스마트 계약 프로그래머들이 사용하기에 간단합니다. 블록체인의 현재 계약과 다른 계약의 잔액을 체크할 수 있으며 다른 계약에 자금(데이터에 붙어 있는)을 보내기 위한 연산이 있습니다. 이러한 작업은 매우 기초적이고 간단한 것처럼 보이지만, UTXO 기반

Qtum blockchain 내에서 적용하는 것들은 그리 간단하지 않습니다. 따라서 이러한 작업의 AAL 구현은 예상 보다 복잡할 수 있습니다.
스마트 계약이 배치된 Qtum-blockchain은 해당 주소에 의해 할당되고 호출되며 '0'으로 설정된 새로 배포된 계약 잔액으로 구성됩니다. 현재 Qtum에는 '0'이 아닌 잔액을 사용하여 배포할 수 있는 계약을 허용하는 프로토콜이 없습니다. 계약에 자금을 보내기 위해서 거래는 OP_EXEC_ASSIGN opcode를 사용 합니다.

아래 출력 스크립트 예제에 따라 계약에 돈을 보냅니다.

1; VM 버전
10000; 거래를 위한 gas limit
100; 퀀텀 사토시 gas price
0xF012; 계약에 보내는 데이터(보통 Solidity ABI를 사용)
0x452b22265803b201ac1f8bb25840cb70afe3303 ; ripemd-160 hash of contract txid
OP_EXEC_ASSIGN

위의 간단한 스크립트는 OP_EXEC_ASSIGN opcode 를 통하여 트랜잭션 과정을 처리한다. out-of-gas 와 기타 예외가 발생하지 않는다고 가정할 경우 계약에 지정된 가치 계산은 outputvalue 입니다. gas 메카니즘의 정확한 세부 사항에 대해서는 다음에 논의합니다. 이 출력값을 blockchain 에 추가하면, 출력값은 UTXO 세트를 가지고 있는 계약의 도메인에 들어갑니다. 이 출력값은 계약의 잔액에서 소비할 수 있는 출력값의 합으로 반영 됩니다.

그림 2는 표준 공개키 해시 출력값으로부터 계약에 자금을 보내는 것을 보여 주고 있지만 어떤 계약에서 다른 곳에 자금을 보내기 위한 방법은 거의 동일합니다. 계약이 다른 계약이나 공개키 해시 주소로 자금을 보내는 경우 소유하고 있는 출력값 중 하나를 보냅니다. 보내는 계약은 자금을 보내기 위한 예상 계약 거래(Expected Contract Transactions)를 포함합니다. 이러한 트랜잭션은 Qtum 네트워크가 유효하기 위한 블록에 존재해야 하기에 특별합니다. 예상 계약 거래는 소비자가 생성하는 것이 아니고 트랜잭션을 확인하고 실행하는 동안 채굴자에 의해 생성됩니다. 따라서 P2P 네트워크에서 브로드캐스트 되지 않습니다.

예상 계약 거래를 수행하는 기본 메커니즘은 새로운 opcode인 그림 3과 같이 작동하는 OP_TXHASH입니다. 내부적으로 OP_EXEC 및 OP_EXEC_ASSIGN에는 두 가지 모드가 있습니다. 출력 스크립트 프로세스의 일부가 실행되면 EVM이 실행됩니다. opcode가 입력 스크립트 프로세스의 일부로 실행되면 이중 실행을 방지하기 위해 EVM이 실행되지 않습니다. 대신 OP_EXEC 및 OP_EXEC_ASSIGN opcode는 작동하지 않는 것처럼 작동하며 지정된 트랜잭션 해시를 기반으로 한 '1' 또는 '0' (다시 말하면 제각각 지출할 수 있거나 지출하지 못하거나) 을 반환합니다. 이런 이유로 OP_TXHASH 는 이 개념의 작용에 아주 중요 합니다. 간단하게 설명하면 OP_TXHASH 는 Bitcoin 스크립트 스택에 현재 지출하는 트랜잭션의 SHA256 해시 지원 기능이 추가된 새로운 opcode입니다. 이러한 OP_EXEC 및 OP_EXEC_ASSIGN opcode는 지출을 시도하는 동안 예상 계약 거래의 목록을 확인 합니다.
트랜잭션이 (일반적으로 OP_TXHASH로부터) 예상 계약 트랜잭션 목록에 있는 opcode에 전

달된 후 결과는 '1' 또는 지출 가능 입니다. 그렇지 않으면 결과는 '0' 이거나 지출 불능 입니다. 이러한 방법으로 출력값(vout)를 사용한 OP_EXEC 및 OP_EXEC_ASSIGN 은 계약 및 계정 추상화 단계에서 vout의 지출이 가능함을 요구할 경우 즉, 계약이 자금을 보내려고 시도할 경우에만 지출 가능하다. 허가된 계약이 되기 위한 안전하고 건전한 방법인 이러한 방법은 일반적인 UTXO 거래의 정렬에서 각각의 계약을 통해서만 지출됩니다.

계약에 사용할 수 있는 출력이 두 개 이상인 경우에는 특정 시나리오가 발생합니다. 각 노드는 다른 출력을 선택할 수 있으므로 OP_EXEC_ASSIGN 트랜잭션을 소비하기 위해 완전히 다른 트랜잭션을 사용합니다. 이는 Qtum에서 특별한 코인 획득 알고리즘인 합의를 통해 해결됩니다. 이것은 사용자 지갑 내에서 사용 되는 표준 코인 획득 알고리즘과 유사 합니다. 그러나 Qtum은 서비스 거부 (DoS) 공격 매개체로서의 위험을 회피하고 단순한 합의 규칙을 실현할 수 있도록 알고리즘을 크게 단순화 합니다. 이 특별한 코인 획득 알고리즘인 합의로 인하여 현재 계약에 지출될 다른 코인을 획득하려고 하는 다른 노드를 위한 가능성이 전혀 없습니다. 다른 출력값을 선택한 모든 마이너/노드는 메인 Qtum 네트워크에서 떨어져 포크해야 하며, 그 블록은 무효 처리 됩니다.

그림 4의 EVM 계약이 pubkeyhash 주소 또는 다른 계약에 돈을 보내는 경우 이 이벤트는 새 트랜잭션을 생성합니다. 합의-특별한 코인 획득 알고리즘-는 계약 풀에서 가지고 있는 출력값 중 베스트를 선택합니다. 이러한 출력값은 단일 OP_TXHASH opcode를 구성 하는 입력 스크립트(scriptsig)에 있어서는 입력값으로 쓰인다. 따라서 출력값은 자금의 목적지이며, (필요한 경우) 거래의 잔여 자금을 계약으로 돌려보내기 위한 교환 출력값입니다. 이 트랜잭션 해시는 예상되는 계약 트랜잭션 목록에 추가된 다음, 트랜잭션 자체는 계약 실행 트랜잭션 바로 뒤의 블록에 추가됩니다. 생성된 트랜잭션의 유효성을 검사하고 실행한 후에는 예상 계약 트랜잭션 목록의 확인 검사를 실행합니다. 그런 다음 이 트랜잭션 해시가 예상 계약 트랜잭션 목록에서 제거 됩니다. 이 모델을 사용하면 OP_TXHASH를 사용하는 대신 하드코드 된 해시를 입력 스크립트로 제공함으로써 지출을 위한 트랜잭션을 위장할 수 없습니다.

위에 설명된 추상화 계층은 코인 획득과 특정 출력값을 인식하지 못하는 EVM 계약을 제출합니다. 대신, EVM 계약은 그들과 다른 계약이 잔고를 가지고 있을 뿐이란 걸 알고 있기에 이러한 계약뿐만 아니라 계약 시스템의 외부 pubkeyhash 주소로 자금을 보낼 수 있습니다. 따라서 Qtum과 이더리움 간의 계약 호환성이 강하고, 이더리움 계약을 Qtum blockchain으로 이식하는데 거의 수정이 필요하지 않습니다.
추가 된 표준 트랜잭션 유형 : 다음은 Qtum에 추가한 표준 트랜잭션 유형입니다. 여기에는 Bitcoin 스크립트 견본으로 설명되어 있습니다. blockchain에 새로운 계약을 배치하려면 다음과 같은 출력 스크립트가 필요 합니다.

1 ; VM 버전
[Gas limit]
[Gas price]
[Contract EVM bytecode]
OP_EXEC

blockchain에 이미 배치된 계약에 자금을 보내는 것은 아래의 스크립트를 요구합니다.

1 ; VM 버전
[Gas limit]
[Gas price]
[Data to send to the contract]
[ripemd 160 hash of contract transaction id]
OP_EXEC_ASSIGN

참고로 예상 계약 트랜잭션 목록을 필요로 하는 지출에 대한 표준 트랜잭션 유형은 없습니다. 따라서 이러한 지출 트랜잭션은 P2P 네트워크에서 브로드캐스트 되지도 유효하지도 않습니다.

2.4 Gas Model(가스 모델)

Bitcoin blockchain에 Turing-완전언어-를 추가하는데 있어 Qtum이 직면한 문제는 마이너에 대한 수수료 지급 결정에 대하여 합리적이지 않고 거래의 크기에만 의존한다는 것입니다. 그런 이유로 트랜잭션을 처리하는 마이너들에 의해 하나의 트랜잭션을 무한히 반복하고 전체 blockchain이 중단될 수 있습니다. 그림 5에 표시된 것처럼, Qtum 프로젝트는 이더리움의 gas 개념을 채택한다. gas 개념에는 각 EVM opcode가 실행되는 가격과 각각의 거래에 지출하고자 하는 gas의 총량이 있습니다. 거래 후 남은 gas는 발송인에게 환불 됩니다.

계약 실행에 필요한 가스가 거래에 사용할 수 있는 가스의 양을 초과할 경우 트랜잭션 및 상태 변경 작업은 되돌려 집니다. 따라서 모든 수정된 영구 저장은 계약 자금의 지출을 포함한 원래 상태로 되돌아가서 나중 사람이 지출하지 않도록 해야 합니다. 복귀에도 불구하고, 컴퓨팅 자원이 이미 소모되는 동안 트랜잭션의 모든 가스가 소비되고 이를 처리하는 마이너에게 주어진다.

비록 Qtum이 이더리움의 가스 모델을 사용한다 하여도 우리는 이더리움과 상당한 차이점이 있는 가스 스케줄 즉, 각 EVM opcode의 가스 가격을 기대하고 있다. 정확한 값은 이더리움의 기존 가격과 Qtum의 각 opcode에 필요한 과정 및 blockchain 리소스의 양을 비교하여 결정됩니다.

계약 자금 조달 또는 배치 거래를 만들 때, 사용자는 가스를 위한 2 개의 특정한 아이템을 지정합니다. GasLimit는 계약 체결에 있어 소모할 수 있는 가스량을 결정합니다. 두 번째 아이템은 Qtum Satoshis에 있는 각 단위의 가스의 정확한 가격을 설정하는 GasPrice입니다. GasPrice는 현실적으로 blockchain 기록인 Bitcoin 통화의 더 작은 단위입니다. 계약 실행의 최대 Qtum 지출은 GasPrice에 의한 GasLimit의 곱셈과 동일하다. 이 최대 지출이 거래에 의해 제공되는 거래 수수료를 초과하는 경우에 GasPrice는 유효하지 않고 채굴되거나 처리될 수 없다. 이러한 최대 지출을 차감한 후의 잔여 거래 수수료는 거래 크기 수수료이며 표준 비

트 코인 수수료 모델과 유사하다.

트랜잭션의 적절한 우선순위를 결정하기 위해 마이너는 두 개의 변수를 고려합니다. 첫째, 거래 크기 수수료는 트랜잭션의 총 크기와 일치해야 합니다. 이는 일반적으로 킬로바이트 당 코인 계산의 최소 금액에 의해 결정됩니다. 두 번째 변수는 계약 실행의 GasPrice입니다. 조합하면, POS 마이너는 블록에 포함시키고 처리하기 위해 가장 중요하고 수익성이 거래를 선택합니다. 따라서 마이너와 유저들이 지불하고자 하는 거래 속도와 가격에 맞는 수수료에 최적화된 자유 시장 수수료 모델이 존재하게 됩니다.

환불 : UTXO 모델을 사용하면 자금 거래 수수료로 마이너에게 지급된 자금은 협상이 불가능합니다. 만약 거래가 예상 보다 마이너가 처리하기가 쉽다 하여도 마이너가 부분적으로 수수료를 환불하는 것은 불가능 합니다. 그래서 가스 모델이 유용하게 사용되려면 보낸 사람에게 자금을 다시 환불하는 방법이 존재해야 합니다. 또한 가스가 떨어진 트랜잭션의 상태는 롤백이 가능하여야 하며 가스 수수료는 마이너에게 돌아가야 합니다.

Qtum에서 gas-fee 환불은 마이너에 의한 coinbase 트랜잭션 일부로 새로운 출력값을 생성함으로써 가능할 수 있습니다. 우리는 환불 출력값이 coinbase 트랜잭션에 존재하는 것이 필요한지 확인하기 위해 새로운 블록 유효성 검증 합의 규칙을 추가합니다. 그렇지 않으면, 마이너는 가스를 환불하지 않도록 선택할 수 있습니다. 환불은 출력 스크립트를 복사하여 거래 자금의 발신자에게 다시 주어집니다. 보안상의 이유로 이 스크립트는 현재 표준 pay-to-pubkeyhash 또는 pay-to-scripthash 스크립트입니다. 우리는 더 많은 보안 연구 후에 제한을 해제할 계획 이다.

참고로, OP_EXEC_ASSIGN에는 계약 자금을 배정하기 위해 아래와 같은 방식이 있다:

입력: (in push order)
- 지출에 대한 트랜잭션 해시 [선택 사항]
- 버전 번호 (VM 버전을 사용 하려면, 현재 단지 1개)
- 가스 한도 (이 실행에서 사용할 수 있는 최대 가스 량)
- 가스 비용 (각 가스 단위에 qtum은 얼마인지)
- 데이터 (이 스마트 계약에 전달될 수 있는 데이터)
- 스마트 계약 주소

출력: (in pop order)
- 소비가능 (자금이 현재 소비 가능할 경우)
따라서,우리는 아래와 같은 EXEC_ASSIGN 예제를 제공합니다:
1
10000
100
0x12344 ...
3d6555b14393b55a4dec8ba043bb286afa96af485

EXEC_ASSIGN

VM 실행으로 인해 out-of-gas 예외가 발생 하는 경우, 이 vout(출력값)는 보완 스크립트 OP_TXHASH 를 사용하는 블록의 다음 트랜잭션에 의해 소모됩니다.

이 트랜잭션에 대해 생성된 vout는 vin[0].prevout 스크립트에서 가져온 pubkeyhash 스크립트입니다. 이 초기 버전의 Qtum에서는 VM 자금 거래에 대하여 pubkeyhash 송신자에게만 허용합니다. 다른 양식은 VM 실행을 초래하는 블록으로 허용될 수 있지만, EVM에서 msg.sender는 "0"이며 out-of-gas 또는 gas-refund 는 자금을 유지하는 계약에서 요구됩니다.

부분 환불 모델 : 가스 모델에 관련되어, 또한 몇 가지 이유로 비지출한 부분도 환불이 필요합니다. 한편으로 사용자는 그들의 계약이 적절히 실행될 것을 확실하게 하기 위해 많은 양의 자금을 보낼 수 있다. 그래도 사용하지 않는 가스는 qtum 환불로 반환됩니다.

가스 반환 주소는 전송 트랜잭션의 vin[0].prevout 스크립트로 blockchain에 표현 됩니다. 가스는 표준 비트 코인 거래 수수료 메커니즘을 사용하여 계약으로 보내집니다. 따라서 새로운 수수료 모델은 거래 수수료를 만들기 위해 가스를 조금씩 증가시킵니다.

```
gas fee = gas limit * gas_price
txfee = vin - vout
tx_relay_fee =txfee - gas_fee
refund = gas_fee - used gas
```

마이너가 트랜잭션 우선순위를 결정하기 위해 단일 "credit price" 가치 아래에서tx_relay_fee과 gas_price를 모두 평가할 수 있도록 하는 제안이 존재 합니다.

계약 체결 시, 총 수수료에서 가스 토큰을 , 즉 gas_price를 곱하여 공제합니다. 계약 실행을 마친 후, 이 gas_fee의 나머지는 마이너들이 블록 보상을 검색 하는 데 사용하는 coinbase 트랜잭션에 출력을 추가하여 주어진 가스 반환 스크립트로 반환해야 합니다. vout가 추가된 coinbase는 vin[0].prevout에서 pubkeyhash입니다. 가스 환불을 받으려면 이는 지출된 pubkeyhash vout 이어야 합니다. 그렇지 않으면, 가스 환급은 out-of-gas의 상태에서 마이너와 함께 남아 있으며, 전송된 자금은 계약에 남게 됩니다.
현재 트랜잭션 당 하나의 EVM 계약 실행만 가능 하다는 점에 유의 하십시오. 따라서 트랜잭션 수수료를 공유하려고 시도하는 두 개의 계약이 실행되는 경우가 발생하지 않습니다. 이 시나리오는 트랜잭션 당 여러 EVM 실행을 하는 기존 문제를 해결한 후 사용할 수 있습니다. 현재 디자인은 트랜잭션 당 여러 개의 계약 실행을 지원합니다.

중요한 가스 위기 사례 : 마이너는 contract-gas(계약 가스) 및 fund-return(펀드 반환) 스크립트에 신중해야 합니다. 기금 반환 스크립트 출력으로 인해 블록이 최대 크기를 초과할 경우에는 계약 트랜잭션을 이 블록에 넣을 수 없습니다. 대신에 gas-return(가스 반환) 스크립

트 실행은 다음에 채굴된 블록에서 다시 수행되어야 합니다. 마이너는 계약을 실행하기 전에 gas-return(가스 반환) 스크립트의 후보 블록에 충분한 용량이 존재하는지 확인해야 합니다. 환불 스크립트가 현재 블록에 맞지 않는 경우 이 규칙을 따르고 있지 않으면 반복 실행을 필요로 하는 계약이 발생 합니다. 반환할 가스 자금이 없는 경우, 자금 반환을 위한 vout 요구 사항이 존재 하지 않습니다.

거래 수수료에 gas_fee을 포함하는 것은 합의에 중요 합니다. 거래를 블록에 추가하는 경우, 마이너스 가스 환급이 발생하거나, gas_fee가 거래 수수료 보다 낮을 때 유효하지 않습니다.

둘 이상의 OP_EXEC 또는 OP_EXEC_ASSIGN opcode가 있는 트랜잭션 출력 스크립트는 유효 하지 않습니다. 이렇게 스크립팅 능력을 제한하는 것은 잠재적인 반복 및 다중 실행 문제에 바람직합니다. 따라서 정적인 분석은 스크립트가 잘못 되었는지 여부를 확인하는 데 충분합니다.

blockchain에 매우 적합한 Qtum의 세부사항 뒤에, 우리는 개념적 스마트 계약 주기의 관리에 대하여 설명합니다. 참고로 속편에서 개념적 프리젠테이션은 과학적 문헌[12, 13, 24, 18, 26, 27, 32]에 의해 뒷받침 됩니다.

3. Smart-Contract Management(스마트 컨트랙트 관리)

위의 규정에 따라, 우리는 그 잠재적인 협력 당사자의 적절한 판단이 결정되기 전에 이루어지는 스마트 계약을 확보하기 위해서는 라이프 사이클 관리가 필수적이라고 가정 합니다. 우리는 해산물 배달 실패와 같은 실생활 사례를 참조하여 사업 거래 갈등이 저사양의 전통적인 계약(conventional contract, CC)에서 생겨난다는 것을 알 수 있습니다. EU 회사(구매자)는 남쪽 아시아 회사(판매인)에 오징어 12,920 킬로그램을 주문 한다. CC에서는 제품 품질의 책임은 운송인이 상품을 인수할 때까지 판매인에게 달려 있습니다. 저사양의 CC에 지정되지 않은 상품의 품질에 관련되어 구매자는 선적 회사(운송인)로 양도되기 전에 상품을 확인하지 않습니다.

스마트 계약 대안은 CC에 존재하는 저사양의 충돌을 해결합니다. 따라서 제 3.1에서 제시된 Qtum 프레임 워크 목표 모델은 [18, 26, 27, 32]로 완전히 공식화된 스마트 계약 라이프 사이클의 속성을 반영합니다. 다음으로, 섹션 3.2는 해산물 선적 케이스를 통하여 작은 라이프 사이클 사례를 제공합니다.

3.1 Lifecycle-Management Goals(라이프사이클 관리 목표)

목표를 논의하기 위해 우리는 다음과 같은 방법을 사용 합니다. 에이전트 지향적 모델링(Agent-Oriented Modeling, AOM)기법[38]은 조직에 속한 사람은 문제를 해결하기 위해 협업 기술을 사용하는 것을 고려한다는 사회기술적 요구 엔지니어링 접근법입니다. 이 섹션에서 우리는 Qtum 스마트 계약 시스템 운용 사례를 통하여 중요한 사회기술적 동작 기능을 차지하기 위한 AOM 목표 모델 유형을 사용합니다. 목표 모델은 문제 영역에 대한 이해를 높이기

위해 기술 및 비기술적 이해당사자간의 정보교환을 향상시킵니다. 참고로 AOM 목표 모델은 새로운 민첩한 소프트웨어 개발 기술에도 도움[39]이 됩니다.

목표 모델은 그림 6과 같은 세 가지 주요 요소로 구성 되어 있습니다. 목표라고 부르며 평행사변형으로 묘사되는 기능적 요구 사항, 우리가 골치 아픈 남자로 묘사되는 역할, 그리고 비기능적 요구 사항으로 표시 됩니다. 후자는 두 가지 변종, 즉 구름으로 묘사된 소프트웨어 관련 비 기능적 요구 사항에 대한 품질 목표, 그리고 타원으로 묘사된 인간 관련 정서적 목표를 가지고 있습니다. 목표 모델은 무결성이 아닌 중앙 루트 가치 제안으로 시작 됩니다. 따라서 가치 제안은 트리 계층 구조로 분해되어 각 하위 목표가 상위 목표 [21]달성을 위한 측면을 나타내고 가장 낮은 하위 목표는 무결성을 가져야 하는 하위 목표로 합니다. 목표는 하위 수준 목표에 상속되는 역할, 품질 및 정서적 목표를 할당할 수 있습니다.

Qtum 프레임 워크의 가치 제안: Qtum 프레임 워크에 대한 목표의 루트는 그림 7에 묘사되며 조직 간 정보 및 가치 전송 물류 자동화의 가치 제안입니다. 우리는 스마트 계약 라이프사이클 관리[26, 27, 32]를 위한 목표로 복잡한 가치 제안을 setup(설치), rollout(롤아웃), enactment(입법), rollback(롤백), termination(종료)으로 분할합니다. 이러한 세련된 목표에 대하여 우리는 섹션 3.2에서 더 많은 경험을 하게 됩니다.

그림 7과 같이 업계 도입을 위한 필수적인 정서적 목표는 안정적으로 의도된 동작을 수행할 수 있는 사회기술적 Qtum 시스템[34]의 신뢰입니다. 이런 경우에 신뢰(trust)는 목표를 달성하기 위한 기술을 사용하는 사람의 의존도에 관련됩니다. 우리는 광범위하게 산업 확산에 영향을 미치는 추가적인 정서적 목표로 경제적으로 실용적(economically viable)이며 쉽게 채택(easy to adopt)할 것을 고려합니다. 전자는 투자에 대한 경제 반환에 있어서 Qtum 시스템 결과를 사용하는 것이고 그에 반해 후자는 Qtum 작업에 대한 항목의 진입 장벽이 낮은 것을 의미합니다.

Qtum 시스템의 모든 개선 부분에 영향을 미치는 가치 제안과 관련된 품질 목표가 있습니다. 이러한 품질 목표는 조직 간 비즈니스 프로세스 인식 공동 작업을 위해 참조 아키텍처[28]에서 파생 됩니다. 아래의 품질 목표는 [9, 17]에 따라 구조화 됩니다. 시스템 실행 시간 동안 다음과 같은 품질 목표를 식별할 필요가 없습니다.

수정 가능(Modifiable)이란 Qtum 시스템이 라이프사이클 동안 비즈니스 상황에 맞게 변경되고 조정된다는 의미입니 다. 또한 상용 소프트웨어를 정기적으로 업데이트하는 조직 간 다른 기종 시스템 환경을 조화롭게 합니다. 통합가능(Integrable) 시스템은 구성 요소 간의 인터페이스 프로토콜이 일치해야 하는 별도의 개발 및 통합 구성 요소로 구성됩니다. 그러므로 Qtum의 구성 요소 간의 통합 능력은 확신 되어야 합니다.

다음에는 런타임할 동안 식별 가능한 Qtum에 대한 품질 목표를 지정 합니다. 상호 운용 가능(Interoperable) 이란 Qtum이 계획, 물류, 생산, 외부 파트너 시스템 등과 같은 비즈니스 기능을 지원하는 시스템의 런타임과 상호 운용 되어야 한다는 것입니다. 동적인 상호 운용성 문제는 비즈니스, 개념적, 기술적 이질성입니다. 보안(secure)은 좋은 평판을 가진 신뢰할 수 있

는 사용자에게 서비스를 제공하면서 사용 및 서비스 거부를 무단으로 시도하는 것에 대한 저항을 말합니다. 보안, 신뢰 및 평판 문제를 해결하기 위해, Qtum에 대한 몇 가지 전략이 가능합니다. Blockchain에서 지원하는 인증 서비스는 공동 작업 당사자를 확인하며 네트워크 이벤트 로그를 모니터하고 검사합니다. 시스템의 통신은 암호화 될 수 있습니다. 고도로 자동화된(Highly automated) 협업을 위해서는 시스템이 전체 스마트 계약 라이프사이클을 포함해야 합니다. 따라서 Qtum은 인간이 남아있는 창의적인 행동에 집중할 수 있도록 하면서 지루하고 반복적인 작업을 처리하는데 있어 의미있는 협업 자동화의 높은 수준 가능성을 제공해야 합니다. 유연한(Flexible) 협업은 이질적인 데이터를 교환하는 다양한 파트너에 의한 활동을 규정하는 고도로 동적인 프로세스입니다[25]. 따라서 Qtum은 이질성 개념 및 기술을 조화하는 다양한 교차 조직 협업 시나리오를 활성화해야 합니다. 사용가능(Usable)은 교차 조직 정보 및 물류 자동화를 위해 사용하기 쉬워야 하며 세 개의 분야로 나눠집니다. 오류 회피는 흔히 발생 하는 공동 작업 오류를 예상 및 방지해야 합니다. 오류 처리는 사용자가 오류를 복구할 수 있도록 시스템을 지원합니다. 학습 능력은 Qtum 시스템을 마스터하기 위해 사용자의 필요한 학습 시간을 말합니다.

마지막으로, 아키텍처를 특정하는 품질 목표가 존재합니다. 완전성(Completeness)은 스마트 계약 수명 주기 관리를 위한 구성 요소 집합으로 구성된 Qtum의 품질입니다. 확장성(Scalable) 이란 하나의 구성으로 두 개 이상의 공동 작업을 결합하기 위한 Qtum의 능력을 말합니다. 적용가능성(Applicable)이란 Qtum은 조직 간 정보 물류 및 가치 이전을 자동화 하는 수단이 된다는 것입니다. 이동성(Portable)은 Qtum은 비즈니스, 개념적, 기술적 시스템 인프라와 관련하여 산업 도메인 및 공동 작업에 독립적인 정보 물류를 지원한다는 것입니다. 참고로, 이것은 또한 모바일 장치가 포함되어 있습니다. 성능기준충족(performant)은 전산 및 의사소통이 낮은 정보 물류의 자동화를 의미 합니다. 따라서 스마트 계약 라이프 사이클의 모든 단계는 바람직한 응답 시간 내에 수행되고 컴퓨팅 파워의 기하급수적인 증가가 필요 없다는 것을 보장하는 것이 중요합니다.

3.2 Lifecycle Management Example(라이프사이클 관리 예)

섹션 3.1의 목표 모델은 실행 중인 해산물 케이스에 투영하기 위해 그림 8에 매핑됩니다. 그림 8의 모델링 표기법은 비즈니스 프로세스 모델 및 표기법(business process model and notation, BPMN)[22]이고 전체 주기는 [26, 27, 18]으로 공식화 됩니다. 녹색 원은 주기의 시작 그리고 빨간색 원은 주기 끝을 나타냅니다. + 기호가 있는 사각형은 섹션 3.1의 주기 단계에 해당 하는 소위 하위 프로세스입니다. 하위 프로세스는 하위 레벨의 비즈니스 프로세스 세부 정보를 숨기는 복합 활동입니다.

그림 8의 각 스마트 계약 수명 주기의 출발점은 조직 간 정보 물류 자동화가 필요한 해산물 운송의 비즈니스 사례입니다. 스마트 계약을 시작하기 위한 준비 플랫폼 역할을 하는 공동 작업 허브[29]가 존재 한다고 가정하면 디자이너는 서비스 유형이 역할과 함께 삽입되는 비즈니스 네트워크 모델(business-network model, BNM)용 템플릿을 만듭니다.

BNM 표본이 구성원 단계에 들어갑니다. 각각의 서비스 종류와 관련된 역할은 스마트 계약에

서 협력 하는 조직 즉, bank2, seller, fridge1, carrier, fridge2, buyer ,bank1 으로 채워집니다. 참고로 여러 후보 조직은 특정 역할을 차지하기 위해 경쟁할 수 있습니다. 역할을 채우기 위한 욕구를 강화하기 위해 잠재적인 파트너 조직은 서비스 제안과 역할이 연관된 서비스 유형으로 일치해야 합니다. 서비스 소비자는 제안을 평가하고 서비스 제안이 수용 가능한지 결정할 수 있습니다.

모든 역할이 채워지고 서비스 유형이 수락 가능한 서비스 제공과 일치하는 경우에, 스마트 계약 교섭은 개시됩니다. 우리는 진행중인 해산물 배달 사례에 있어 합의에 이르지 않고 갑작스런 종료를 원하는 당사자가 없다고 가정합니다. 대신 구매자는 해산물이 저장되는 컨테이너 내부의 온도에 관한 의무를 소개하는 수정 제안을 제공합니다. 우리는 선적 컨테이너가 온도 임계값 위반이 발생 했을 때 운송인, 판매자 및 구매자에 게 실시간으로 알려주는 사물 인터넷(Internet-of-Things, IoT) 센서를 갖추고 있다고 가정합니다. 구매자의 수정 제안은 이 사례에서 해산물의 가격 인하는 품질 저하 상태에 따른다고 정의합니다. 만약 온도 변화로 소비를 위해 더 이상 적합하지 않은 해산물이 발생한 경우에, 구매자는 선적 도착 후 구매를 거절할 권리가 있습니다.

수정 제안은 계약 수립을 위한 전제 조건으로 다른 모든 당사자에 의해 수용되고 합의가 형성됩니다. 스마트 계약은 분산된 관리 인프라(distributed governance infrastructure, DGI)를 줄여야 하는 조정 기관입니다. 따라서 실행 중인 사건의 각 당사자는 각각의 의무를 줄이는 로컬 계약 복사본을 받습니다. 예를 들면, carrier를 위한 의무 사항은 해산물 선적 콘테이너 안쪽의 온도가 결코 20도를 초과하여서는 안 된다는 것입니다. 의무 사항은 IOT[15]센서에 연결되어 있는 모니터 및 지정된 비즈니스-네트워크 모델 에이전트(business-network model agents, BNMA)에 의해 관찰됩니다.

다음으로, 모든 협력 당사자들은 신흥 DGI에 각각의 개인 프로세스[12]를 지정할 수 있습니다. 예를 들어, 우리는 buyer가 먼저 유로화로 구매해야만 하는 Bitcoin에 의한 피어-투-피어 지불을 가정합니다. bank1를 통한 구매 및 지불은 정부가 암호화폐의 사용에 관한 규정을 부과하는 준수와 보고 단계로 구성된 프로세스를 수반합니다. seller의 bank2와 bank1 간의 정보 교환을 활성화하기 위해 통신 종료 포인트를 설정해야 합니다. 이렇게 하면 판매자 준수 데이터의 관리가 자동화됩니다.

fridge1의 영역에서 온도 임계값 의무 위반이 발생한다고 가정할 경우, 지정된 BNMA는 이벤트를 확장시키고 buyer는 위반 심각성을 확인합니다. 온도 위반이 일정기간 지속된 결과로 저하된 해산물 품질이 아직도 구매자가 허용하는 저가를 통한 성공적인 판매가 가능하면, 응답은 fridge1 역할에 탈락한 다른 회사에 더 높은 냉각을 요구하는 후자 일지도 모릅니다. 해산물이 심각하게 부패하여 대상 국가에서 판매할 수 없다면, buyer는 거래를 붕괴시켜 파괴적인 롤백을 촉발시킵니다. 해산물 선적은 동의한 국가에 있는 buyer와 bank2를 통해 seller에게 지불이 완료되면, 종료 단계는 DGI를 해산하고 모든 협력 당사자를 풀어 놓는다.

다음은 그림 8의 수명 주기 관리에 대한 세부적인 공동 작업 요소 간의 관계를 설명합니다.

4. Value-Transfer Protocol(가치 전송 프로토콜)

Qtum 프레임 워크의 중요한 부분은 그림 7로 묘사되는 가치 제안과 일치하는 조직간 정보 교류 및 가치 전송을 조정하는 가치 전송 프로토콜 (VTP)의 개념 이다. 섹션 4.1은 VTP를 형성하는 프로세스 유형의 관계를 설명합니다. 섹션 4.2는 유틸리티를 사용하여 VTP를 명시하기 위한 구체적인 스마트 계약 언어에 대한 필요성을 설명합니다. 마지막으로, 섹션 4.3은 VTP 지원 언어 대 이더리움이 사용하는 Solidity 의 기능을 설명합니다.

4.1 Cross-Organizational Processes(상호 조직간 프로세스)

VTP는 세 가지 유형의 협력 프로세스로 구성되어 있습니다. 그림 9는 섹션 3에 소개 된 해산물 배달을 위한 BPMN 표기법으로 나타낸 단순화 된 BNM 이다. BNM 은 하위 프로세스 순서가 조직의 역할을 나타내는 레이블을 사용한 서비스 유형 [12, 13]에 대한 지침이라고 가정합니다.

또한 BNM은 안무 제어 흐름을 구축하기 위하여 서비스 유형 하위 프로세스를 연결하는 작업으로 구성되어 있다고 가정합니다. 간단하게, 그림 9는 AND 분할 및 AND 연결을 가지고 레이블이 없는 안무 작업을 보여준다.

BNM은 국제 통화 거래에 대한 준비를 위해 은행을 알려주는 해산물 판매자와 함께 개시하고, 운송인이 목적지로 가기위해 선적하기 전에 해산물을 냉각합니다. 목적지 국가에서는 현지 은행이 두 국가 간의 통화 거래를 처리 하는 동안 해산물을 다시 냉각 합니다. 마지막으로, 구매자는 지역 판매를 위한 해산물을 수령합니다.

BNM의 carrier 하위 프로세스를 위해, 몇몇 후보자 조직이 해산물 carrier의 역할을 채우기를 위해 존재한다고 가정합니다. 그림 10은 서비스 유형 프로세스 뷰 [12, 13]의 형태로 하위 수준 구체화에 대한 간단한 예제를 보여 줍니다. 그림 10의 단순화 된 프로세스는 캐리어가 원산지 국가에 있는 냉장고에서 해산물을 받고 판매자의 은행에 청구한다고 가정합니다. 다음으로, 3 개의 평행한 가지는 그 온도 감시, 납품 서류 준비 그리고 목적지 회사에 있는 냉각 회사의 안내를 동시에 가져올 것을 요구합니다. 이 단순화 된 절차를 준수할 것을 약속하는 후보자 조직만 서비스 제공 캐리어가 될 수 있습니다. 참고로 공동 작업 허브 [30]는 해당 서비스 제공 조직과 후자를 일치시키기 위하여 서비스 유형 프로세스 뷰를 제공할 수 있습니다.

세 번째 VTP 요소로서, 그림 11은 캐리어가 내부적으로 사용하는 로컬 계약을 보여줍니다. 참고로 그림 10의 서비스 유형 프로세스 뷰 와는 다르게 로컬 계약은 inform buyer와 charge bank2로 구성된 두 개의 추가 작업으로 구성됩니다. 따라서 로컬 계약은 시행 동작과 관련 하여[12, 13] 서비스 유형 프로세스 뷰의 하위 클래스 이다. 다시 말해, 프로세스 뷰의 모든 작업은 외부적으로 발생하지만 캐리어는 경쟁 우위를 구성하거나 외부 디스플레이에 대해 관심이 없는 개인 정보를 보장하는 방식으로 숨겨진 추가 단계를 삽입할 수 있는 옵션을 가지고 있습니다.

4.2 Qtum Smart-Contract Language(Qtum 스마트 계약 언어)

섹션 4.1의 VTP 시나리오를 지원하기 위하여 현재의 스마트 계약 공용어 Solidity는 포함된 개념 및 속성에 관련하여 필수적, 실용적인 수준이 없습니다. 대신, VTP 관리와 비교하여 더 나은 유용성을 가진 Qtum 스마트 계약 언어(Qtum smart contract language, QSCL) 및 컴파일러를 개발하고자 하는 것이 목표입니다.

높은 수준의 QSCL의 개념과 속성은 그림 12에 보여줍니다. 섹션 4.1의 VTP 시나리오는 전용 언어가 존재 하는 eSourcing 프레임 워크, 현재 시맨틱 웹 도메인에 대해 지정되어 있는 eSourcing Markup Language (eSML)[31]와 비슷합니다. 우리는 독창적인 Qtum 가상 머신에 대한 언어 컴파일러와 함께 QSCL을 만들기 위하여 blockchain 영역에 eSML의 개념과 속성을 매핑할 예정입니다.

간단히 말하면 추가적인 세부사항을 위해 독자에게 eSML을 찾아보게 하는 동안 그림 12의 속성으로 개념적 질의형식에 따라 구성됩니다. 하나의 QSCL 사례는 BNM 정의와 유사 합니다 (그림9). QSCL의 Who 개념은 관련 리소스 및 데이터 정의와 함께 계약 당사자를 고유하게 정의하기 위한 구조를 구성합니다. Where 개념은 특정 스마트 계약이 보유 하는 비즈니스 상황 및 법적 상황 공급을 지정 합니다.What 개념은 이러한 프로세스 뷰 및 기본 작업에 대한 라이프사이클 정의와 함께 교환된 가치와 서비스 유형 프로세스 뷰 (그림 10)에 대한 정의를 참작합니다. 따라서 QSCL 사례의 What 부분에서 그림 9와 비교하여 여러 서비스 유형 프로세스 뷰를 정의할 수 있습니다. 마지막으로, conjoinment(결합) 구조는 조직 간 데이터 흐름를 위해 특별히 정의된 교환 채널입니다. Monitorability(모니터링 가능) 구조는 폴링 또는 메시징 원칙을 사용하는 전용 작업 모니터링에 대한 유연한 정의를 허용 합니다.

4.3 Comparative Discussion(비교 토론)

스마트 계약 온톨로지(존재론)[31]를 사용하여, 우리는 비공식적으로 Qtum 프레임 워크를 위해 구축한 QSCL 대 기존의 Solidity의 적합성을 검사 합니다. 일반적인 관찰로서, Solidity는 자바 스크립트와 유사한 구문을 사용하여 주로 낮은 수준의 Blockchain 조작 명령에 초점을 맞춘 문법과 언어입니다. 그러나 타사 API를 가져오고 외부 함수 호출을 수행할 수도 있습니다. Solidity에서 소위 외부 함수는 다른 계약과 거래를 통해 호출할 수 있는 스마트 계약 인터페이스의 일부입니다.

Solidity의 튜링 완전언어 때문에, 그것은 QSCL이 구체화하는 스마트 계약 온톨로지의 모든 개념과 속성에 대해 번거로운 지원을 정의하기 위해 원칙적으로 가능 합니다. 그러나 패턴 기반 설계, 프로세스 인식, 프로세스 매칭 등의 개념은 Solidity에서 어떤 식으로든 채택되지 않습니다. 번거로운 해결 방법 발견에 관하여, 최근 학회 간행물[43]은 스마트 계약에서 신뢰할 수 없는 비즈니스 프로세스의 점검 및 실행의 타당성을 입증하기 위해 Solidity를 사용 합니다.

Solidity의 가장 강조할 점 하나는 QSCL[31]의 디자인 초기와는 다르게 역사적으로 공식적인

검증 수단에 의해 뒷받침 되지 않았습니다. 그러한 공식적으로 검증할 수 있는 표현이 없다는 것은 계약이 정확하고 안전 문제에서 자유로울 경우 체결에 앞서서 아는 것은 불가능합니다. Solidity와 관련된 보안 사건은 최근 Why, Solidifier, 또는 Casper 와 같은 검증 도구의 개발과 적용을 촉발시켰으며 이더리움 전반에 걸쳐 proof-of-work에서 proof-of-stake으로 전환하도록 유도합니다.

5. Conclusions(결론)

이 백서는 새로운 스마트 계약 블록 체인 기술 솔루션을 위한 Qtum 프레임 워크를 제공합니다. 우리는 지분 증명 유효성 검사를 사용하는 특별한 Qtum 트랜잭션 프로세스 구현을 보여줍니다. 또한, Qtum은 이더리움 가상 머신(EVM)을 비트 코인의 비 소비 트랜잭션 출력 프로토콜과 함께 통합 합니다. 참고로 Qtum EVM은 끊임없이 이전 버전과 호환됩니다. 또한 Qtum 프레임 워크는 스마트 계약 수명 주기 관리가 협업 당사자의 적절한 보안 검증을 지원하는 데 중요하다는 것을 인식합니다. Qtum 라이프 사이클 관리를 지원하기 위해 현재 공용어인 Solidity 는 적합성이 부족합니다. 따라서 새로운 Qtum 프레임 워크는 향상된 유틸리티와 새로운 스마트 계약 언어를 필요로 합니다.

Qtum에서 proof-of-stake 적용은 아직도 proof-of-work를 적용하고 확장하지 않은 이더리움 대안으로써 계산하는데 따른 노력의 상당한 절약을 이끌어 냅니다. 이더리움도 또한 proof-of-stake 채택을 계획하고 있지만 그런 새로운 버전이 공개 될지는 불분명합니다. 또한 비 지출 된 트랜잭션 출력값의 사용은 이더리움의 계정 관리에 비교하여 확장 가능성이 더 높습니다. 단순한 지불 검증과 결합하여 Qtum은 이미 스마트 계약 모바일 장치 솔루션을 개발하고 있습니다. 확장하지 않은 이더리움 솔루션은 모바일 솔루션을 허용하지 않지만, Qtum은 모바일 전략으로 민주화와 고도로 분산된 proof-of-stake 거래 검증을 달성하는 것을 목표로 합니다.

Qtum 프레임 워크는 미래의 개발이 충족해야 하는 품질 기준을 명확하게 이해하고 있습니다. 기능적 요구 사항과 관련해, Qtum은 스마트 계약 수명 주기 관리를 위한 애플리케이션 계층을 개발할 계획입니다. 라이프사이클 관리는 이더리움이 최근 경험한, 나중에 다양한 하드포크가 생겨나는 보안 위반을 줄이기 위해 협력 당사자를 점검하는데 중요합니다.

Qtum의 정보 교류를 위한 가치 전송 프로토콜은 여러 협업 조직을 위한 비즈니스 네트워크 모델로 구성 되어 있습니다. 협업 조직은 비즈니스 네트워크 모델에서 서비스 유형 프로세스 뷰의 지정된 런타임 동작과 일치해야 하는 로컬 계약을 서비스에 제공할 수 있습니다. 멀티 계층화 된 스마트 계약 관리 계층을 사용하여 협력 당사자들의 비즈니스 비밀의 개인 정보를 보호합니다. 그들은 로컬 계약의 확장 단계를 숨김으로써 경쟁 우위에 자리 잡게 됩니다.

요약 하자면, Qtum 프레임 워크는 스마트 계약이 광범위한 사용자 채택을 달성하기 위한 필수 품질 요구 사항을 고려해야 하는 사회기술적 산물인 것을 인식 합니다. Qtum 응용 프로그램을 통한 지속적인 실제 산업 프로젝트는 지속적인 경험적 요구 사항을 수확하는 결과를 가져 옵니다. 고도로 분산된 지분 증명 거래 처리를 지원하는 모바일 전략은 최첨단 기술의 발

전을 목표로 합니다. 또한, Qtum은 스마트 계약 수명 주기 관리를 위해서는 현재 솔루션이 충분히 주의를 기울이지 않는 정교한 프런트 엔드 사용자 경험으로 응용 프로그램 계층 개발을 필요로 한다는 것도 인식 합니다.