

멀티버스엑스(MultiversX EGLD)

해당 번역본은 이용자가 디지털자산 관련 정보를 편리하게 확인할 수 있도록
단순 참고용으로 번역된 것이며, 투자 권유를 목적으로 하지 않습니다.
그 내용이 정확하지 않을 수 있으므로 정확한 정보 습득을 위해서는
원문을 참고하시거나 원문 작성 주체 측에 문의하시기 바랍니다.
두나무는 해당 번역본 내용의 진실성 및 정확성을 보장하지 않고,
제공된 정보에 의한 투자 결과에 대하여 어떠한 책임도 지지 않으며,
투자 손실은 투자자에게 귀속됩니다

MultiversX

적응형 상태 샤딩과 Secure Proof of Stake를 통한 고확장성 퍼블릭 블록체인

[기술 백서 — 릴리스 2 — 리비전 2]

업데이트: MultiversX로의 명칭 변경

2019년 6월 19일 - MultiversX 팀

<https://www.multiversx.com/>

초록

비트코인과 이후 이더리움에 의해 안전한 퍼블릭 블록체인이 등장하면서, 상당한 관심과 자본 유입이 촉발되어 전 세계적인 무허가 혁신의 물결을 위한 토대가 마련되었습니다. 그러나 탈중앙화·보안·확장성을 갖춘 퍼블릭 블록체인을 만드는 일은 매우 고된 과제가 입증되었습니다.

본 논문은 실용적 확장성을 위해 진정한 상태 샤딩 방식을 도입하고, 에너지와 계산 낭비를 제거하며, Secure Proof of Stake(SPoS) 합의를 통해 분산적 공정성을 보장함으로써 현 기술 수준을 넘어서는 새로운 아키텍처인 MultiversX를 제안합니다. 보안에 강하게 초점을 맞추어, MultiversX 네트워크는 시빌 공격, Nothing at Stake 공격 등 알려진 보안 문제에 대한 저항성을 확보하도록 설계되었습니다. 상호연결성을 지향하는 생태계에서, 우리의 스마트 컨트랙트 솔루션은 설계 단계에서 상호운용성을 보장하기 위해 EVM 호환 엔진을 제공합니다.

예비 시뮬레이션과 테스트넷 결과는 MultiversX가 비자의 평균 처리량을 초과하고, 기존의 실행 가능한 접근법 대비 세 자릿수(1000배)가 넘는 개선을 달성함을 보여주며, 장기적 지속가능성을 보장하기 위해 부트스트래핑 및 저장 비용을 대폭 줄입니다.

I 서론

1 일반적 측면

비트코인과 이더리움과 같은 암호화폐 및 스마트 컨트랙트 플랫폼은 상당한 관심을 불러일으켰으며, 전자 결제, 탈중앙화 애플리케이션 및 잠재적 디지털 가치 저장 수단을 위한 유망한 해결책이 되었습니다. 그러나 핵심 지표에서 중앙화된 대응물과 비교할 때 [1], 현재의 상황은 현존 퍼블릭 블록체인의 반복들이 특히 확장성과 관련해 심각한 한계를 보이며, 이로 인해 대중적 채택이 저해되고 공공 사용이 지연되고 있음을 시사합니다. 사실, 블록체인 트릴레마 패러다임의 트레이드오프가 부과하는 현재의 엔지니어링 경계를 다루는 일은 극도로 도전적임이 입증되었습니다 [2]. 여러 해결책이 제안되었으나, 그중 의미 있고 실행 가능한 결과를 보여준 것은 많지 않습니다. 따라서 확장성 문제를 해결하기 위해서는 퍼블릭 블록체인 인프라의 전면적 재고가 필요했습니다.

2 과제 정의

확장을 위해 설계된 혁신적 퍼블릭 블록체인 솔루션을 만드는 과정에서 여러 과제를 적절히 해결해야 합니다:

- **완전한 탈중앙화:** 신뢰할 수 있는 제3자의 필요를 제거하여 단일 장애 지점을 없앴.
- **강인한 보안:** 안전한 트랜잭션을 허용하고, 알려진 공격 벡터를 기반으로 한 모든 공격을 방지함.
- **높은 확장성:** TPS로 측정했을 때 중앙화된 대응물과 최소한 동등한 성능을 네트워크가 달성하도록 함.
- **효율성:** 최소한의 에너지와 계산 요구로 모든 네트워크 서비스를 수행함.
- **부트스트래핑 및 저장 향상:** 데이터 저장 및 동기화 비용에서 경쟁력을 확보함.
- **크로스체인 상호운용성:** 설계로써 강제되어, 외부 서비스와의 무제한 통신을 허용함.

이러한 과제를 바탕으로, 우리는 MultiversX를 안전하고 효율적이며 확장 가능하고 상호운용 가능한 퍼블릭 블록체인 인프라의 전면 재고로써 만들었습니다. MultiversX의 주요 기여는 두 가지 초석적 구성 요소에 놓여 있습니다:

1. 진정한 상태 샤딩(State Sharding) 접근: 블록체인과 계정 상태를 여러 샤드로 효과적으로 분할하고, 서로 다른 참여 검증자들이 병렬로 처리함.
2. Secure Proof of Stake 합의 메커니즘: 에너지 집약적 PoW 알고리즘의 필요를 제거하면서, 장기 보안과 분산적 공정성을 보장하는 PoS의 개선 변형.

3 적응형 상태 샤딩

MultiversX는 필요와 활성 네트워크 노드 수에 따라 샤드 계산과 재조직을 가능하게 하는 동적으로 적응형 샤딩 메커니즘을 제안합니다. 각 에폭의 시작 시 샤드 내 노드의 재할당은 점진적이고 비결정적이며, 일시적인 생존성(liveness) 페널티를 유발하지 않습니다. 적응형 상태 샤딩은 정적 모델과 비교해 추가적인 도전을 동반합니다. 핵심 포인트 중 하나는 샤드 수가 변할 때 동기화-통신 필요성으로 인해 도입되는 전체 지연 페널티를 방지하기 위해 샤드 분할과 병합이 어떻게 수행되는가에 있습니다. 이때 지연(latency)은, 자신의 샤드 주소 공간 할당이 변경된 이후 새로운 상태를 가져오기 위해 노드가 요구하는 통신 오버헤드를 의미합니다.

MultiversX는 이 문제를 다음과 같이 해결합니다:

1. 계정 주소 공간을 샤드로 분할하는데, 이는 특정 에폭의 정확한 샤드 수만 알면 구축 가능한 이진 트리를 사용합니다. 이 방법을 통해 누적 지연이 줄어들고 네트워크 생존성이 두 가지 방식으로 개선됩니다. 첫째, 설계된 모델 덕분에 계정 주소 공간의 분할이 계층에 의해 미리 결정됩니다. 따라서 분할 오버헤드가 없으며, 하나의 샤드는 두 개의 샤드로 나뉘고 각 샤드는 이전 주소 공간의 절반과 그에 연관된 상태만 유지합니다. 둘째, 상태 중복 메커니즘을 통해 병합이 형제 노드에 의해 준비되므로 지연이 감소합니다.
2. 각 샤드 내 노드의 균형을 맞추어 전체 아키텍처의 균형을 달성하는 기법을 도입합니다. 이 기법은 네트워크의 각 노드에 균형 잡힌 작업 부하와 보상을 보장합니다.
3. 트랜잭션을 해당 샤드로 자동 라우팅하는 내장 메커니즘을 설계하여 결과적으로 지연을 크게 줄입니다. 이 라우팅 알고리즘은 IV.4절 – MultiversX 샤딩 접근에서 설명됩니다.
4. 부트스트래핑과 저장 관점에서 상당한 개선을 달성하기 위해, MultiversX는 샤드 프루닝(pruning) 메커니즘을 사용합니다. 이를 통해 초당 수만 건의 트랜잭션 처리량(TPS)에서도 아키텍처의 지속가능성이 보장됩니다.

4 Secure Proof of Stake (SPoS)

우리는 Algorand [3]의 무작위 선택 메커니즘 아이디어를 확장한 Secure Proof of Stake 합의 메커니즘을 소개합니다. 이 메커니즘은 다음과 같은 측면에서 차별화됩니다:

1. MultiversX는 지연을 줄여 샤드 내 각 노드가 라운드 시작 시 합의 그룹(블록 제안자와 검증자)을 결정할 수 있게 하는 개선을 도입합니다. 이는 매 블록에 난수 인자 r 이 저장되며, 블록 제안자가 이전 r 에 대해 BLS 서명 [4]을 사용하여 이를 생성하기 때문에 가능합니다.
2. 블록 제안자는 공개키와 난수 인자의 해시가 가장 작은 합의 그룹 내 검증자입니다. Algorand [3]에서는 무작위 위원회 선택이 최대 12초가 소요될 수 있으나, MultiversX에서는 네트워크 지연을 제외하면 합의 그룹 무작위 선택 시간이 현저히 줄어(추정 100ms 미만)집니다. 이 과정에서 통신 요구가 전혀 없으므로, MultiversX는 매 라운드마다 새롭고 무작위로 선택된 그룹이 원장에 새 블록을 성공적으로 기록할 수 있습니다. 이 개선의 트레이드오프는, 공격자가 라운드 시간 프레임보다 빠르게 적응할 수 없으며 블록 제안을 하지 않을 수 있다는 전제에 의존합니다. 무작위성 소스의 보안을 더욱 강화하는 방법으로는 검증 가능한 지연 함수(VDFs)를 사용하는 것이 있으며, 이는 무작위성 소스를 변조할 가능성을 낮출 수 있습니다. 다만 VDF에 대한 연구는 아직 진행 중이며, 소수의 실험적 구현만 존재합니다.
3. PoS 아키텍처에서 일반적으로 사용되는 스테이크 요인 외에, MultiversX는 rating이라 불리는 추가 가중치를 도입해 합의 메커니즘을 정교화합니다. 노드가 합의 그룹에 선택될 확률은 스테이크와 rating 모두를 고려합니다. 블록 제안자의 rating은 각 에폭 종료 시 재계산되며, 슬래싱이 발생할 경우에는 즉시 감소하여 공정성을 강화합니다.
4. 합의 그룹은 블록 서명을 위해 2회 통신 라운드를 사용하는 수정된 BLS 다중서명 기법 [5]을 사용합니다.
5. MultiversX는 SPoS 합의 메커니즘과 같은 핵심 프로토콜 구현에 대해 알고리즘의 정확성을 검증하기 위해 형식적 검증(formal verification)을 고려합니다.

II 아키텍처 개요

1 엔티티(Entities)

MultiversX에는 두 가지 주요 엔티티가 있습니다: 사용자와 노드입니다. 사용자는 각각 하나 이상의 공개/개인키 쌍(Pk/sk)을 보유하며(예: 하나 또는 여러 개의 지갑 앱), MultiversX 네트워크를 사용하여 서명된 트랜잭션을 전송해 가치 이전 또는 스마트 컨트랙트 실행을 수행합니다. 사용자는 자신의 계정 주소(공개 키로부터 파생됨) 중 하나로 식별될 수 있습니다. 노드는 MultiversX 네트워크를 형성하는 디바이스로서, 수동적일 수도 있고 처리 작업에 적극적으로 참여할 수도 있습니다. 자격을 갖춘 검증자(eligible validator)는 MultiversX 네트워크의 적극적 참여자입니다. 이들은 합의를 실행하고, 블록을 추가하며, 상태를 유지하고, 그 기여에 대해 보상을 받는 역할을 담당합니다. 각 자격 검증자는 필요한 스테이크를 예치한 주소와 노드 ID로부터 파생된 공개키를 통해 고유하게 식별됩니다. MultiversX 프로토콜 내 엔티티 간의 관계는 그림 1과 같습니다.

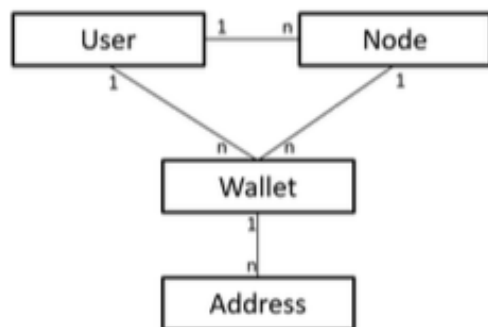


그림 1: MultiversX 엔티티 간의 관계

더 나아가 네트워크는 샤드라 불리는 더 작은 단위로 나뉩니다. 자격 검증자는 알고리즘에 의해 샤드에 배정되는데, 이는 트리 레벨에 따라 노드들이 샤드 전반에 고르게 분포되도록 보장합니다. 각 샤드에는 무작위로 선택된 합의 그룹이 포함됩니다. 어떤 블록 제안자는 트랜잭션을 모아 새로운 블록을 구성할 책임이 있습니다. 검증자들은 제안된 블록을 승인하거나 거부하며, 이를 통해 블록을 검증하고 블록체인에 기록합니다.

2 내재 토큰(Intrinsic token)

MultiversX는 eGold(약칭 EGLD)라 불리는 내재 유틸리티 토큰을 통해 네트워크 사용 권한을 부여합니다. 트랜잭션 처리, 스마트 컨트랙트 실행, 네트워크에 대한 다양한 기여에 대한 보상은 모두 EGLD로 지불됩니다. 수수료, 지불, 잔액에 대한 언급은 모두 EGLD 기준으로 간주됩니다.

3 위협 모델(Threat model)

MultiversX는 비잔틴 적대 모델을 가정합니다. 여기서 샤드 내 자격 검증자 중 최소 $\frac{2}{3}n + 1$ 노드는 정직하다고 가정합니다. 프로토콜은 지분을 보유하거나 높은 평판을 가진 적대자, 메시지 지연 또는 충돌 메시지 전송, 다른 노드 침해, 버그 보유 또는 담합하는 적대자의 존재를 허용합니다. 그러나 샤드 내 자격 검증자 중 $\frac{2}{3}n + 1$ 이상이 정직하거나 침해되지 않았다면, 프로토콜은 합의를 달성할 수 있습니다.

프로토콜은 고도로 적응적인 적대자를 가정하지만, 그들은 라운드 시간 프레임보다 빠르게 적응할 수 없습니다. 적대자의 계산 능력은 한정되어 있으므로, 선택된 암호학적 원시의 보안 수준에 의해 보장되는 가정은 다항 시간 내 튜링 기계로 풀 수 있는 문제의 복잡도 클래스 내에서 확고히 유지됩니다. 정직한 노드들의 네트워크는 잘 연결된 그래프를 형성한다고 가정하며, 그 메시지 전파는 제한된 시간 Δ 안에 이루어집니다.

공격 벡터 예방

1. **시빌 공격**: 네트워크 참여 시 스테이크를 잠금으로써 완화됩니다. 이렇게 하면 새로운 ID 생성을 위한 비용이 최소 스테이크와 동일해집니다.
2. **Nothing at Stake**: 제안자뿐 아니라 다중 서명이 필요하고, 스테이크 슬래싱이 있기 때문에 제거됩니다. 블록당 보상과 잠금된 스테이크의 비율은 그러한 행위를 억제합니다.
3. **롱레인지 공격**: 프루닝 메커니즘, 매 라운드마다 무작위로 선택되는 합의 그룹(단일 제안자가 아님), 스테이크 잠금을 통해 완화됩니다. 게다가 pBFT 합의 알고리즘은 최종성을 보장합니다.
4. **DDoS 공격**: 합의 그룹이 매 라운드(수 초 단위)마다 무작위로 샘플링되므로, 짧은 시간 프레임으로 인해 DDoS는 거의 불가능합니다.

우리가 고려한 다른 공격 벡터에는 샤드 점령 공격, 트랜잭션 검열, 이중지불, 뇌물 공격 등이 있습니다.

4 연대기(Chronology)

MultiversX 네트워크에서 타임라인은 에폭과 라운드로 나뉩니다. 에폭은 고정된 기간(하루로 설정되며 아키텍처 발전에 따라 수정 가능)을 가지며, 종료 시 샤드 재조직과 프루닝이 발생합니다. 에폭은 다시 라운드로 나뉘며, 일정한 시간 동안 지속됩니다. 각 라운드마다 샤드 내 무작위로 선택된 합의 그룹이 한 개의 블록을 원장에 기록할 수 있습니다.

새로운 검증자는 스테이크를 잠금으로써 네트워크에 참여할 수 있으며, 이는 V.2 Secure Proof of Stake 장에서 설명됩니다. 이들은 현재 에폭 e 의 미할당 노드 풀에 추가되고, 에폭 $e + 1$ 시작 시 샤드 대기 리스트에 배정되며, 다음 에폭 $e + 2$ 에서야 합의에 참여하고 보상을 받을 수 있는 자격 검증자가 됩니다.

타임라인에 대한 세부 사항은 IX.1절에서 더 자세히 설명됩니다.

III 관련 연구

MultiversX는 Ethereum [6], Omniledger [7], Zilliqa [8], Algorand [3], ChainSpace [9]의 아이디어에 기반하고 영감을 받아 설계되었습니다. 우리의 아키텍처는 현존 기술 수준을 넘어서는 것으로, 기존 모델의 확장판이라 볼 수 있으며, 성능을 개선하는 동시에 보안, 확장성, 탈중앙화 간의 더 나은 내쉬 균형 상태를 달성하는 것을 목표로 합니다.

1 Ethereum

Ethereum [6]의 성공은 EVM [10], Solidity [11], Web3j [12]를 통한 탈중앙화 애플리케이션 레이어의 도입에 크게 기인합니다. Dapp은 Ethereum의 핵심 기능 중 하나였으나, 확장성은 심각한 제약으로 드러났습니다. 이 문제를 해결하기 위한 연구가 상당히 진행되었지만, 현재까지 결과는 미미했습니다. 다만 몇 가지 유망한 개선안이 제안되고 있습니다. Casper [13]는 현행 PoW 합의를 PoS로 교체할 업데이트를 준비하고 있으며, Plasma 기반 사이드체인과 샤딩은 가까운 미래에 제공될 것으로 기대되어 Ethereum의 확장성 문제를 부분적으로나마 완화할 것으로 예상됩니다 [14].

Ethereum과 비교했을 때, MultiversX는 샤딩을 통한 트랜잭션 처리 병렬성을 사용하면서 SPoS 합의를 구현하여 PoW 알고리즘에서 발생하는 에너지 및 계산 낭비를 제거합니다.

2 Omniledger

Omniledger [7]는 무허가 환경에서도 장기적 보안을 보장하는 새로운 스케일아웃 분산 원장을 제안합니다. 이는 트랜잭션을 처리하는 크고 통계적으로 대표성 있는 샤드를 선택하기 위해 편향 저항적 공개 난수 프로토콜을 사용함으로써 보안성과 정확성을 확보합니다. 샤드 간 트랜잭션을 원자적으로 커밋하기 위해 Omniledger는 Atomix라는 효율적 크로스샤드 커밋 프로토콜을 도입합니다. 이는 두 단계의 클라이언트 주도 “lock/unlock” 프로토콜 개념으로, 노드가 트랜잭션을 샤드 전반에 완전히 커밋하거나, 거부 증명을 얻어 부분적으로 완료된 트랜잭션에 의해 영향을 받은 상태를 취소하고 해제할 수 있도록 합니다. Omniledger는 또한 병렬 샤드 내 트랜잭션 처리, 집단 서명된 상태 블록을 통한 원장 프루닝, 저가치 트랜잭션에 대한 저지연 “trust-but-verify” 검증을 통해 성능을 최적화합니다. Omniledger의 합의는 ByzCoinX라 불리는 BFT 변형으로, 성능과 DoS 공격에 대한 강건성을 높입니다.

Omniledger와 비교했을 때, MultiversX는 상태 샤딩에 대해 적응적 접근을 취하며, 더 빠른 합의 그룹 무작위 선택을 지원하고, 매 라운드(수 초 단위)마다 검증자 집합을 교체함으로써(매 에폭, 즉 하루 단위가 아님) 보안을 개선합니다.

3 Zilliqa

Zilliqa [8]는 최초의 트랜잭션 샤딩 아키텍처로, 채굴 네트워크를 샤드로 나누어 병렬로 트랜잭션을 처리하고 높은 처리량을 달성할 수 있게 합니다. 특히, 이 설계는 네트워크에 더 많은 노드가 참여할수록 트랜잭션 처리율이 높아지도록 합니다. 핵심은 샤드가 서로 다른 트랜잭션을 중복 없이 처리하게 하여 이중지불을 방지하는 데 있습니다. Zilliqa는 신원 확립과 시빌 공격 방지를 위해 PoW를 사용하며, 합의에는 pBFT [15]를 사용합니다.

Zilliqa와 비교했을 때, MultiversX는 트랜잭션 샤딩뿐만 아니라 상태 샤딩도 사용하여 샤딩의 한계를 더욱 확장합니다. MultiversX는 PoW 메커니즘을 완전히 제거하고 SPoS를 사용합니다. 두 아키텍처 모두 자체 스마트 컨트랙트 엔진을 구축하지만, MultiversX는 단순히 EVM 호환성을 목표로 하는 데 그치지 않고, Ethereum용으로 작성된 SC가 우리 VM에서 원활히 실행되도록 하며, 더 나아가 블록체인 간 상호운용성을 달성하는 것을 목표로 합니다.

4 Algorand

Algorand [3]는 중앙화 시스템의 편의성과 효율성을 유지하면서도, 기존 탈중앙화 구현의 비효율성과 약점을 극복하는 공용 원장을 제안합니다. 리더와 검증자 집합은 마지막 블록의 값에 서명한 결과를 기반으로 무작위로 선택됩니다. 이 선택은 조작에 면역이며 마지막 순간까지 예측할 수 없습니다. 합의는 새로운 메시지 전달 기반 비잔틴 합의를 기반으로 하며, 이를 통해 커뮤니티와 프로토콜이 하드포크 없이 발전할 수 있습니다.

Algorand와 비교했을 때, MultiversX는 단일 블록체인을 갖지 않고, 샤딩을 통해 트랜잭션 처리량을 증가시킵니다. 또한 MultiversX는 합의 그룹 선택 시간을 12초 이상에서 1초 미만으로 단축함으로써 Algorand의 무작위 선택 아이디어를 개선하지만, 적대자가 라운드 내에서 적응할 수 없다는 가정을 둡니다.

5 Chainspace

Chainspace [9]는 트랜잭션의 고무결성과 투명한 처리를 위한 분산 원장 플랫폼입니다. 이는 언어에 구애받지 않고 프라이버시 친화적인 스마트 계약을 사용해 확장성을 확보합니다. 샤딩된 아키텍처는 S-BAC이라 불리는 새로운 분산 원자적 커밋 프로토콜을 통해 일관성을 보장하고 높은 감사 가능성을 제공하면서, 선형적으로 확장 가능한 트랜잭션 처리량을 지원합니다. 프라이버시 기능은 최신 영지식 기법을 통해 구현되며, 합의는 BFT로 보장됩니다.

Chainspace에서는 샤드 내 노드 수가 늘어날수록 TPS가 감소하지만, MultiversX의 접근법은 합의 그룹 크기가 고정되어 있기 때문에 샤드 내 노드 수에 영향을 받지 않습니다. Chainspace의 강점은 언어에 구애받지 않는 스마트 계약 접근법에 있지만, MultiversX는 EVM 호환성을 위한 추상화 계층 구축에 중점을 둡니다. 두 프로젝트는 성능 향상을 위해 각기 다른 상태 샤딩 접근법을 사용합니다. 그러나 MultiversX는 높은 처리량 아키텍처에서 발생할 블록체인 크기 문제를 예견하고, 효율적인 프루닝 메커니즘을 사용함으로써 한 발 더 나아갑니다. 또한 MultiversX는 샤드 중복성을 도입하여, 갑작스러운 노드 수 변화와 악의적 샤드 점령에 대한 더 높은 저항성을 제공합니다. 이는 샤딩 블록체인에 새롭게 도입된 기능입니다.

IV 적응형 상태 샤딩을 통한 확장성

1 왜 샤딩인가

샤딩은 처음에는 데이터베이스에서 사용되었으며, 데이터를 여러 기계에 분산시키는 방법입니다. 이 확장 기법은 블록체인에 적용되어 상태와 트랜잭션 처리를 분할할 수 있으며, 그 결과 각 노드는 전체 트랜잭션 중 일부만 다른 노드와 병렬로 처리하게 됩니다. 충분한 수의 노드가 각 트랜잭션을 검증하여 시스템이 높은 신뢰성과 보안을 유지한다면, 블록체인을 샤드로 분할하는 것은 다수의 트랜잭션을 병렬로 처리하게 하여 트랜잭션 처리량과 효율성을 크게 개선할 수 있습니다. 샤딩은 검증자 네트워크가 확장될수록 처리량을 증가시킬 수 있으며, 이는 수평적 확장(horizontal scaling)이라 불리는 성질입니다.

2 샤딩의 유형

포괄적이고 철저한 소개 [16]에서는 세 가지 주요 샤딩 유형을 강조합니다: 네트워크 샤딩, 트랜잭션 샤딩, 상태 샤딩입니다.

- **네트워크 샤딩**은 노드를 샤드로 그룹화하는 방식을 다루며, 통신을 최적화하는 데 사용될 수 있습니다. 샤드 내에서의 메시지 전파는 전체 네트워크로의 전파보다 훨씬 빠르게 수행될 수 있습니다. 이는 모든 샤딩 접근에서 첫 번째 도전 과제이며, 노드를 샤드로 매핑하는 메커니즘은 특정 샤드를 장악한 공격자를 고려해야 합니다.
- **트랜잭션 샤딩**은 트랜잭션이 어떤 샤드에서 처리될지를 결정하는 방식입니다. 계정 기반 시스템에서는 송신자의 주소를 기준으로 트랜잭션을 샤드에 할당할 수 있습니다.
- **상태 샤딩**은 가장 어려운 접근 방식입니다. 앞서 설명한 메커니즘에서는 모든 노드가 전체 상태를 저장하지만, 상태 샤딩된 블록체인에서는 각 샤드가 상태의 일부만 유지합니다. 서로 다른 샤드에 속한 계정을 다루는 모든 트랜잭션은 메시지를 교환하고 서로 다른 샤드의 상태를 업데이트해야 합니다. 악의적 공격에 대한 회복력을 높이려면, 샤드 내 노드는 주기적으로 재배포되어야 합니다. 그러나 노드를

샤드 간에 이동시키면 새로 추가된 노드가 최신 상태를 다운로드하는 동기화 오버헤드가 발생합니다. 따라서 각 에폭마다 모든 노드를 재배포하는 것이 아니라 일부 노드만 재배포해야 동기화 과정에서 다owntime을 방지할 수 있습니다.

3 샤딩의 방향

일부 샤딩 제안은 트랜잭션만 [8] 또는 상태만 [17] 샤딩하려 하며, 이는 각 노드가 대량의 상태 데이터를 저장하거나 슈퍼컴퓨터 수준의 성능을 요구하는 방식으로 트랜잭션 처리량을 증가시킵니다 [2]. 그러나 최근에는 저장 또는 계산 성능을 손상시키지 않고 트랜잭션과 상태 샤딩을 모두 성공적으로 수행할 수 있다는 주장이 있었습니다 [13].

하지만 샤딩은 새로운 도전 과제도 도입합니다. 예: 단일 샤드 점령 공격, 크로스샤드 통신, 데이터 가용성, 샤드를 숨기는 추상화 계층의 필요성 등이 있습니다. 그러나 이러한 문제들이 올바르게 해결된다는 전제하에, 상태 샤딩은 전체적으로 상당한 개선을 가져옵니다. 즉, 트랜잭션 처리량은 병렬 트랜잭션 처리 덕분에 크게 증가하며, 트랜잭션 수수료는 크게 감소합니다. 이는 블록체인 기술의 대중 채택에 있어 걸림돌로 여겨졌던 두 가지 주요 기준을 장점과 유인 요인으로 바꾸어 줍니다.

4 MultiversX의 샤딩 접근

네트워크, 트랜잭션, 상태 샤딩을 결합하는 복잡성을 다루기 위해, MultiversX의 접근은 다음과 같은 목표를 염두에 두고 설계되었습니다:

1. 가용성을 저해하지 않는 확장성: 샤드 수의 증가 또는 감소는 다owntime을 유발하지 않고 상태 업데이트 중에도 영향을 최소화하면서 극히 일부 노드에만 영향을 주어야 함.
2. 분배와 즉시 추적 가능성: 트랜잭션의 목적지 샤드를 결정하는 과정이 결정적이고 계산이 단순해야 하며, 추가적인 통신 라운드가 필요 없어야 함.
3. 효율성과 적응성: 어느 시점에서든 샤드는 가능한 균형을 유지해야 함.

방법 설명

다음 에폭 e_{i+1} 에서의 최적 샤드 수 $N_{sh,i+1}$ 를 계산하기 위해, 블록 내 트랜잭션 수에 대한 임계 계수 θ_{TX} 를 정의했습니다. 변수 $optN$ 은 샤드 내 최적 노드 수를 나타내며, ϵ_{sh} 는 샤드가 변동할 수 있는 노드 수를 나타내는 양의 값입니다. $totalN_i$ 는 에폭 e_i 의 모든 샤드에서의 총 노드 수(자격 검증자, 대기 리스트 노드, 새로 추가된 노드 포함)를 의미하며, $N_{TXB,i}$ 는 에폭 e_i 의 모든 샤드에서 블록당 평균 트랜잭션 수입니다. $N_{sh,0}$ 는 1로 간주합니다. 전체 샤드 수 $N_{sh,i+1}$ 는 네트워크의 노드 수 $totalN_i$ 가 변하거나 블록체인 활용도가 필요할 경우 변경됩니다: 즉, 노드 수가 한 에폭에서 다음 에폭으로 n_{Split} 임계값을 초과하고, 블록당 평균 트랜잭션 수가 θ_{TX} 를 초과할 경우 $N_{TXB,i} > \theta_{TX}$, 또는 노드 수가 n_{Merge} 임계값 아래로 떨어질 경우 **ComputeShardsN** 함수와 같이 변경됩니다.

- 1: function COMPUTESHARDSN($totalN_{i+1}$, $N_{sh,i}$)
- 2: $n_{Split} \leftarrow (N_{sh,i} + 1) \cdot (optN + \epsilon_{sh})$
- 3: $n_{Merge} \leftarrow (N_{sh,i} - 1) \cdot a$
- 4: $N_{sh,i+1} \leftarrow N_{sh,i}$
- 5: if ($totalN_{i+1} > n_{Split}$ and $N_{TXB,i} > \theta_{TX}$) then

```

6:    $N_{sh,i+1} \leftarrow \frac{totalN_{i+1}}{optN + \epsilon_{sh}}$ 
7:   else if ( $totalN_{i+1} < nMerge$ ) then
8:      $N_{sh,i+1} \leftarrow \frac{totalN_{i+1}}{optN}$ 
9:   return  $N_{sh,i+1}$ 

```

한 에폭에서 다음 에폭으로 넘어가면서 활성 노드 수가 변할 수 있습니다. 만약 이것이 샤드 수에 영향을 준다면, 누구나 트랜잭션 분배에 사용되는 두 개의 마스크 m_1 과 m_2 를 계산할 수 있습니다.

```

1: function COMPUTEM1ANDM2( $N_{sh}$ )
2:    $n \leftarrow \lceil \log_2 N_{sh} \rceil$ 
3:    $m_1 \leftarrow (1 \ll n) - 1$ 
4:    $m_2 \leftarrow (1 \ll (n - 1)) - 1$ 
5:   return  $m_1, m_2$ 

```

주요 목표가 초당 수천 건 이상의 트랜잭션 처리량을 확보하고 크로스샤드 통신을 줄이는 것이므로, MultiversX는 트랜잭션에 포함된 계정 주소(*addr*)를 기반으로 자동으로 관련 샤드를 결정하고 해당 샤드로 라우팅하는 분배 메커니즘을 제안합니다.

```

1: function COMPUTESHARD( $N_{sh}, addr, m_1, m_2$ )
2:    $shard \leftarrow (addr \text{ and } m_1)$ 
3:   if ( $shard > N_{sh}$ ) then
4:      $shard \leftarrow (addr \text{ and } m_2)$ 
5:   return  $shard$ 

```

이 샤딩 방식은 계정 주소를 분배하고 상태 전이를 관리하는 이진 트리 구조에 기반합니다. 그림 2는 트리의 예시를 보여줍니다.

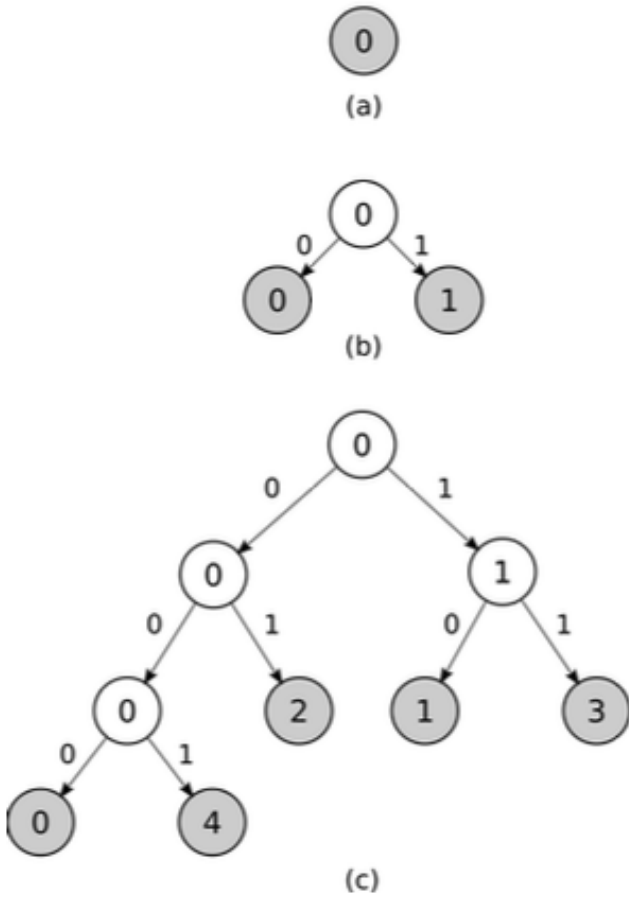


그림 2: 샤딩 트리 구조 예시

트리 구조는 샤드 할당, 형제 노드 계산 등을 위한 결정적 매핑에 사용되는 계정 주소 공간의 논리적 표현에 불과합니다. 이진 트리의 리프는 샤드와 그 ID 번호를 나타냅니다. 루트(노드/샤드 0)에서 시작해 하나의 샤드/리프(a)만 있는 경우, 모든 계정 주소는 해당 샤드에 매핑되며 모든 트랜잭션이 이곳에서 실행됩니다. 이후 N_{sh} 공식에 따라 두 개의 샤드가 필요하다면(b), 주소 공간은 마지막 비트를 기준으로 동일하게 분할됩니다.

트리가 불균형(c)해질 수도 있는데, 이는 N_{sh} 가 2의 거듭제곱이 아닐 때 발생하며 마지막 레벨의 리프만 영향을 받습니다. 샤드 수가 다시 2의 거듭제곱에 도달하면 구조는 균형을 되찾습니다.

이진 트리의 불균형은 마지막 레벨에 위치한 샤드가 더 높은 레벨의 샤드보다 절반의 주소 공간을 가지게 하여, 해당 샤드의 노드가 낮은 수수료 수익을 얻게 할 수 있습니다(블록 보상은 영향 없음). 그러나 이는 매 에폭마다 각 샤드의 노드 1/3이 무작위로 재분배되는 방식(Chronology 절 참고)과 트리 레벨에 따른 균형 잡힌 노드 분배로 해결됩니다.

트리를 루트에서 리프로 탐색할 때, 브랜치 인코딩은 계정 주소의 마지막 n 비트를 나타내며, 이 주소에서 발생한 트랜잭션이 해당 샤드에서 처리됨을 의미합니다. 반대로 리프에서 루트로 탐색할 경우, 구조의 진화, 형제 샤드, 분할된 부모 샤드에 대한 정보를 나타냅니다. 이를 통해 N_{sh} 가 증가할 때 분할될 샤드나 N_{sh} 가 감소할 때 병합될 샤드를 쉽게 계산할 수 있습니다. 이 구조 덕분에 상태 샤딩 메커니즘은 항상 주소와 연관된 상태를 동일한 샤드 내에 유지합니다.

N_{sh} 를 알면, 어떤 노드든 통신 없이 재분배 과정을 따를 수 있습니다. 새 샤드 ID는 순차적으로 할당되며, 샤드 수가 줄어들면 높은 번호의 샤드가 제거됩니다. 예를 들어, N_{sh} 에서 $N_{sh} - 1$ 로 줄어드는 경우, 제거되는 샤드는 가장 높은 번호의 샤드($sh_{merge} = N_{sh} - 1$)입니다. 이 샤드가 병합될 샤드 번호는 트리 구조에 따라 그 형제의 번호입니다.

```

1: function COMPUTESIBLING( $sh_{merge}, n$ )
2:    $sibling \leftarrow (sh_{merge} \oplus (1 \ll (n - 1)))$ 
3:   return  $sibling$ 

```

샤드 중복성, 상태 전이 추적, 빠른 확장을 위해 임의 샤드 번호 p 의 형제와 부모를 결정하는 것도 중요합니다.

```

1: function COMPUTEPARENTSIBLINGS( $n, p, N_{sh}$ )
2:    $mask_1 \leftarrow 1 \ll (n - 1)$ 
3:    $mask_2 \leftarrow 1 \ll (n - 2)$ 
4:    $sibling \leftarrow (p \oplus mask_1)$ 
5:    $parent \leftarrow \min(p, sibling)$ 
6:   if ( $sibling \geq N_{sh}$ ) then
7:      $sibling \leftarrow (p \oplus mask_2)$ 
8:      $sibling2 \leftarrow (sibling \oplus mask_1)$ 
9:      $parent \leftarrow \min(p, sibling)$ 
10:    if ( $sibling2 \geq N_{sh}$ ) then ▷ sibling is a shard
11:      return  $parent, sibling, NULL$ 
12:    else ▷ sibling is a subtree with shards ( $sibling, sibling2$ )
13:      return  $parent, sibling, sibling2$ 
14:  else ▷ sibling is a shard
15:    return  $parent, sibling, NULL$ 

```

샤드 중복성(Shard redundancy)

블록체인에서 상태 샤딩은 샤드 내 온라인 노드 수가 부족하거나 분포가 지리적으로 편중될 경우 샤드 실패에 취약합니다. 만약 하나의 샤드가 실패한다면(연락 불가, 노드가 모두 오프라인이거나 합의 불가), 전체 아키텍처가 모든 샤드의 블록을 다운로드하고 검증하는 슈퍼 풀 노드(super-full node)에만 의존하게 될 위험이 있습니다 [2]. 그림 3에서 보듯, 우리 프로토콜은 마지막 트리 레벨의 샤드가 형제 샤드의 상태도 보유하도록 강제하는 보호 메커니즘을 도입했습니다. 이 방식은 통신을 줄이고, 병합 시 형제 샤드가 이미 데이터를 가지고 있으므로 부트스트래핑을 제거합니다.

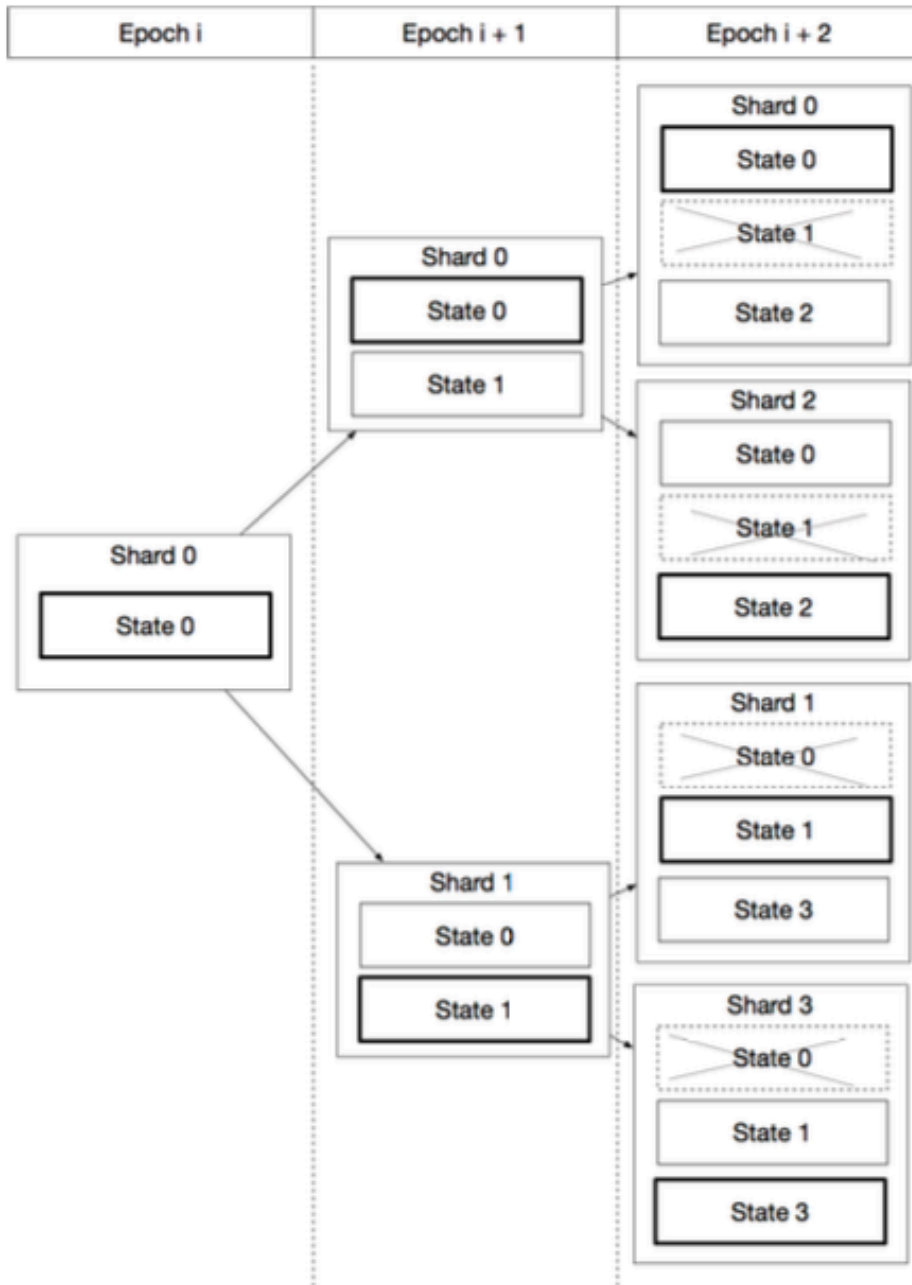


그림 3: 에폭 간 샤드 중복성

컨텍스트 스위칭(Context switching)

샤딩된 퍼블릭 블록체인에서 보안을 유지하려면 컨텍스트 스위칭이 중요합니다 [7]. 이는 일정한 시간 간격마다 무작위 기준으로 활성 노드를 샤드 간에 재할당하는 것을 의미합니다. MultiversX에서는 컨텍스트 스위칭이 보안 향상 효과를 주지만, 동시에 여러 상태 간 일관성을 유지하기 위한 복잡성도 증가시킵니다. 상태 전이는 성능에 가장 큰 영향을 미치는데, 이는 활성 노드 이동 시 새 샤드에서 상태, 블록체인, 트랜잭션을 재 동기화해야 하기 때문입니다. 에폭 시작 시 생존성을 유지하기 위해 각 샤드의 노드 중 1/3 미만만 고르게 재 분배됩니다. 이 메커니즘은 악의적 그룹의 형성을 효과적으로 방지합니다.

5 공증(Meta) 체인

네트워크 및 글로벌 데이터 운영(노드 참여, 노드 이탈, 자격 검증자 리스트 계산, 노드의 샤드 대기 리스트 배정, 특정 샤드의 블록 합의, 무효 블록 검증을 위한 챌린지 등)은 메타체인에 의해 공증(notarize)됩니다. 메타체인 합의는 모든 다른 샤드와 통신하며 크로스샤드 작업을 조정하는 별도의 샤드에서 실행됩니다. 각 에폭의 매 라운드마다 메타체인은 다른 샤드들로부터 블록 헤더와 필요한 경우 무효 블록 검증 증명을 수신합니다. 이 정보는 메타체인의 블록에 집계되며 합의를 거칩니다. 블록이 합의 그룹에 의해 검증되면, 샤드는 블록, 미니블록(VII장 참고), 자격 검증자, 대기 리스트 노드 등에 대한 정보를 요청하여 크로스샤드 트랜잭션을 안전하게 처리할 수 있습니다. 크로스샤드 트랜잭션 실행, 샤드 간 통신, 메타체인과의 상호작용에 대한 세부 내용은 VII장에서 설명됩니다.

V Secure Proof of Stake를 통한 합의

1 합의 분석

최초의 블록체인 합의 알고리즘은 작업증명(PoW)에 기반하며, 비트코인, 이더리움 및 다른 블록체인 플랫폼에서 사용됩니다. PoW에서는 각 노드가 수학적 퍼즐을 풀어야 하며(계산은 어렵지만 검증은 쉬움), 퍼즐을 가장 먼저 푼 노드가 보상을 받습니다 [18]. PoW 메커니즘은 이중지불, DDoS, 시빌 공격을 성공적으로 방지했으나, 높은 에너지 소비라는 대가가 있습니다.

지분증명(PoS)은 PoW 합의 메커니즘의 에너지 및 계산 집약적 사용에 대한 대안으로 제안된 새로운 효율적 합의 메커니즘입니다. PoS는 Cardano [19], Algorand [3]와 같은 많은 새로운 아키텍처에서 발견되며, 이더리움 차기 버전에서도 사용될 수 있습니다. PoS에서는 다음 블록을 제안할 노드가 스테이크(지분, wealth), 무작위성, 나이(age)의 조합에 따라 선택됩니다. 이는 PoW의 에너지 문제를 완화하지만, Nothing at Stake 공격과 중앙화 위험 증가라는 두 가지 중요한 문제를 야기합니다.

Constellation [20]에서 제시한 Proof of Meme은 네트워크 내 노드의 과거 참여도에 기반한 알고리즘입니다. 노드의 행위는 블록체인 내 가중치 행렬에 저장되며 시간이 지나면서 변화할 수 있습니다. 또한 새로운 노드는 평판을 쌓아 신뢰를 얻을 수 있습니다. Sybil 공격에 대한 주요 단점은 NetFlow 알고리즘으로 완화됩니다.

Delegated Proof of Stake(DPoS)는 Bitshares [21], Steemit [22], EOS [23] 등에서 발견되며, 권위증명(Proof of Authority)과 PoS의 하이브리드입니다. 새로운 블록을 배포할 책임이 있는 소수의 노드가 이해관계자(stakeholder)에 의해 선출됩니다. 높은 처리량을 가지지만, 뇌물 수수, 부패 등 인간적·사회적 문제에 취약합니다. 또한 소수의 대표만 존재하기 때문에 DDoS 공격과 중앙화에도 취약합니다.

2 Secure Proof of Stake (SPoS)

MultiversX의 합의 접근은 무작위 검증자 선택, 스테이크와 평판(rating)에 따른 자격 부여, 최적 합의 그룹 크기를 결합하는 방식으로 이루어집니다. 알고리즘은 다음 단계로 설명됩니다:

1: 각 노드 n_i 는 공개키(Pk), 평판(기본값 0), 잠금된 스테이크의 튜플로 정의됩니다. n_i 가 합의에 참여하려면, 먼저 스마트 컨트랙트를 통해 등록해야 하며, 이때 최소 스테이크와 기타 정보(Pk_s , 실제 지갑 주소를 사용하지 않도록 Pk 로부터 파생된 공개키 및 $nodeid$)를 포함한 트랜잭션을 전송해야 합니다.

2: 노드 n_i 는 노드 풀에 들어가 현재 에폭 e 종료 시 샤드 배정을 기다립니다. 샤드 배정 메커니즘은 에폭 e 에 합류한 모든 노드와 매 샤드의 $1/3$ 미만이 재배포되는 노드 집합을 만듭니다. 이 집합 내 노드는 샤드 대기 리스트에 재배포됩니다. W_j 는 샤드 j 의 대기 리스트를 의미하며, N_{sh} 는 샤드 수를 의미합니다. 각 노드는 비공개키 sk 를 가지고 있으며, 이는 공개되어서는 안 됩니다.

$$n_i = (Pk_i, rating_i, stake_i)$$

$$n_i \in W_j, 0 \leq j < N_{sh}$$

3: 해당 에폭이 끝나면, 노드는 샤드 j 의 자격 노드 리스트(E_j)로 이동합니다.

$$n_i \in W_{j,e-1} \rightarrow n_i \notin W_{j,e}, n_i \in E_{j,e}$$

4: E_j 에 속한 각 노드는 합의 그룹의 일원으로 선택될 수 있습니다. 이는 무작위 소스(이전 블록에 추가됨), 라운드 r , 가변 매개변수 집합에 기반한 결정적 함수에 의해 이루어집니다. 난수는 가십(gossip)을 통해 샤드 내 모든 노드가 알 수 있으며, 실제로 이전 합의 그룹이 블록에 서명하기 전까지 예측할 수 없습니다. 이 성질은 난수 소스로 적합하며 고도로 적응적인 악의적 공격을 방지합니다.

합의 그룹(선택된 노드 집합) N_{chosen} 을 반환하는 선택 함수를 정의합니다. 첫 번째 노드는 블록 제안자이며, 매개변수는 E, r, sig_{r-1} (이전 블록 서명)입니다.

$$N_{chosen} = f(E, r, sig_{r-1}), N_{chosen} \subset E$$

5: 블록은 블록 제안자에 의해 생성되고, 검증자들은 수정된 pBFT에 기반해 블록에 공동 서명을 합니다.

6: 어떤 이유로든 블록 제안자가 할당된 시간 내 블록을 생성하지 않으면(악의적 행위, 오프라인 등), 라운드 r 은 마지막 블록의 무작위 소스와 함께 사용되어 새로운 합의 그룹을 선택합니다.

만약 블록 제안자가 악의적으로 행동한다면, 나머지 그룹 구성원은 그 평판을 낮추는 부정적 피드백을 적용합니다. 이렇게 하면 해당 노드가 다시 선택될 가능성이 줄어들거나 사라집니다. 라운드 r 에서 블록 제안자(n_i)에 대한 피드백 함수는 다음과 같이 계산됩니다:

$$feedbackFunction = ff(n_i, ratingModifier, r)$$

여기서 $ratingModifier < 0$ 이면 슬래싱이 발생하며, 노드 n_i 는 스테이크를 잃습니다.

합의 프로토콜은 E 리스트에 포함된 많은 잠재적 검증자(수백 개 노드)와 선택 전까지 검증자 순서를 예측할 수 없는 성질로 인해 DDoS 공격에도 안전합니다.

샤드 수 증가에 따라 통신 오버헤드가 커지는 것을 줄이기 위해, 합의는 복합 블록(composite block)에 대해 실행됩니다. 복합 블록은 다음으로 구성됩니다:

- **원장 블록(Ledger block):** 샤드 원장에 추가될 블록으로, 모든 샤드 내 트랜잭션과 확인 증명이 수신했던 크로스샤드 트랜잭션을 포함함.
- **여러 개의 미니블록(mini-blocks):** 각기 다른 샤드로의 크로스샤드 트랜잭션을 포함함.

합의는 복합 블록(샤드 내 및 크로스샤드 트랜잭션 모두 포함)에 대해 단 한 번 실행됩니다. 합의에 도달하면, 각 샤드의 블록 헤더가 메타체인으로 전송되어 공증됩니다.

VI 암호학 계층 (Cryptographic Layer)

1 서명 분석(Signature Analysis)

디지털 서명은 메시지 인증, 데이터 무결성, 부인 방지(non-repudiation)와 같은 속성을 제공함으로써 정보 보안을 달성하기 위해 사용되는 암호학적 원시입니다 [24].

기존 블록체인 플랫폼에서 사용되는 대부분의 기법은 이산 로그(DL) 문제에 의존합니다: 일방향 지수 함수 $y \rightarrow \alpha^y \bmod p$. 밑을 사용하는 이산 로그 계산은 어려운 것으로 과학적으로 입증되었습니다 [25].

타원 곡선 암호(ECC)는 정수의 순환군 대신 점들의 순환군을 사용합니다. 이 방식은 계산 노력을 줄여, 160–256비트 길이의 키로도 RSA, Elgamal, DSA 등이 1024–3072비트 키 길이에서 제공하는 동일한 보안 수준을 확보할 수 있습니다(표 1 [24] 참고).

알고리즘 계열	암호 시스템	보안 수준(비트)	80	128	192	256
정수 인수분해	RSA	키 길이(비트)	1024	3072	7680	15360
이산 로그	DH, DSA, Elgamal	키 길이(비트)	1024	3072	7680	15360
타원 곡선	ECDH, ECDSA	키 길이(비트)	160	256	384	512
대칭키	AES, 3DES	키 길이(비트)	80	128	192	256

표 1: 보안 수준별 공개키 알고리즘의 비트 길이

ECC가 훨씬 짧은 매개변수 길이로 동일한 보안 수준을 제공할 수 있는 이유는, 타원 곡선 그룹에 대한 기존 공격이 정수 DL 공격보다 약하기 때문입니다. 이러한 알고리즘의 복잡도는 평균적으로 $\sqrt{p}$ 단계가 필요합니다. 즉, 256비트 소수 p 를 사용하는 타원 곡선은 평균적으로 2^{128} 단계가 필요하므로 [24], 이 수준의 보안을 제공합니다.

이더리움과 비트코인은 모두 ECDSA 서명 알고리즘을 사용하는 타원 곡선 암호를 사용합니다. 그러나 이 알고리즘의 보안은 난수 발생기에 크게 의존합니다. 난수 발생기가 매번 다른 수를 생성하지 못하면 개인키가 노출될 수 있기 때문입니다 [26].

또 다른 디지털 서명 기법은 EdDSA로, 빠른 연산을 지원하는 비틀린 에드워즈 곡선을 기반으로 한 슈노르(Schnorr) 변형입니다 [27]. ECDSA와 달리, 이는 비변조성(provably non-malleable)을 보장합니다. 즉, 단일 서명에서 동일한 타원 곡선상의 점을 정의하는 또 다른 매개변수 집합을 찾는 것은 불가능합니다 [28], [29]. 또한 EdDSA는 개인키와 메시지를 해싱해 계산한 논스를 사용하므로, 난수 발생기의 취약성이

개인키를 드러내는 공격 벡터를 피할 수 있습니다.

슈노르 서명 변형은 본질적 다중서명 기능과 무작위 오라클 모델에서의 증명 가능한 보안성 덕분에 점차 주목 받고 있습니다 [8], [30], [31]. 다중서명 스킴은 각자의 키를 가진 여러 서명자가 동일한 메시지에 서명해 단일 서명을 생성하는 서명 및 검증 알고리즘의 결합입니다 [32], [33]. 비효율적 방법은 각 노드가 서명을 계산한 뒤 모두 연결(concatenate)하는 것이지만, 이는 서명자 수에 비례해 문자열 크기가 증가해 비현실적입니다. 실용적 해법은 출력을 단일 고정 크기 서명으로 집계하는 것이며, 이는 참여자 수와 무관합니다. 그러나 이러한 기법 대부분은 위조 키(rogue-key) 공격에 취약합니다. 이를 해결하기 위해 각 서명자가 공개 키에 대응하는 개인키를 실제로 보유하고 있음을 증명하는 단계가 필요합니다 [34].

Bellare와 Neven [35]은 공개키 집계 없이도 안전한 다중서명 스킴을 제안했습니다. 이 방식에서 참여자는 자신의 부분 서명 값 R_i 를 먼저 커밋하고 해시를 배포해 다른 서명자가 이를 함수화할 수 없도록 합니다. 각 서명자는 서로 다른 챌린지를 계산합니다. 그러나 이 방식은 공개키 집계를 희생합니다. 따라서 집계된 서명을 검증하려면 각 서명자의 공개키가 필요합니다.

Gregory Maxwell 등 [29]은 “one more discrete logarithm”(OMDL) 가정하에 평문 공개키 모델에서 또 다른 다중서명 스킴을 제안했습니다 [36]. 이 접근은 통신 라운드를 3회에서 2회로 줄이고, 키 집계를 재도입했지만 더 높은 복잡도 비용을 요구합니다.

BLS [4]는 Weil 페어링을 기반으로 하는 흥미로운 서명 기법으로, 특정 타원 곡선에 대한 계산적 디피-헬만 가정을 기반으로 짧은 서명을 생성합니다. BLS는 배치 검증, 서명 집계, 공개키 집계 등의 유용한 속성을 지녀, 임계값 서명과 다중서명 스킴의 좋은 후보입니다.

Boneh, Drijvers, Neven은 최근 BLS 다중서명 스킴을 제안했으며 [5], 이는 [35], [30]의 이전 연구 아이디어를 사용하여 위조 키 공격 방어를 추가했습니다. 이 스킴은 단 두 번의 페어링으로 다중서명을 검증할 수 있고, 개인키 지식 증명이 필요하지 않습니다(평문 공개키 모델에서 동작). 또 다른 장점은 다중서명이 단 두 차례의 통신 라운드로 생성된다는 점입니다.

추적성과 보안상의 이유로, 축소된 검증자 집합에 기반한 합의는 각 서명자의 공개키를 필요로 합니다. 이 맥락에서, 우리 분석은 MultiversX에서 블록 서명을 위한 최적 다중서명 스킴으로 BLS 다중서명 [5]을 결론지었습니다. 이는 단 두 차례의 통신 라운드만 필요해 다른 옵션보다 전반적으로 더 빠릅니다.

2 MultiversX에서의 블록 서명

블록 서명을 위해, MultiversX는 $bn256$ 이중선형 그룹 위에서 BLS 다중서명 스킴을 사용하는 타원 곡선 암호를 사용합니다. 이는 256비트 Barreto–Naehrig 곡선 위에서 Optimal Ate 페어링을 구현합니다. 이중선형 페어링은 다음과 같이 정의됩니다:

$$e : g_0 \times g_1 \rightarrow g_t$$

여기서 g_0, g_1, g_t 는 $bn256$ 이 정의한 소수 차수 p 의 타원 곡선이며, e 는 이중선형 사상(페어링 함수)입니다. G_0, G_1 는 각각 g_0, g_1 의 생성자입니다. 또한, H_0 는 메시지를 곡선 g_0 위의 점으로 매핑하는 해싱 함수입니다:

$$H_0 : M \rightarrow g_0$$

여기서 M 은 임의 길이의 모든 이진 메시지 집합입니다. MultiversX에서 사용하는 서명 스킴은 두 번째 해싱 함수도 사용하며, 이는 모든 서명자에게 알려진 매개변수를 가집니다:

$$H_1 : M \rightarrow \mathbb{Z}_p$$

각 서명자 i 는 자신의 개인/공개키 쌍 (sk_i, Pk_i) 를 가지고 있으며, 여기서 sk_i 는 $\mathbb{Z}_p$ 에서 무작위로 선택됩니다. 각 키 쌍은 $Pk_i = sk_i \cdot G_1$ 의 성질을 가집니다.

특정 라운드에서의 모든 서명자의 공개키 집합 $L = Pk_1, Pk_2, \dots, Pk_n$ 은 MultiversX의 경우 합의 그룹 내 모든 노드의 공개키 집합입니다. 블록 서명 과정은 서명과 검증의 두 단계로 나뉩니다.

실용적 서명 – 라운드 1

합의 그룹 리더가 트랜잭션을 담은 블록을 생성하고, 서명한 뒤 이를 그룹 구성원에게 브로드캐스트합니다.

실용적 서명 – 라운드 2

합의 그룹의 각 구성원(리더 포함)은 블록을 수신하면 이를 검증하고, 유효하다면 BLS로 서명해 리더에게 전송합니다:

$$Sig_i = sk_i \cdot H_0(m)$$

여기서 Sig_i 는 g_0 위의 점입니다.

실용적 서명 – 라운드 3

리더는 일정 시간 동안 서명을 수집합니다. 만약 $2/3 \cdot n + 1$ 이상의 유효 서명을 수신하지 못하면 합의 라운드는 중단됩니다. 그러나 $2/3 \cdot n + 1$ 이상의 유효 서명을 수신하면, 이를 사용해 집계 서명을 생성합니다:

$$SigAgg = \sum_i H_1(Pk_i) \cdot Sig_i \cdot B[i]$$

여기서 $SigAgg$ 는 g_0 위의 점입니다. 리더는 이후 집계 서명과 함께 서명자 비트맵 B 를 블록에 추가합니다. 여기서 1은 리스트 L 내 해당 서명자의 서명이 집계 서명에 포함되었음을 나타냅니다.

실용적 검증

검증자는 공개키 리스트 L , 서명자 비트맵 B , 집계 서명 $SigAgg$, 메시지 m (블록)을 가지고 집계 공개키를 계산합니다:

$$PkAgg = \sum_i H_1(Pk_i) \cdot Pk_i \cdot B_i$$

결과 $PkAgg$ 는 g_1 위의 점입니다. 최종 검증은 다음과 같습니다:

$$e(G_1, SigAgg) == e(PkAgg, H_0(m))$$

여기서 e 는 페어링 함수입니다.

VII 크로스샤드 실행 (Cross-shard Execution)

크로스샤드 트랜잭션이 어떻게 실행되고, 샤드와 메타체인 간의 통신이 어떻게 이루어지는지를 심층적으로 예시하기 위해, 전체 과정을 두 개의 샤드와 메타체인만으로 단순화합니다. 한 사용자가 자신의 지갑(샤드 0 주소 보유)에서 트랜잭션을 생성해, 샤드 1 주소를 가진 다른 사용자에게 EGLD를 전송하려 한다고 가정합니다. 이 경우 크로스샤드 트랜잭션을 처리하기 위해 그림 4의 단계가 필요합니다.

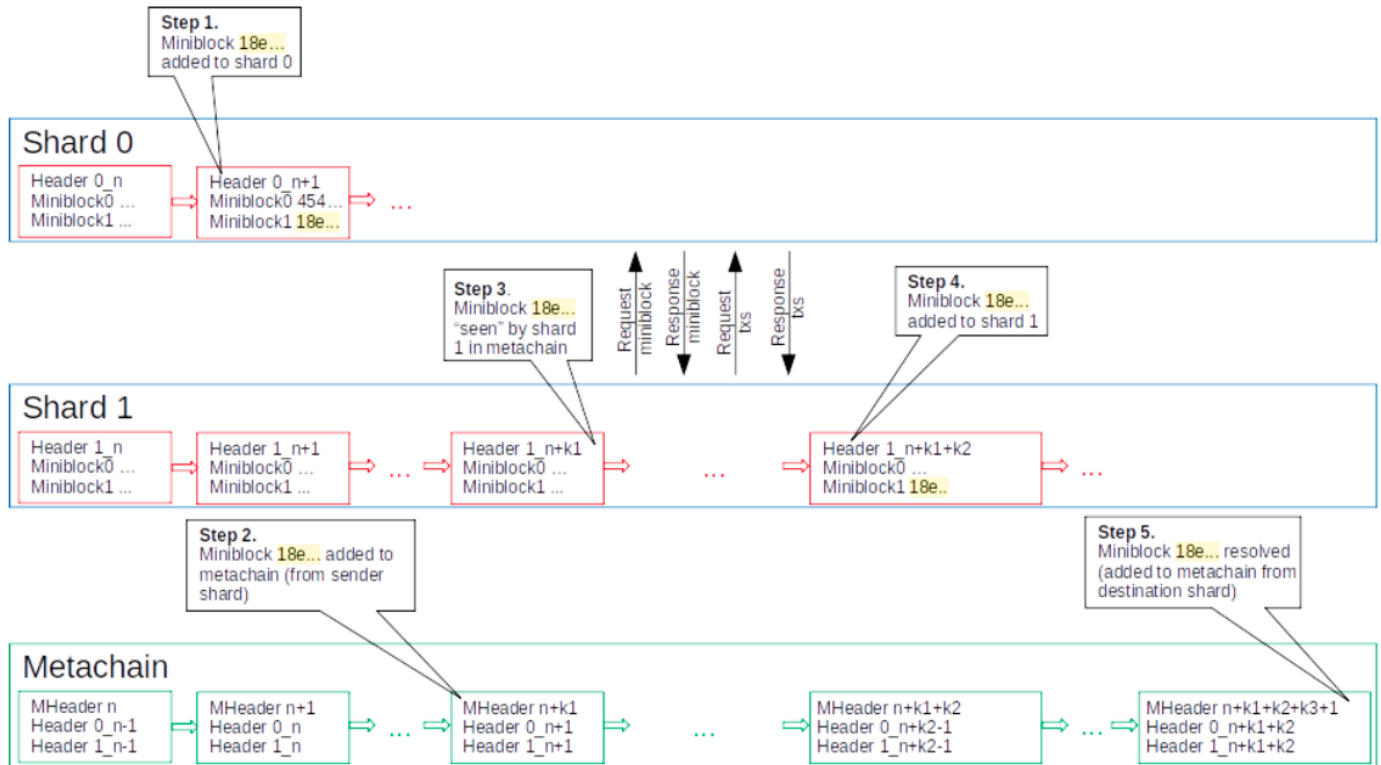


그림 4: 크로스샤드 트랜잭션 처리

V장 – Secure Proof of Stake를 통한 합의에서 언급했듯, 블록 구조는 블록 논스, 라운드, 제안자, 검증자, 타임스탬프 등 블록 관련 정보를 담은 블록 헤더와, 실제 트랜잭션을 담은 샤드별 미니블록 리스트로 구성됩니다. 각 미니블록은 송신자가 현재 샤드에 있고 수신자가 다른 샤드에 있는 트랜잭션, 또는 송신자가 다른 샤드에 있고 수신자가 현재 샤드에 있는 트랜잭션을 포함합니다.

우리의 예에서, 샤드 0 블록에는 보통 3개의 미니블록이 있습니다:

- **미니블록 0:** 샤드 0 내부 트랜잭션 포함
- **미니블록 1:** 송신자가 샤드 0, 수신자가 샤드 1인 크로스샤드 트랜잭션 포함
- **미니블록 2:** 송신자가 샤드 1, 수신자가 샤드 0인 크로스샤드 트랜잭션 포함. 이들은 이미 송신 샤드 1에서 처리되었으며, 현재 샤드에서 처리 후 최종화됩니다.

동일한 송신자-수신자 쌍을 가진 미니블록 수에는 제한이 없습니다. 즉, 같은 송신자-수신자 조합의 여러 미니블록이 동일 블록에 나타날 수 있습니다.

1 처리 과정 (Processing)

현재 크로스샤드 실행에서의 원자적 처리 단위는 미니블록입니다. 즉, 미니블록 내 모든 트랜잭션이 한 번에 처리되거나 전혀 처리되지 않으며, 실패 시 다음 라운드에서 재시도됩니다.

우리의 크로스샤드 트랜잭션 전략은 비동기(asynchronous) 모델을 사용합니다. 검증과 처리는 먼저 송신자의 샤드에서, 그다음 수신자의 샤드에서 이루어집니다. 트랜잭션은 송신자의 샤드에서 먼저 분배되며, 이곳에서 해당 계정의 현재 잔액 검증 등 모든 검증을 완료할 수 있습니다. 이후 수신자의 샤드에서는 메타체인이 제공하는 실행 증명만 필요하며, 서명 검증과 재사용 공격(replay attack) 확인을 거쳐 최종적으로 수신자의 잔액을 갱신하고 트랜잭션 금액을 추가합니다.

샤드 0은 미니블록 0의 내부 트랜잭션과, 수신자가 샤드 1인 크로스샤드 트랜잭션 집합(미니블록 1)을 처리합니다. 블록 헤더와 미니블록은 메타체인으로 전송됩니다. 메타체인은 샤드 0 블록을 공증(notarize)하며, 새로운 메타체인 블록(메타블록)을 생성합니다. 이 메타블록은 각 미니블록에 대한 다음 정보를 포함합니다: 송신 샤드 ID, 수신 샤드 ID, 미니블록 해시.

샤드 1은 메타블록에서 미니블록 1의 해시를 가져오고, 샤드 0으로부터 해당 미니블록을 요청한 뒤, 트랜잭션 리스트를 파싱합니다. 필요한 트랜잭션이 누락되면 이를 요청합니다. 그런 다음 동일한 미니블록 1을 샤드 1에서 실행하고, 결과 블록을 메타체인으로 전송합니다. 공증이 완료되면 크로스샤드 트랜잭션 집합은 최종화됩니다.

다음 다이어그램은 트랜잭션이 최종화되기까지 필요한 라운드 수를 보여줍니다. 라운드는 미니블록에 최초 포함된 시점부터 마지막 미니블록이 공증될 때까지를 의미합니다.

VIII 스마트 컨트랙트 (Smart Contracts)

스마트 컨트랙트의 실행은 모든 미래 블록체인 아키텍처에서 핵심 요소입니다. 그러나 기존 솔루션의 대부분은 트랜잭션과 데이터 종속성을 제대로 설명하지 않았습니다. 이 맥락은 다음 두 가지 시나리오로 이어집니다:

1. 스마트 컨트랙트 트랜잭션 간에 직접적 상관관계가 없는 경우(그림 5 참조), 어떤 아키텍처든 순서에 상관없이(out of order) 스케줄링을 사용할 수 있습니다. 이는 스마트 컨트랙트가 언제, 어떤 샤드에서 실행되든 추가 제약이 없음을 의미합니다.

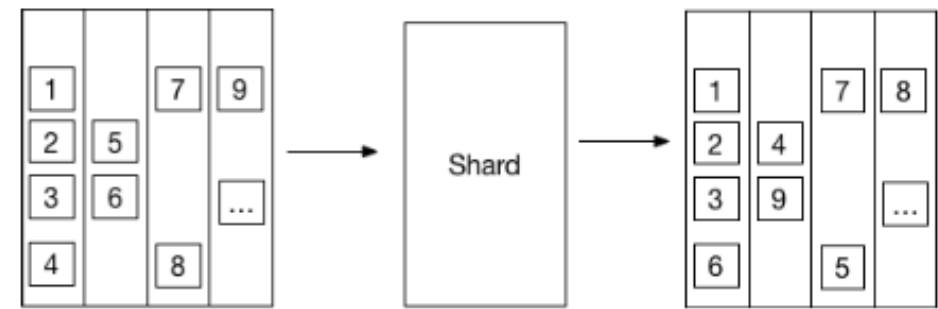


그림 5: 순서와 무관하게 실행될 수 있는 단순 스마트 컨트랙트에서의 독립적 트랜잭션 처리

2. 두 번째 시나리오는 상호 연관된 스마트 컨트랙트 트랜잭션이 유발하는 병렬성(parallelism)을 의미합니다 [37]. 이 경우(그림 6 참조)는 성능에 추가적인 부담을 주고 복잡성을 상당히 증가시킵니다. 기본적으로, 컨트랙트가 올바른 순서와 올바른 위치(샤드)에서 실행되도록 보장하는 메커니즘이 필요

합니다. 이를 해결하기 위해 MultiversX 프로토콜은 스마트 컨트랙트를 해당 정적 종속성이 존재하는 동일 샤드로 배치 및 이동시키는 해법을 제안합니다. 이 방식으로 대부분, 혹은 모든 스마트 컨트랙트 호출은 동일 샤드 내에서 종속성을 가지게 되어, 크로스샤드 잠금/해제가 필요 없게 됩니다.

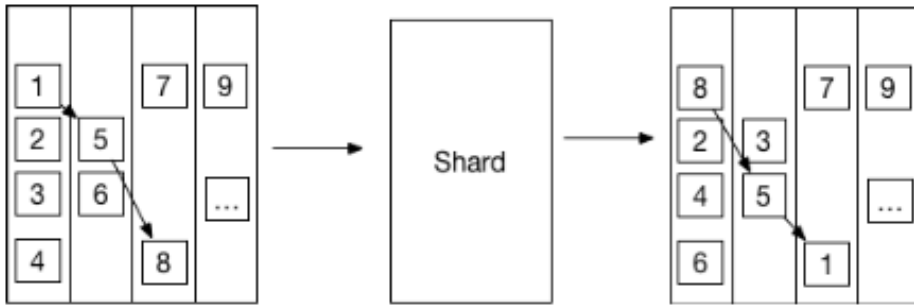


그림 6: 순차적으로만 실행될 수 있는 상관된 스마트 컨트랙트를 위한 메커니즘

MultiversX는 MultiversX 가상머신(MultiversX Virtual Machine, VM) 구현에 집중하고 있으며, 이는 EVM 호환 엔진입니다. EVM 호환성은 채택을 위해 매우 중요한데, 이더리움 플랫폼 위에 구축된 수많은 스마트 컨트랙트가 존재하기 때문입니다.

MultiversX 가상머신 구현은 시스템 내부를 스마트 컨트랙트 개발자로부터 격리해 올바른 추상화 계층을 보장하며, 이는 그림 7과 같이 표시됩니다.

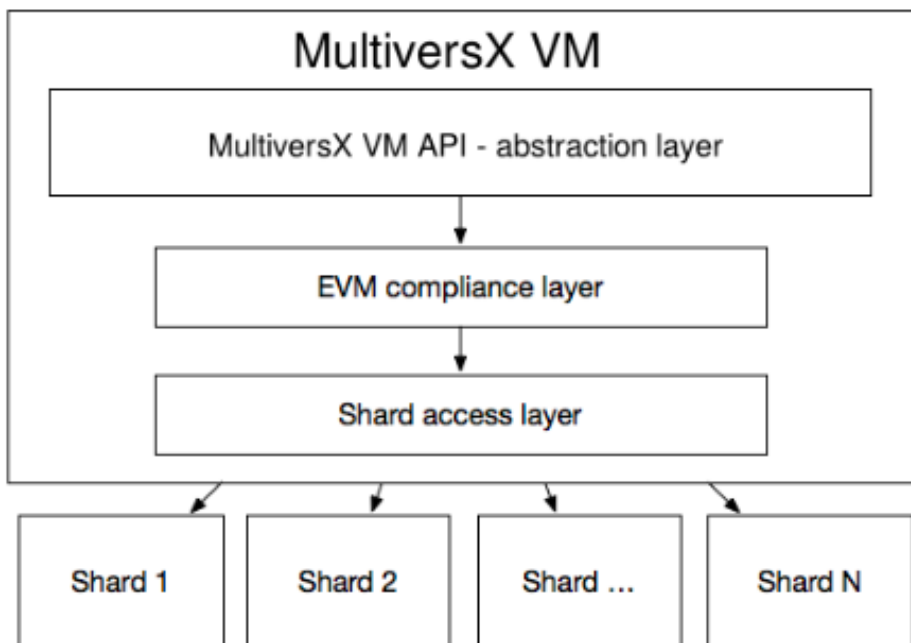


그림 7: 스마트 컨트랙트를 위한 추상화 계층

MultiversX에서는 코스모스(Cosmos) [38]에서 제안된 것처럼 VM 수준에서 어댑터 메커니즘을 사용해 크로스체인 상호운용성을 구현할 수 있습니다. 이 접근은 특수 어댑터와, MultiversX와 상호운용할 각 체인마다 어댑터 스마트 컨트랙트 간의 외부 통신 매체를 필요로 합니다. 가치 교환은 특수한 자산 수탁 스마트 컨트랙트를 통해 이루어지며, 이들은 외부 체인의 네이티브 토큰을 보관하고, MultiversX 네이티브 토큰을 발행할 수 있습니다.

1 VM 인프라

MultiversX는 K 프레임워크 위에 VM 인프라를 구축합니다. K 프레임워크는 실행 가능한 의미론적 프레임워크로, 프로그래밍 언어, 연산체계(calculi), 타입 시스템, 형식적 분석 도구 등을 정의할 수 있습니다 [39]. K 프레임워크를 사용하는 가장 큰 장점은 스마트 컨트랙트 언어를 모호함 없이 정의할 수 있다는 것이며, 이는 정의되지 않은 동작이나 탐지하기 어려운 버그 가능성을 제거합니다.

K 프레임워크는 실행 가능하다는 점에서, 언어의 의미론적 명세가 그 언어의 동작 인터프리터로 직접 사용될 수 있습니다. 즉, 프로그램을 K 프레임워크 코어 구현에 직접 실행할 수도 있고, 여러 프로그래밍 언어로 인터프리터를 생성할 수도 있습니다. 이를 “백엔드(backends)”라 부릅니다. 실행 속도와 상호운용성 편의를 위해, MultiversX는 자체 맞춤형 K 프레임워크 백엔드를 사용합니다.

2 스마트 컨트랙트 언어

K 프레임워크의 큰 장점 중 하나는, K로 정의된 언어에 대해 추가 프로그래밍 없이도 인터프리터를 생성할 수 있다는 것입니다. 이는 생성된 인터프리터가 “구조적으로 올바름(correct-by-construction)”을 의미합니다. 이미 K 프레임워크로 명세된 스마트 컨트랙트 언어나 현재 개발 중인 명세도 존재합니다.

MultiversX 네트워크는 세 가지 저수준 언어를 지원할 예정입니다: IELE VM, KEVM, WASM.

- **IELE VM**은 LLVM 스타일의 중간 수준 언어이지만 블록체인에 맞게 조정된 언어입니다. 이는 K로 직접 작성되었으며, K 프레임워크 외부에는 다른 명세나 구현이 존재하지 않습니다 [40]. 사람이 읽기 쉽고 빠르며, EVM의 몇 가지 한계를 극복하도록 설계되었습니다. MultiversX는 IELE의 변형 버전을 사용하며, 대부분의 변경 사항은 계정 주소 관리와 관련됩니다. 개발자는 IELE로 직접 스마트 컨트랙트를 작성할 수도 있지만, 대부분은 Solidity를 작성한 뒤 Solidity→IELE 컴파일러를 사용할 것입니다(그림 8 참조).
- **KEVM**은 K로 작성된 Ethereum Virtual Machine(EVM)의 버전입니다 [41]. K 버전에서는 EVM의 특정 취약점이 수정되거나 아예 제외됩니다.
- **WebAssembly(WASM)**는 스택 기반 가상 머신을 위한 이진 명령 형식으로, 스마트 컨트랙트 실행에 사용할 수 있습니다. WASM 인프라는 개발자가 C/C++, Rust, C#, 기타 언어로 스마트 컨트랙트를 작성할 수 있도록 지원합니다.

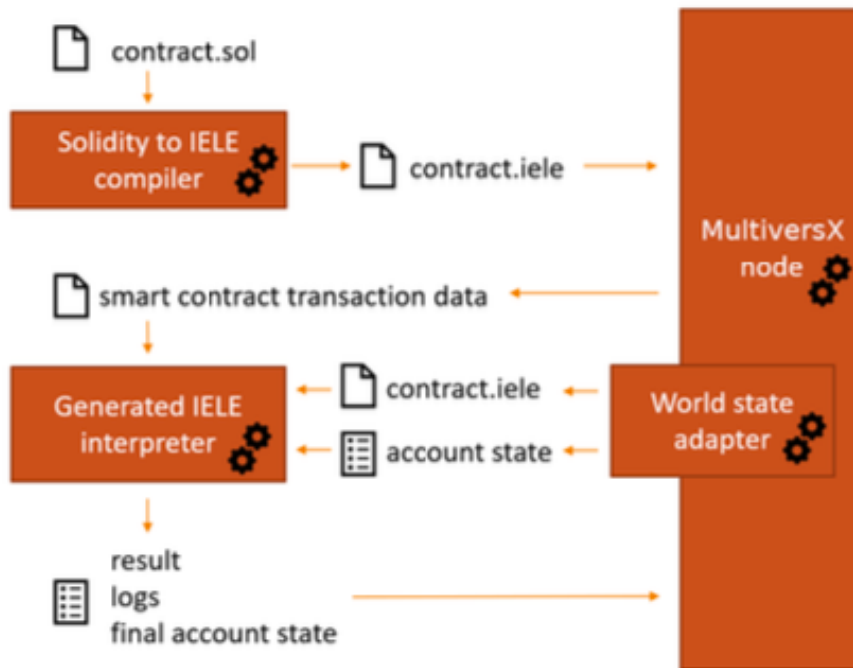


그림 8: MultiversX VM 실행

언어 명세와 인터프리터 생성은 절반의 과제일 뿐입니다. 나머지 절반은 생성된 인터프리터를 MultiversX 네트워크와 통합하는 것입니다. 이를 위해 우리는 공통 VM 인터페이스를 구축했으며, 이를 통해 어떤 VM 이든 MultiversX 노드에 플러그인할 수 있습니다(그림 9 참조). 각 VM은 이 인터페이스를 구현하는 어댑터를 가집니다. 각 스마트 컨트랙트는 컴파일된 VM의 바이트코드 형태로 저장되며, 해당 VM에서 실행됩니다.

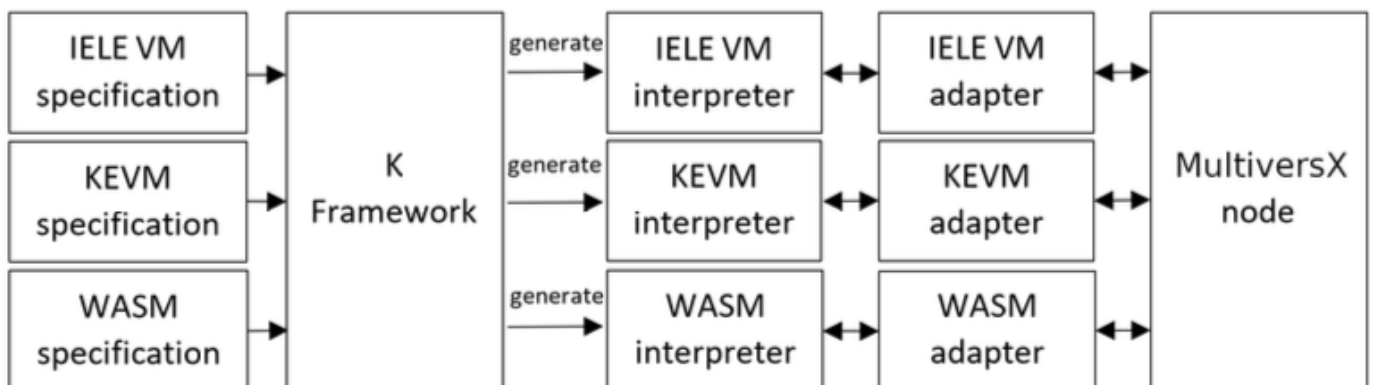


그림 9: MultiversX VM 구성 요소

3 형식 모델링 및 검증 지원

스마트 컨트랙트 언어가 K 프레임워크로 형식적으로 정의되었으므로, 이 언어로 작성된 스마트 컨트랙트에 대해 형식 검증(formal verification)을 수행할 수 있습니다. 이를 위해 요구사항도 형식적으로 모델링해야 하며, 이는 K 프레임워크를 사용해 수행할 수 있습니다 [42].

4 샤딩 아키텍처에서의 스마트 컨트랙트

샤딩된 아키텍처에서의 스마트 컨트랙트는 아직 초기 연구·개발 단계에 있으며 심각한 과제를 안고 있습니다. Atomix [7]나 S-BAC [9]와 같은 프로토콜이 출발점이 됩니다. 동적 스마트 컨트랙트 종속성은 SC를 동일 샤드로 이동시켜 해결할 수 없는데, 배포 시점에 모든 종속성을 계산할 수 없기 때문입니다.

현재 연구 중인 해결책은 다음과 같습니다:

1. 서로 다른 샤드에 위치한 스마트 컨트랙트를 원자적으로 실행할 수 있도록 하는 잠금 메커니즘. 이를 통해 관련 SC는 모두 동시에 실행되거나 전혀 실행되지 않도록 보장합니다. 이 과정은 다중 상호작용 메시지와 서로 다른 샤드의 합의 동기화를 필요로 합니다 [9].
2. Ethereum 2.0을 위한 크로스샤드 컨트랙트 인출(cross-shard contract yanking) 제안은 실행 시점에 스마트 컨트랙트 코드와 데이터를 호출자 샤드로 이동시킵니다. 원자적 실행은 필요 없으나, 이동된 SC는 잠금 메커니즘이 필수이며, 이는 해당 SC가 다른 트랜잭션을 위해 실행되는 것을 차단합니다. 이 방식은 단순하지만 SC의 전체 내부 상태를 전송해야 합니다 [43].

Ethereum 모델을 따라, MultiversX는 다음과 같은 트랜잭션 유형을 가집니다:

1. **SC 생성 및 배포:** 수신자 주소는 비어 있고, 데이터 필드에는 스마트 컨트랙트 코드가 바이트 배열로 포함됨.
2. **SC 메서드 호출:** 트랜잭션의 수신자 주소가 비어 있지 않으며, 해당 주소에 코드가 존재함.
3. **지불 트랜잭션:** 트랜잭션의 수신자 주소가 비어 있지 않으며, 해당 주소에는 코드가 없음.

MultiversX는 스마트 컨트랙트에 대해 비동기적 크로스샤드 실행 모델을 사용합니다. 사용자가 스마트 컨트랙트 실행 트랜잭션을 생성하면, 스마트 컨트랙트가 현재 샤드에 없을 경우 이는 지불 트랜잭션으로 처리됩니다. 이 경우 송신자의 계정에서 트랜잭션 금액이 차감되고, 송신 샤드 블록에 포함되며, 수신 샤드로 향하는 미니블록에 기록됩니다. 트랜잭션은 메타체인에서 검증된 뒤 수신 샤드에서 처리됩니다. 수신 샤드에서는 수신자 주소가 해당 샤드 내에 존재하는 스마트 컨트랙트이므로, 이를 SC 메서드 호출로 처리합니다. 스마트 컨트랙트 호출 시 송신 계정을 그림자처럼 반영하는 임시 계정이 생성되며, 트랜잭션 금액이 해당 계정 잔액으로 할당되고 스마트 컨트랙트가 실행됩니다. 실행 후 스마트 컨트랙트는 다른 샤드 계정을 포함한 여러 계정에 영향을 미치는 결과를 반환할 수 있습니다. 동일 샤드 계정에 영향을 주는 결과는 같은 라운드에서 즉시 실행됩니다. 그러나 다른 샤드 계정에 영향을 주는 경우, 스마트 컨트랙트 실행 출력을 저장하는 **스마트 컨트랙트 결과(SCR, Smart Contract Results)** 트랜잭션이 생성됩니다. SCR 미니블록은 각 대상 샤드에 대해 만들어지며, 이는 크로스샤드 트랜잭션과 동일하게 메타체인에서 검증된 뒤 해당 샤드에서 처리됩니다. 만약 한 스마트 컨트랙트가 실행 중 다른 샤드의 스마트 컨트랙트를 동적으로 호출하면, 이는 중간 결과로 저장되어 계정 처리와 동일한 방식으로 취급됩니다.

이 방식은 여러 단계를 거치며, 크로스샤드 스마트 컨트랙트 호출의 최종화에는 최소 5라운드가 필요합니다. 그러나 잠금 및 상태 이동은 필요하지 않습니다.

IX 부트스트래핑과 스토리지 (Bootstrapping and Storage)

1 타임라인 분할 (Timeline division)

지분증명(PoS) 시스템은 일반적으로 타임라인을 에폭으로 나누고, 각 에폭을 더 작은 라운드로 나눕니다 [19]. 타임라인과 용어는 아키텍처에 따라 다르지만, 대부분 유사한 접근을 사용합니다.

에폭(Epochs)

MultiversX 프로토콜에서 각 에폭은 고정된 기간(초기에는 24시간으로 설정되며, 여러 테스트넷 검증 단계를 거쳐 업데이트될 수 있음)을 가집니다. 이 시간 동안 샤드의 구성은 변하지 않습니다. 시스템은 에폭 간에 샤드 수를 조정하여 확장성 요구에 대응합니다. 담합을 방지하기 위해, 에폭이 끝난 후 각 샤드의 구성은 변경되어야 합니다. 모든 노드를 샤드 간에 완전히 재분배하면 가장 높은 수준의 보안을 제공하지만, 부트스트래핑 지연 때문에 시스템의 생존성을 저해합니다. 이 때문에 각 에폭이 끝날 때, 샤드에 속한 자격 검증자의 1/3 미만이 비결정적이고 균등하게 다른 샤드의 대기 리스트로 재분배됩니다.

새 에폭이 시작되기 전, 샤드로의 검증자 분포는 추가 통신 없이도 결정될 수 있으며, 이는 그림 10에 표시되어 있습니다.

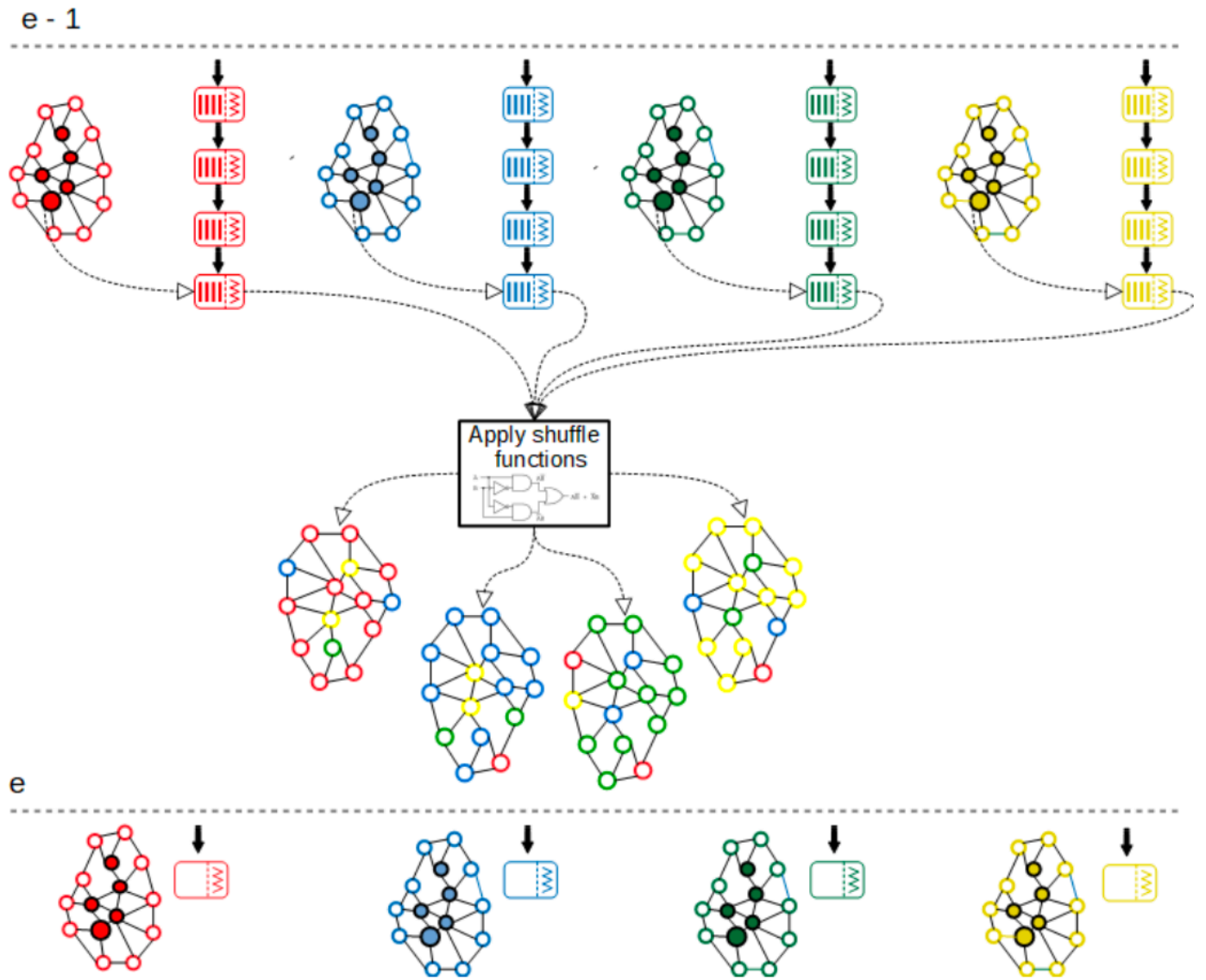


그림 10: 각 에폭 종료 시 노드 셔플링

노드 셔플링 과정은 여러 단계로 실행됩니다:

1. 현재 에폭 e_i 에서 등록된 새로운 노드는 에폭이 끝날 때까지 미할당 노드 풀에 남아 있습니다.

2. 각 샤드의 노드 중 $1/3$ 미만이 무작위로 선택되어 재배포 대상으로 지정되며, 할당 노드 풀에 추가됩니다.
3. 네트워크 노드 수 k_i 와 네트워크 사용량을 기반으로 새로운 샤드 수 $N_{sh,i+1}$ 가 계산됩니다.
4. 모든 샤드의 대기 리스트에 있었고 현재 동기화된 노드는 자격 검증자 리스트에 추가됩니다.
5. 미할당 노드 풀의 새로운 노드들은 에폭 e_{i+1} 동안 모든 샤드의 대기 리스트에 균등하게 무작위 분배됩니다.
6. 할당 노드 풀의 재배포 노드는, 에폭 e_{i+2} 에서 분할이 필요한 샤드의 대기 리스트로 더 높은 비율로 재분배됩니다.

라운드(Rounds)

각 라운드는 5초의 고정 시간(여러 테스트넷 검증 단계를 거쳐 업데이트될 수 있음)을 가집니다. 각 라운드 동안, 무작위로 선택된 블록 검증자 집합(블록 제안자 포함)이 각 샤드에서 새 블록을 생성할 수 있습니다. 라운드마다 이 집합은 자격 노드 리스트를 기반으로 변경됩니다(IV장에서 상세 설명됨).

앞서 설명했듯, 에폭 내 샤드 재구성과 라운드 내 임의의 검증자 선택은 불공정한 연합 형성을 억제하며, DDoS 및 뇌물 공격 가능성을 줄이고, 탈중앙성과 높은 트랜잭션 처리량을 유지합니다.

2 프루닝 (Pruning)

높은 처리량은 분산 원장이 빠르게 커지게 하며, 부트스트래핑 비용(시간 + 저장공간)을 증가시킵니다. 이는 XI.1절에서 강조됩니다.

이 비용은 블록체인의 전체 상태를 압축된 구조로 요약할 수 있는 효율적 프루닝 알고리즘을 사용함으로써 해결할 수 있습니다. 프루닝 메커니즘은 pBFT [15]의 안정적 체크포인트와 유사하며, 원장의 전체 상태를 압축합니다.

MultiversX 프로토콜은 효율적 프루닝 알고리즘 [7]을 사용합니다. 이를 자세히 설명하면, e 는 현재 에폭, a 는 현재 샤드라 할 때:

1. 샤드 노드는 e 의 계정 잔액을 머클 트리 [44]로 추적합니다.
2. 각 에폭이 끝나면, 블록 제안자는 상태 블록 $sb(a, e)$ 를 생성합니다. 여기에는 머클 트리 루트의 해시가 블록 헤더에 저장되고, 잔액은 블록 본문에 저장됩니다.
3. 검증자는 $sb(a, e)$ 를 검증하고 합의를 실행합니다.
4. 합의가 이루어지면, 블록 제안자는 $sb(a, e)$ 를 샤드 원장에 저장하며, 이는 에폭 $e + 1$ 의 제네시스 블록이 됩니다.
5. 에폭 $e + 1$ 이 끝나면, 노드는 $sb(a, e)$ 의 본문과 그 이전 모든 블록을 삭제합니다.

이 메커니즘을 사용하면, 새로운 노드의 부트스트래핑은 매우 효율적입니다. 실제로 이들은 마지막 유효 상태 블록에서 시작해 이후 블록만 계산하면 되며, 전체 히스토리를 재계산할 필요가 없습니다.

X 보안 평가 (Security Evaluation)

1 무작위성 소스 (Randomness source)

MultiversX는 무작위 수를 여러 운영에 사용합니다. 예: 블록 제안자와 검증자를 합의 그룹으로 무작위 샘플링하거나, 에폭 종료 시 노드를 샤드 간에 셔플링할 때 등입니다. 이러한 기능은 MultiversX의 보안 보장을 강화하므로, 편향될 수 없고(unbiasable), 예측 불가능한(provably unpredictable) 무작위 수를 사용하는 것이 중요합니다. 또한 무작위 수 생성은 효율적이어야 하며, 이는 확장 가능하고 높은 처리량의 블록체인 아키텍처에 적합해야 합니다.

이러한 속성은 BLS 서명 스킴과 같은 비대칭 암호 기법에서 찾을 수 있습니다. BLS의 중요한 특성 중 하나는 동일한 개인키로 동일한 메시지를 서명하면 항상 동일한 결과가 나온다는 것입니다. 이는 ECDSA에서 결정적 k 생성과 유사하며, 난수 매개변수를 사용하지 않기 때문에 가능합니다:

$$sig = sk \cdot H(m)$$

여기서 H 는 사용된 곡선 위의 점으로 매핑하는 해시 함수이고, sk 는 개인키입니다.

2 MultiversX에서의 무작위성 생성

매 라운드마다 무작위 수가 생성되며, 블록 제안자가 이를 블록에 추가합니다. 이는 각 무작위 수가 이전 무작위 소스에 대한 블록 제안자의 서명이므로 예측할 수 없습니다.

무작위 수 생성 과정은 합의 라운드의 일부로서 다음과 같이 상세히 설명됩니다:

1. 새로운 합의 그룹이 이전 블록 헤더의 무작위 소스를 사용하여 선택됩니다. 합의 그룹은 블록 제안자와 검증자로 구성됩니다.
2. 블록 제안자는 이전 무작위 소스를 BLS로 서명하고, 이를 제안된 블록 헤더에 새로운 무작위 소스로 추가한 뒤 블록을 합의 그룹에 브로드캐스트합니다.
3. 합의 그룹의 각 구성원은 블록 검증의 일부로 무작위 소스를 검증하고, 자신의 블록 서명을 블록 제안자에게 보냅니다.
4. 블록 제안자는 검증자의 블록 서명을 집계하고, 집계 서명과 새로운 무작위 소스를 포함한 블록을 샤드 전체에 브로드캐스트합니다.

라운드마다 진화하는 무작위 소스는 편향될 수 없고 검증 가능한 블록체인으로 볼 수 있으며, 각 무작위 수는 이전 무작위 수에 연결되고 이를 기반으로 검증할 수 있습니다.

3 “K” 블록 최종성 스킴 (“K” block finality scheme)

라운드 n 에서 서명된 블록은, 블록 $n + 1, n + 2, \dots, n + k$ 가 모두 서명된 경우에만 최종적(final)입니다. 또한 최종 블록은 되돌릴 수 없습니다. 메타체인은 한 샤드의 포크가 다른 샤드에 영향을 주지 않도록 최종 블록만 공증합니다. 샤드 또한 메타체인의 최종 블록만 고려하므로, 메타체인이 포크되더라도 영향을 받지 않습니다. 최종성과 정확성은 블록 생성 시와 블록 검증 시 모두 확인됩니다.

선택된 k 값은 10이며, 이는 최대 2블록 길이의 포크만 허용함을 의미합니다. 샤드 내 노드의 33%가 악의적 일지라도, 동일 라운드에서 동일 합의로 악의적 초다수($> \frac{2}{3} \cdot n + 1$)가 선택될 확률은 10^{-9} 입니다. 이 경우 그들은 블록 m 을 제안하고 서명할 수 있으나, 이는 메타체인에 의해 공증되지 않습니다. 메타체인이 블록 m 을 공증하려면, 블록 $m + 1$ 이 그 위에 생성되어야 합니다. $m + 1$ 을 생성하기 위해서는 다음 합의 그룹이 블록 m 을 승인해야 하며, 이는 악의적 그룹일 경우에만 가능합니다. 따라서 연속적으로 또 다른 악의

적 초다수가 선택될 확률은 다시 10^{-9} ($5.38 \cdot 10^{-10}$ 정확히)입니다. 두 개의 연속된 악의적 블록이 서명될 확률은 결과적으로 10^{-18} 입니다. 또한, 연속적으로 선택된 그룹은 담합해야 하므로, 그렇지 않다면 블록은 서명되지 않습니다.

4 피셔맨 챌린지 (Fisherman challenge)

악의적 다수가 무효 블록을 제안하면, 샤드 상태 루트는 잘못된 결과로 변조됩니다(무효 변경이 상태 트리에 포함됨). 계정들의 결합 머클 증명을 제공함으로써, 정직한 노드는 증거와 함께 챌린지를 제기할 수 있습니다. 정직한 노드는 트랜잭션 블록, 챌린지 블록 적용 전 영향을 받은 계정들의 축소된 머클 트리, 스마트 컨트랙트 상태를 제공하여 무효 트랜잭션/상태를 입증할 수 있습니다. 제한된 시간 내 증거와 함께 챌린지가 제기되지 않으면, 해당 블록은 유효한 것으로 간주됩니다. 무효 챌린지의 비용은 챌린지를 제기한 노드의 전체 스테이크입니다.

메타체인은 챌린지와 증거를 통해 불일치를 탐지합니다. 무효 트랜잭션이나 무효 상태 루트가 발생한 경우, 합의 그룹은 슬래싱될 수 있습니다. 챌린저는 슬래싱된 일부 금액을 보상으로 받을 수 있습니다. 또 다른 문제는 악의적 그룹이 무효 블록을 정직한 노드로부터 숨기는 경우입니다. 하지만 현재 합의 그룹이 생성한 블록을 형제 샤드와 옴저버 노드에 전송하도록 의무화하면, 데이터는 숨겨질 수 없습니다. 통신 오버헤드는 형제 샤드로 intra-shard 미니블록만 전송하여 줄입니다. 크로스샤드 미니블록은 항상 관심 있는 노드가 접근 가능한 별도 토픽으로 전송됩니다. 결국, 여러 정직한 노드가 챌린지를 제기할 수 있습니다. 또 다른 보안 강화는 P2P 토픽 설정에 의해 제공됩니다. 샤드에서 메타체인으로의 통신은 정의된 토픽/채널을 통해서만 이루어지며, 정직한 검증자는 이를 청취할 수 있습니다. 메타체인은 다른 채널에서 오는 메시지는 수락하지 않습니다. 이 솔루션은 챌린지가 발생할 경우 메타체인에 일부 지연을 초래할 수 있지만, 챌린지는 매우 드물고, 발견 확률이 높으며, 실패 시 노드는 전체 스테이크를 잃게 되므로 발생 가능성은 극히 낮습니다.

5 샤드 재조직 (Shard reorganization)

각 에폭 이후, 각 샤드의 노드 중 1/3 미만이 균등하고 비결정적으로 다른 샤드에 재분배됩니다. 이는 담합을 방지합니다. 이 방식은 재분배된 노드가 부트스트래핑 오버헤드를 추가하지만, 이들이 재분배된 에폭 동안은 합의에 참여하지 않으므로 생존성은 영향을 받지 않습니다. IX.2절에서 설명했듯, 프루닝 메커니즘이 이 시간을 현실적인 수준으로 줄여줍니다.

6 합의 그룹 선택 (Consensus group selection)

각 라운드가 끝나면, 마지막으로 커밋된 블록의 난수 시드, 현재 라운드, 자격 노드 리스트를 사용해 새로운 검증자 집합이 선택됩니다. 메시지 전파 지연으로 인한 네트워크 비동기화 상황에서는, 프로토콜이 복구 메커니즘을 갖추고 있어, 라운드 r 과 마지막 블록의 난수 시드를 모두 고려해 새로운 합의 그룹을 선택합니다. 이를 통해 포크를 방지하고 마지막 블록에 동기화할 수 있습니다. 검증자 그룹이 알려지는 시간 창(라운드 시간)이 매우 짧아, 공격 벡터를 최소화합니다.

7 노드 평판 (Node rating)

스테이크 외에도, 자격 검증자의 평판은 합의 그룹 선택 확률에 영향을 줍니다. 블록 제안자가 정직하고 그 블록이 블록체인에 커밋되면 평판은 증가합니다. 반대로 실패하면 평판은 감소합니다. 이렇게 각 검증자는 정직성을 유지하고, 최신 클라이언트 소프트웨어를 실행하며, 서비스 가용성을 높여 네트워크가 설계된 대로 기능하도록 유인됩니다.

8 샤드 중복성 (Shard redundancy)

트리의 최하위 레벨에서 형제 샤드로 분배된 노드(IV.4절 참조)는 서로의 블록체인 데이터와 애플리케이션 상태를 추적합니다. 샤드 중복성 개념을 도입함으로써, 네트워크 노드 수가 감소할 때 일부 형제 샤드가 병합되어야 할 경우, 대상 노드는 즉시 샤드 병합 과정을 시작합니다.

XI 실제 문제의 이해 (Understanding the real problems)

1 중앙화 vs 탈중앙화 (Centralized vs Decentralized)

블록체인은 처음에 중앙화된 금융 시스템의 대안으로 등장했습니다 [45]. 분산 아키텍처가 제공하는 자유와 익명성은 부정할 수 없는 장점이지만, 성능은 실제 환경에서 전 세계적 규모로 분석되어야 합니다.

가장 관련성 높은 성능 지표는 초당 트랜잭션 수(TPS)이며, 이는 표 2에서 볼 수 있습니다. 신뢰성과 효율성을 대규모로 검증받은 기존 중앙화 시스템과 새로운 탈중앙화 아키텍처의 TPS 비교는 객관적이지만 다소 불안한 현실을 보여줍니다 [46], [47], [48], [49].

아키텍처	유형 / 분산도	TPS (평균)	TPS (최대 한계)
VISA	중앙화	3500	55000
Paypal	중앙화	200	450
Ripple	허가형	1500	55000
NEO	혼합형	1000	10000
Ethereum	탈중앙화	15	25
Bitcoin	탈중앙화	2	7

표 2: 중앙화 vs 탈중앙화 TPS 비교

블록체인 아키텍처의 확장성은 여전히 해결되지 않은 중요한 문제입니다. 예를 들어, 현 블록체인 아키텍처가 VISA 수준의 처리량에서 작동할 경우 발생할 데이터 저장과 부트스트래핑의 영향을 생각해봅시다. 이러한 사고 실험을 수행하면, 여러 2차적 문제들의 규모가 분명해집니다(그림 11 참조).

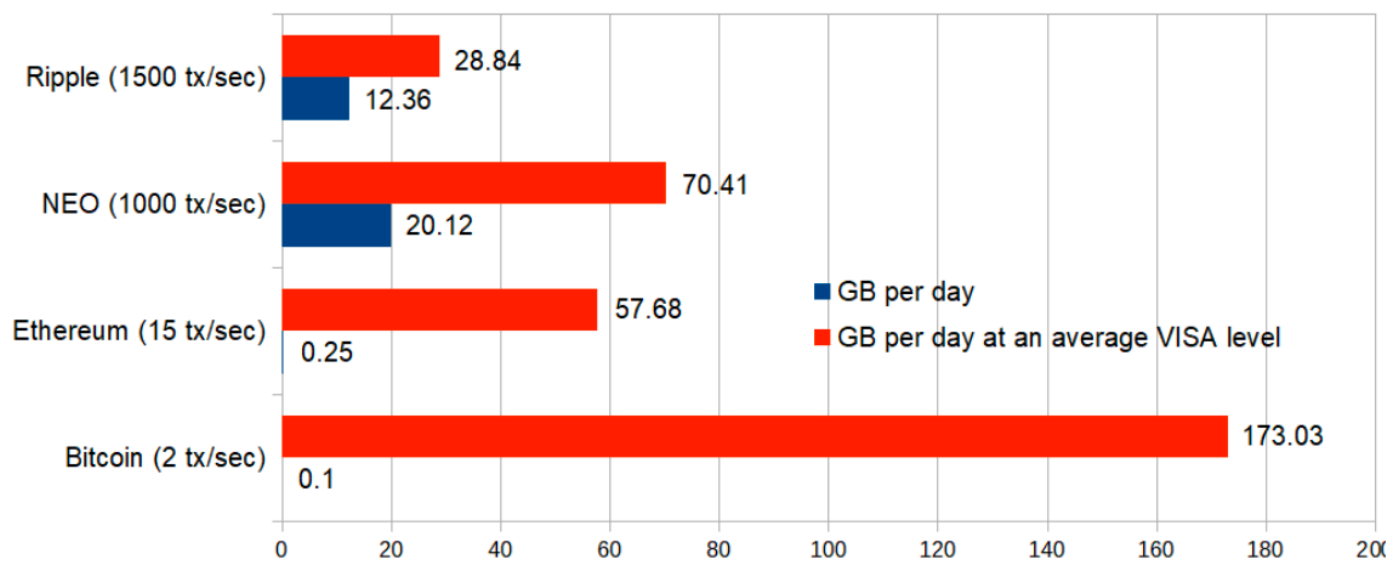


그림 11: VISA 평균 TPS에서 작동하는 검증된 분산 아키텍처의 스토리지 추정치

XII 블록체인 성능 패러다임 (The blockchain performance paradigm)

블록체인 기반 분산 아키텍처를 설계하는 과정에는 여러 도전 과제가 있습니다. 아마도 가장 어려운 과제 중 하나는 맥락적 압력 조건(contextual pressure conditions) 하에서 운영성을 유지하는 것입니다. 성능 압력을 결정하는 주요 요소는 다음과 같습니다:

- 복잡성
- 시스템 규모
- 트랜잭션 볼륨

복잡성(Complexity)

시스템 성능을 제한하는 첫 번째 요소는 합의 프로토콜입니다. 더 복잡한 프로토콜일수록 병목이 커집니다. PoW 합의 아키텍처에서는, 시스템을 탈중앙화 상태로 유지하고 ASIC 내성을 보장하기 위해 채굴 복잡성이 크게 성능 패널티를 유발합니다 [50]. 이를 극복하기 위해 PoS는 네트워크 관리 복잡성을 줄이는 대신, 네트워크의 일부에 연산 집종을 허용하지만, 제어 메커니즘에 더 큰 복잡성을 초래합니다.

시스템 규모(System size)

기존 검증된 아키텍처에 노드 수를 확장하면 성능 저하가 심각하게 발생하며, 계산 비용이 증가합니다. 샤딩은 좋은 접근으로 보이지만, 샤드 크기가 중요한 역할을 합니다. 작은 샤드는 민첩하지만 악의적 그룹의 영향을 받을 가능성이 높고, 큰 샤드는 더 안전하지만 재구성이 시스템 생존성에 영향을 줍니다.

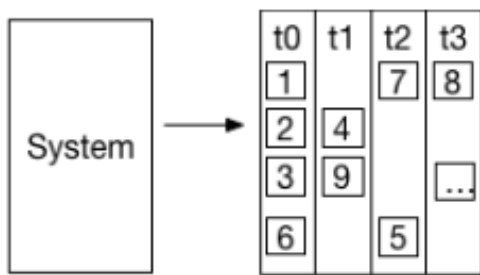
트랜잭션 볼륨(Transaction volume)

다른 요소보다 더 중요한 마지막 항목은 트랜잭션 처리 성능입니다. 이 기준의 영향을 정확히 측정하려면, 다음 두 가지 관점에서 분석해야 합니다:

- **C1 트랜잭션 처리량(throughput)** – 단위 시간당 시스템이 처리할 수 있는 트랜잭션 수(TPS), 즉 시스템의 출력 [51].
- **C2 트랜잭션 최종성(finality)** – 특정 트랜잭션이 처리되는 속도, 즉 시작에서 최종화까지 걸리는 시간.

C1. 트랜잭션 처리량

단일 체인 아키텍처의 TPS는 매우 낮으며, 사이드체인 같은 우회 방식을 통해 증가시킬 수 있습니다 [52]. 샤딩 아키텍처(우리의 접근처럼)에서는, TPS는 샤드 수, 검증자/블록 제안자의 계산 능력, 메시징 인프라에 의해 좌우됩니다 [8]. 일반적으로 그림 13에서처럼, 이는 대중에게 잘 인식됩니다. 하지만 이 지표는 중요성에도 불구하고 단편적 시각만 제공합니다.

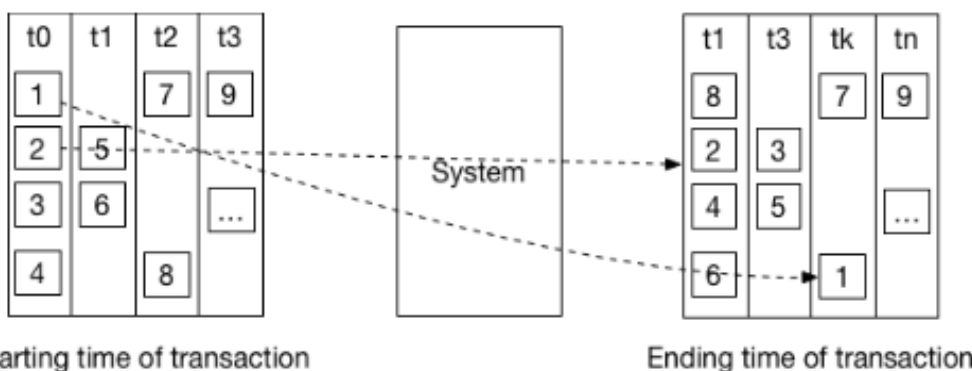


Processed transaction per time unit

그림 13: 트랜잭션 처리량

C2. 트랜잭션 최종성

보다 섬세한 측면은, 시스템이 TPS 1000을 처리할 수 있더라도, 특정 트랜잭션 하나를 처리하는 데는 시간이 걸릴 수 있다는 점입니다. TPS 외에도, 최종성은 검증자/블록 제안자의 계산 능력, 메시징 인프라, 그리고 특히 분배 알고리즘(결정 시점)과 라우팅 프로토콜(실행 위치)에 의해 좌우됩니다. 많은 아키텍처가 이 점을 언급하지 않지만, 사용자 입장에서는 극히 중요한 요소입니다. 그림 14에서는, 특정 트랜잭션이 시작부터 끝까지 실행되는 데 필요한 총 시간을 고려합니다.



Starting time of transaction

Ending time of transaction

그림 14: 트랜잭션 최종성

MultiversX에서는 분배 메커니즘(제V장에서 상세히 설명됨)이 트랜잭션을 올바른 샤드로 직접 라우팅함으로써, 전체 지연을 줄여 최종성을 개선합니다.

XII 블록체인 성능 패러다임 (The blockchain performance paradigm)

블록체인 기반 분산 아키텍처 설계 과정에는 여러 도전 과제가 있습니다. 아마도 가장 어려운 과제 중 하나는 맥락적 압력 조건(contextual pressure conditions) 하에서 운영성을 유지하는 것입니다. 성능 압력을 결정하는 주요 구성 요소는 다음과 같습니다:

- 복잡성(Complexity)
- 시스템 규모(System size)
- 트랜잭션 볼륨(Transaction volume)

복잡성(Complexity)

시스템 성능을 제한하는 첫 번째 요소는 합의 프로토콜입니다. 더 복잡한 프로토콜일수록 병목 현상이 커집니다. PoW 합의 아키텍처에서는, 시스템을 탈중앙화 상태로 유지하고 ASIC 내성을 보장하기 위해 채굴 복잡성이 크게 성능 패널티를 유발합니다 [50]. 이를 극복하기 위해 PoS는 네트워크 관리 복잡성을 줄이는 대신, 네트워크의 일부에 연산을 집중시키지만 제어 메커니즘에 더 큰 복잡성을 초래합니다.

시스템 규모(System size)

기존 검증된 아키텍처에서 노드 수를 확장하면 성능 저하가 심각하게 발생하며, 계산 비용이 증가합니다. 샤딩은 좋은 접근으로 보이지만, 샤드 크기가 중요한 역할을 합니다. 작은 샤드는 민첩하지만 악의적 그룹의 영향을 받을 가능성이 더 크고, 큰 샤드는 더 안전하지만 재구성이 시스템 생존성(liveness)에 영향을 줍니다.

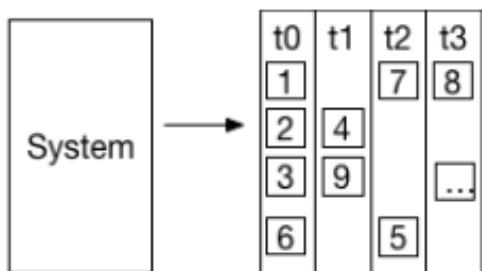
트랜잭션 볼륨(Transaction volume)

다른 요소보다 더 중요한 마지막 항목은 트랜잭션 처리 성능입니다. 이 기준의 영향을 정확히 측정하려면, 다음 두 가지 관점에서 분석해야 합니다:

- **C1 트랜잭션 처리량(throughput)** – 단위 시간당 시스템이 처리할 수 있는 트랜잭션 수(TPS), 즉 시스템의 출력 [51].
- **C2 트랜잭션 최종성(finality)** – 특정 트랜잭션이 처리되는 속도, 즉 시작에서 최종화까지 걸리는 시간.

C1. 트랜잭션 처리량

단일 체인 아키텍처의 TPS는 매우 낮으며, 사이드체인 같은 우회 방식을 통해 증가시킬 수 있습니다 [52]. 샤딩 아키텍처(우리의 접근처럼)에서는, TPS는 샤드 수, 검증자/블록 제안자의 계산 능력, 메시징 인프라에 의해 좌우됩니다 [8]. 일반적으로 그림 13에서처럼, 이는 대중에게 잘 인식됩니다. 하지만 이 지표는 중요성에도 불구하고 단편적 시각만 제공합니다.



Processed transaction per time unit

그림 13: 트랜잭션 처리량

C2. 트랜잭션 최종성

보다 섬세한 측면은, 시스템이 TPS 1000을 처리할 수 있더라도, 특정 트랜잭션 하나를 처리하는 데는 시간이 걸릴 수 있다는 점입니다. TPS 외에도, 최종성은 검증자/블록 제안자의 계산 능력, 메시징 인프라, 그리고 특히 분배 알고리즘(결정 시점)과 라우팅 프로토콜(실행 위치)에 의해 좌우됩니다. 많은 아키텍처가 이 점을 언급하지 않지만, 사용자 입장에서는 극히 중요한 요소입니다. 그림 14에서는, 특정 트랜잭션이 시작부터 끝까지 실행되는 데 필요한 총 시간을 고려합니다.

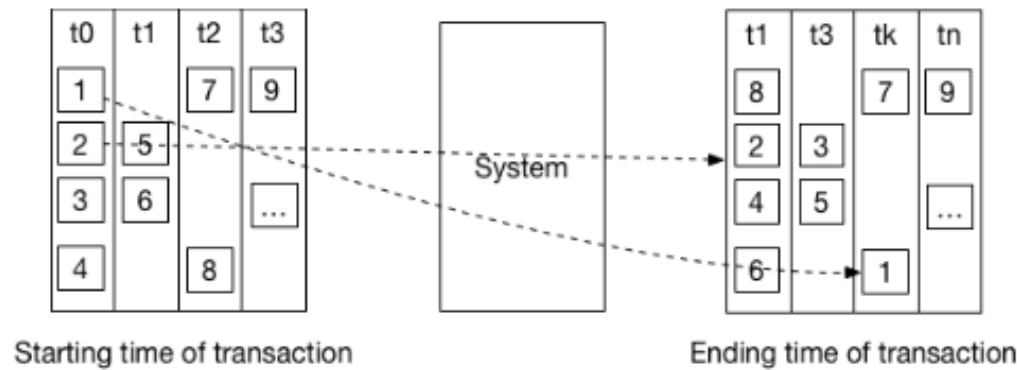


그림 14: 트랜잭션 최종성

MultiversX에서는 분배 메커니즘(제V장에서 상세히 설명됨)이 트랜잭션을 올바른 샤드로 직접 라우팅함으로써, 전체 지연을 줄여 최종성을 개선합니다.

XIII 결론 (Conclusion)

1 성능 (Performance)

그림 12에 제시된 성능 테스트와 시뮬레이션은, 고확장성 분산 원장으로서 이 솔루션의 효율성을 반영합니다. 네트워크에 더 많은 노드가 합류할수록, 우리의 샤딩 접근은 처리량이 선형적으로 증가함을 보여줍니다. 선택된 합의 모델은 여러 차례의 통신 라운드를 포함하므로, 결과는 네트워크 품질(속도, 지연, 가용성)에 크게 영향을 받습니다. 우리의 테스트넷 시뮬레이션은 전 세계 평균 네트워크 속도를 사용했으며, 이론적 최대 한계에서 MultiversX가 단 2개의 샤드만으로 평균 VISA 수준을 초과하고, 16개의 샤드에서는 VISA의 최고치에 접근함을 시사합니다.

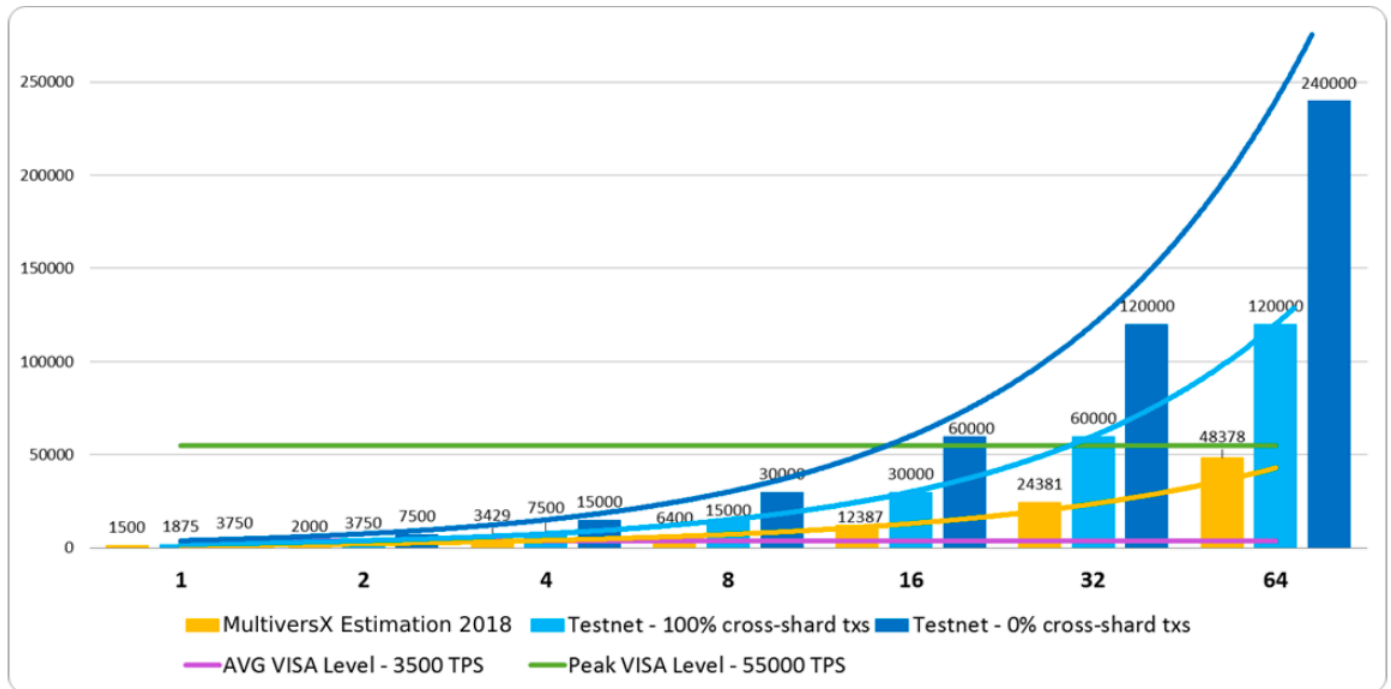


그림 12: 글로벌 네트워크 속도 8MB/s에서 초당 트랜잭션 수로 측정된 네트워크 처리량

2 진행 중 및 향후 연구 (Ongoing and future research)

우리 팀은 MultiversX 설계를 지속적으로 재평가하고 개선하여, 이를 가장 설득력 있는 퍼블릭 블록체인 아키텍처 중 하나로 만들고자 합니다. 우리의 목표는, 적응형 상태 샤딩을 통해 확장성을 해결하면서도, Secure Proof of Stake 합의 메커니즘을 통해 보안과 에너지 효율을 유지하는 것입니다. 향후 연구 방향은 다음과 같습니다:

1. **강화학습(Reinforcement learning)**: 자주 거래하는 클라이언트를 동일한 샤드에 배치하여 전체 비용을 줄이는 방식으로 샤딩 과정의 효율성을 높이는 것을 목표로 함.
2. **AI 감독(AI supervision)**: 악의적 행위 패턴을 탐지하는 AI 감독자를 생성. 이 기능을 프로토콜에 통합하되, 탈중앙화를 훼손하지 않는 방식은 아직 불확실함.
3. **합의 요소로서의 신뢰성(Reliability)**: 기존 프로토콜은 스테이크와 평판 사이에서 가중치를 두지만, 최근 블록 이력에 대한 합의를 통해 분산적으로 계산되는 신뢰성 지표를 추가할 계획.
4. **크로스체인 상호운용성(Cross-chain interoperability)**: Decentralized Identity Foundation [53]이나 Blockchain Interoperability Alliance [54]가 시작한 표준을 구현하고 기여할 예정.
5. **프라이버시 보존 트랜잭션(Privacy preserving transactions)**: 참가자의 신원을 보호하면서도 프라이버시를 유지한 채 감사 기능을 제공하기 위해, 지식증명 기반 영지식 증명(ZK-SNARKs, Zero-Knowledge Succinct Non-Interactive Argument of Knowledge) [55]를 사용할 예정.

3 종합 결론 (Overall Conclusions)

MultiversX는 진정한 상태 샤딩 아키텍처 위에 새롭게 제안된 Secure Proof of Stake 알고리즘을 적용하여, VISA 수준의 처리량과 초 단위 확인 시간을 달성하는 최초의 고확장성 퍼블릭 블록체인입니다. MultiversX의 적응형 상태 샤딩 접근은 Omniledger의 제안을 개선하여 보안성과 처리량을 높이며, 내장된 자동 트랜잭션 라우팅과 상태 중복 메커니즘을 통해 지연을 크게 줄입니다. 또한 샤드 프루닝 기법을 사

용하여, 다른 접근 방식과 비교했을 때 부트스트래핑과 저장 비용을 상당히 줄입니다. 새롭게 도입된 Secure Proof of Stake 합의 알고리즘은 분산적 공정성을 보장하고, Algorand의 무작위 선택 아이디어를 개선하여 합의 그룹 무작위 선택 시간을 12초에서 100ms로 단축합니다. 상태 샤딩과 효율적인 Secure Proof of Stake 합의 알고리즘의 결합은 초기 추정치에서 유망한 결과를 보여주었으며, 이는 최근 테스트넷 결과로 검증되었습니다.

참고문헌 (References)

- [1] G. Hileman and M. Rauchs, “2017 Global Cryptocurrency Benchmarking Study,” Social Science Research Network, Rochester, NY, SSRN Scholarly Paper ID 2965436, Apr. 2017. [Online]. Available: <https://papers.ssrn.com/abstract=2965436>
- [2] “The Ethereum Wiki - Sharding FAQ,” 2018, original-date: 2014-02-14T23:05:17Z. [Online]. Available: <https://github.com/ethereum/wiki/wiki/Sharding-FAQ>
- [3] Y. Gilad, R. Hemo, S. Micali, G. Vlachos, and N. Zeldovich, “Algorand: Scaling Byzantine Agreements for Cryptocurrencies,” in Proceedings of the 26th Symposium on Operating Systems Principles, ser. SOSP '17. New York, NY, USA: ACM, 2017, pp. 51–68. [Online]. Available: <http://doi.acm.org/10.1145/3132747.3132757>
- [4] D. Boneh, B. Lynn, and H. Shacham, “Short signatures from the weil pairing,” in Advances in Cryptology – ASIACRYPT '01, LNCS. Springer, 2001, pp. 514–532.
- [5] D. Boneh, M. Drijvers, and G. Neven, “Compact multi-signatures for smaller blockchains,” in Advances in Cryptology – ASIACRYPT 2018, ser. Lecture Notes in Computer Science, vol. 11273. Springer, 2018, pp. 435–464.
- [6] V. Buterin, “Ethereum: A Next-Generation Smart Contract and Decentralized Application Platform,” 2013. [Online]. Available: <https://www.ethereum.org/pdfs/EthereumWhitePaper.pdf>
- [7] E. Kokoris-Kogias, P. Jovanovic, L. Gasser, N. Gailly, E. Syta, and B. Ford, “OmniLedger: A Secure, Scale-Out, Decentralized Ledger via Sharding,” Tech. Rep. 406, 2017. [Online]. Available: <https://eprint.iacr.org/2017/406>
- [8] “The ZILLIQA Technical Whitepaper,” 2017. [Online]. Available: <https://docs.zilliqa.com/whitepaper.pdf>
- [9] M. Al-Bassam, A. Sonnino, S. Bano, D. Hrycyszyn, and G. Danezis, “Chainspace: A Sharded Smart Contracts Platform,” arXiv:1708.03778 [cs], Aug. 2017. [Online]. Available: <http://arxiv.org/abs/1708.03778>

- [10] G. Wood, "Ethereum: A Secure Decentralised Generalised Transaction Ledger," 2017. [Online]. Available: <https://ethereum.github.io/yellowpaper/paper.pdf>
- [11] "Solidity — Solidity 0.4.21 documentation." [Online]. Available: <https://solidity.readthedocs.io/en/v0.4.21/>
- [12] "web3j," 2018. [Online]. Available: <https://github.com/web3j>
- [13] "Casper," 2018. [Online]. Available: <http://ethresear.ch/c/casper>
- [14] "The State of Ethereum Scaling, March 2018 – Highlights from EthCC on Plasma Cash, Minimum Viable Plasma, and More... – Medium," 2018. [Online]. Available: <https://medium.com/loom-network/the-state-of-ethereum-scaling-march-2018-74ac08198a36>
- [15] M. Castro and B. Liskov, "Practical Byzantine Fault Tolerance," in Proceedings of the Third Symposium on Operating Systems Design and Implementation, ser. OSDI '99. Berkeley, CA, USA: USENIX Association, 1999, pp. 173–186. [Online]. Available: <http://dl.acm.org/citation.cfm?id=296806.296824>
- [16] Y. Jia, "Op Ed: The Many Faces of Sharding for Blockchain Scalability," 2018. [Online]. Available: <https://bitcoinmagazine.com/articles/op-ed-many-faces-sharding-blockchain-scalability/>
- [17] "Using Merkliz tree to shard block validation | Deadalnix's den," 2016. [Online]. Available: <https://www.deadalnix.me/2016/11/06/using-merkliz-tree-to-shard-block-validation/>
- [18] S. Nakamoto, "Bitcoin: A Peer-to-Peer Electronic Cash System," p. 9, 2008.
- [19] "Why we are building Cardano – Introduction." [Online]. Available: <https://why.cardano.com/>
- [20] "Constellation – a blockchain microservice operating system – White Paper," 2017. [Online]. Available: <https://github.com/Constellation-Labs/Whitepaper>
- [21] "Bitshares – Delegated Proof-of-Stake Consensus," 2014. [Online]. Available: <https://bitshares.org/technology/delegated-proof-of-stake-consensus/>
- [22] dantheman, "DPOS Consensus Algorithm – The Missing White Paper," May 2017. [Online]. Available: <https://steemit.com/dpos/@dantheman/dpos-consensus-algorithm-this-missing-white-paper>
- [23] "EOS.IO Technical White Paper v2," 2018. [Online]. Available: <https://github.com/EOSIO/Documentation/blob/master/TechnicalWhitePaper.md>

- [24] C. Paar and J. Pelzl, *Understanding Cryptography: A Textbook for Students and Practitioners*. Berlin Heidelberg: Springer-Verlag, 2010. [Online]. Available: <https://www.springer.com/gp/book/9783642041006>
- [25] C. Schnorr, "Efficient signature generation by smart cards," *Journal of Cryptology*, vol. 4, pp. 161–174, Jan. 1991.
- [26] K. Michaelis, C. Meyer, and J. Schwenk, "Randomly Failed! The State of Randomness in Current Java Implementations," in *Topics in Cryptology – CT-RSA 2013*, ser. Lecture Notes in Computer Science. Springer, Berlin, Heidelberg, Feb. 2013, pp. 129–144. [Online]. Available: https://link.springer.com/chapter/10.1007/978-3-642-36095-4_9
- [27] D. J. Bernstein, P. Birkner, M. Joye, T. Lange, and C. Peters, "Twisted Edwards Curves," in *Progress in Cryptology – AFRICACRYPT 2008*, ser. Lecture Notes in Computer Science. Springer, Berlin, Heidelberg, Jun. 2008, pp. 389–405. [Online]. Available: https://link.springer.com/chapter/10.1007/978-3-540-68164-9_26
- [28] A. Poelstra, "Schnorr Signatures are Non-Malleable in the Random Oracle Model," 2014. [Online]. Available: <https://download.wpsoftware.net/bitcoin/wizardry/schnorr-mall.pdf>
- [29] C. Decker and R. Wattenhofer, "Bitcoin Transaction Malleability and MtGox," *arXiv:1403.6676 [cs]*, vol. 8713, pp. 313–326, 2014. [Online]. Available: <http://arxiv.org/abs/1403.6676>
- [30] G. Maxwell, A. Poelstra, Y. Seurin, and P. Wuille, "Simple Schnorr Multi-Signatures with Applications to Bitcoin," *Tech. Rep. 068*, 2018. [Online]. Available: <https://eprint.iacr.org/2018/068>
- [31] Y. Seurin, "On the Exact Security of Schnorr-Type Signatures in the Random Oracle Model," in *Advances in Cryptology – EUROCRYPT 2012*, ser. Lecture Notes in Computer Science. Springer, Berlin, Heidelberg, Apr. 2012, pp. 554–571. [Online]. Available: https://link.springer.com/chapter/10.1007/978-3-642-29011-4_33
- [32] K. Itakura and K. Nakamura, "A public-key cryptosystem suitable for digital multisignatures," 1983.
- [33] S. Micali, K. Ohta, and L. Reyzin, "Accountable-subgroup Multisignatures: Extended Abstract," in *Proceedings of the 8th ACM Conference on Computer and Communications Security*, ser. CCS '01. New York, NY, USA: ACM, 2001, pp. 245–254. [Online]. Available: <http://doi.acm.org/10.1145/501983.502017>

- [34] T. Ristenpart and S. Yilek, “The Power of Proofs-of-Possession: Securing Multiparty Signatures against Rogue-Key Attacks,” in *Advances in Cryptology – EUROCRYPT 2007*, ser. Lecture Notes in Computer Science. Springer, Berlin, Heidelberg, May 2007, pp. 228–245. [Online]. Available: https://link.springer.com/chapter/10.1007/978-3-540-72540-4_13
- [35] M. Bellare and G. Neven, “Multi-signatures in the Plain public-Key Model and a General Forking Lemma,” in *Proceedings of the 13th ACM Conference on Computer and Communications Security*, ser. CCS ’06. New York, NY, USA: ACM, 2006, pp. 390–399. [Online]. Available: <http://doi.acm.org/10.1145/1180405.1180453>
- [36] D.-P. Le, A. Bonnet, and A. Gabillon, “Multisignatures as Secure as the Diffie-Hellman Problem in the Plain Public-Key Model,” in *Pairing-Based Cryptography – Pairing 2009*, ser. Lecture Notes in Computer Science. Springer, Berlin, Heidelberg, Aug. 2009, pp. 35–51. [Online]. Available: https://link.springer.com/chapter/10.1007/978-3-642-03298-1_3
- [37] T. Dickerson, P. Gazzillo, M. Herlihy, and E. Koskinen, “Adding Concurrency to Smart Contracts,” in *Proceedings of the ACM Symposium on Principles of Distributed Computing*, ser. PODC ’17. New York, NY, USA: ACM, 2017, pp. 303–312. [Online]. Available: <http://doi.acm.org/10.1145/3087801.3087835>
- [38] J. Kwon and E. Buchman, “Cosmos Network – Internet of Blockchains,” 2017. [Online]. Available: <https://cosmos.network/whitepaper>
- [39] G. Roşu and T. F. Şerbanuta, “An overview of the K semantic framework,” *The Journal of Logic and Algebraic Programming*, vol. 79, no. 6, pp. 397–434, 2010.
- [40] T. Kasampalis, D. Guth, B. Moore, T. Serbanuta, V. Serbanuta, D. Filaretti, G. Rosu, and R. Johnson, “Iele: An intermediate-level blockchain language designed and implemented using formal semantics,” Tech. Rep., 2018.
- [41] E. Hildenbrandt, M. Saxena, X. Zhu, N. Rodrigues, P. Daian, D. Guth, and G. Rosu, “Kevm: A complete semantics of the ethereum virtual machine,” Tech. Rep., 2017.
- [42] “How Formal Verification of Smart Contracts Works | RV Blog.” [Online]. Available: <https://runtimeverification.com/blog/how-formal-verification-of-smart-contracts-works/>
- [43] “Cross-shard contract yanking.” [Online]. Available: <https://ethresear.ch/t/cross-shard-contract-yanking/1450>

- [44] R. C. Merkle, "A Certified Digital Signature," in *Advances in Cryptology — CRYPTO' 89 Proceedings*, ser. Lecture Notes in Computer Science. Springer, New York, NY, Aug. 1989, pp. 218–238. [Online]. Available: https://link.springer.com/chapter/10.1007/0-387-34805-0_21
- [45] A. Veysov and M. Stolbov, "Financial System Classification: From Conventional Dichotomy to a More Modern View," Social Science Research Network, Rochester, NY, SSRN Scholarly Paper ID 2114842, Jul. 2012. [Online]. Available: <https://papers.ssrn.com/abstract=2114842>
- [46] "XRP – The Digital Asset for Payments." [Online]. Available: <https://ripple.com/xrp/>
- [47] "Visa – Annual Report 2017," 2018. [Online]. Available: https://s1.q4cdn.com/050606653/files/doc_financials/annual/2017/Visa-2017-Annual-Report.pdf
- [48] "PayPal Reports Fourth Quarter and Full Year 2017 Results (NASDAQ:PYPL)," 2018. [Online]. Available: <https://investor.paypal-corp.com/releasedetail.cfm?releaseid=1055924>
- [49] M. Schwarz, "Crypto Transaction Speeds 2018 – All the Major Cryptocurrencies," 2018. [Online]. Available: <https://www.abitgreedy.com/transaction-speed/>
- [50] "The Ethereum Wiki – Mining," 2018. [Online]. Available: <https://github.com/ethereum/wiki/wiki/Mining>
- [51] "Transaction throughput." [Online]. Available: https://docs.oracle.com/cd/E17276_01/html/programmer_reference/transapp_throughput.html
- [52] W. Martino, M. Quaintance, and S. Popejoy, "Chainweb: A Proof-of-Work Parallel-Chain Architecture for Massive Throughput," 2018. [Online]. Available: <http://kadana.io/docs/chainweb-v15.pdf>
- [53] "DIF – Decentralized Identity Foundation." [Online]. Available: <http://identity.foundation/>
- [54] H. I. World, "Blockchain Interoperability Alliance: ICON x Aion x Wanchain," Dec. 2017. [Online]. Available: <https://medium.com/helloiconworld/blockchain-interoperability-alliance-icon-x-aion-x-wanchain-8aeaafb3ebdd>
- [55] S. Goldwasser, S. Micali, and C. Rackoff, "The Knowledge Complexity of Interactive Proof-systems," in *Proceedings of the Seventeenth Annual ACM Symposium on Theory of Computing*, ser. STOC '85. New York, NY, USA: ACM, 1985, pp. 291–304. [Online]. Available: <http://doi.acm.org/10.1145/22145.22178>

