轻客户端原理与应用

MAP Protocol

Phil

2023.02

目录

01

轻客户端概述

全节点

轻节点

轻客户端用途

全节点 Full Node

- 一般区块链网络运行的都是Full Node，会通过p2p与其他节点建立连接。
- Full Node会从创世区块开始，接收链上所有区块，并执行验证链上所有的交易状态。
- Full Node需要较高的硬件配置，用于处理交易的执行，并保存所有的交易状态
- 目前在链TPS越来越高的情况下，Full Node的硬件配置要求越来越高

轻节点 Light Client Node

- Light Client Node也通过p2p与其他节点建立连接，但是不会去执行所有的交易，只同步consensus验证需要的最少内容，一般为block header；
- Light client不执行交易，只验证共识和区块的有效性；
- Light client不需要信任任何的其他节点，通过保存的block header以及提交proof，可以验证链上用户余额，交易，交易执行结果的有效性；

轻客户端用途

- 对硬件配置要求较低，可运行在嵌入式设备、移动设备甚至浏览器里面，提供去中心化钱包服务；

- 可运行在区块链上，通过智能合约实现轻客户端，提供去中心化跨链验证；

02

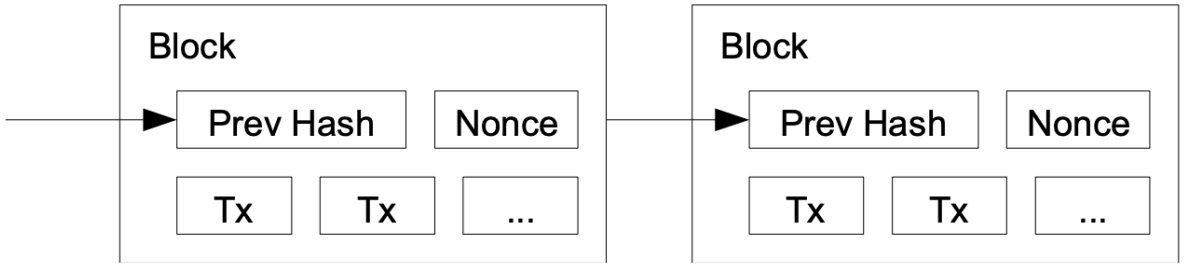
比特币SPV节点

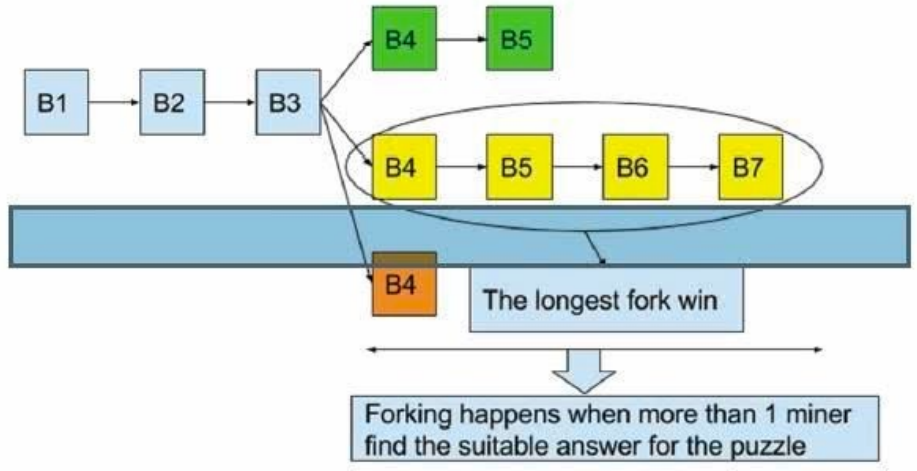
Nakamoto共识

SPV验证

Nakamoto共识

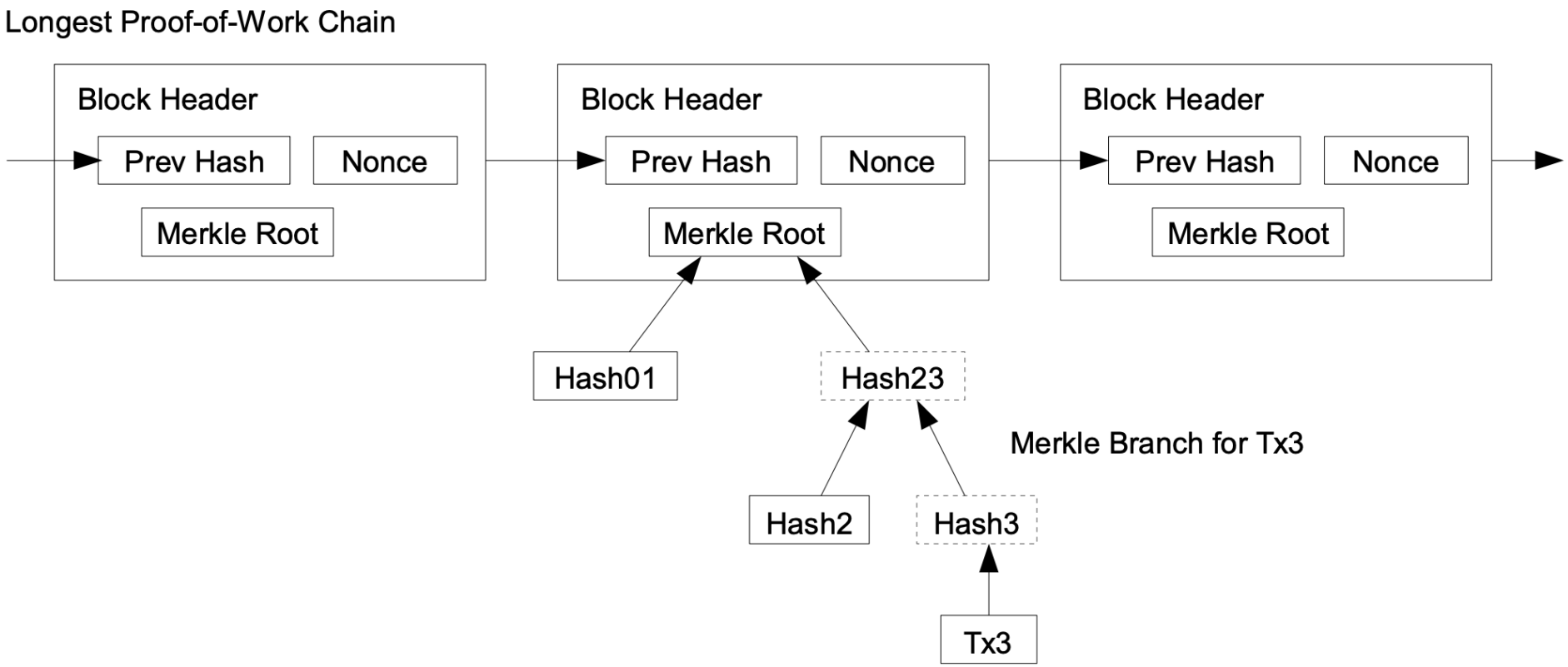
- PoW算法
  - 基于nonce，block header可自验证
- 难度算法
  - 区块难度每2016个区块调整
  - 可基于block header 时间戳验证区块难度
- 分叉算法
  - 最长链原则





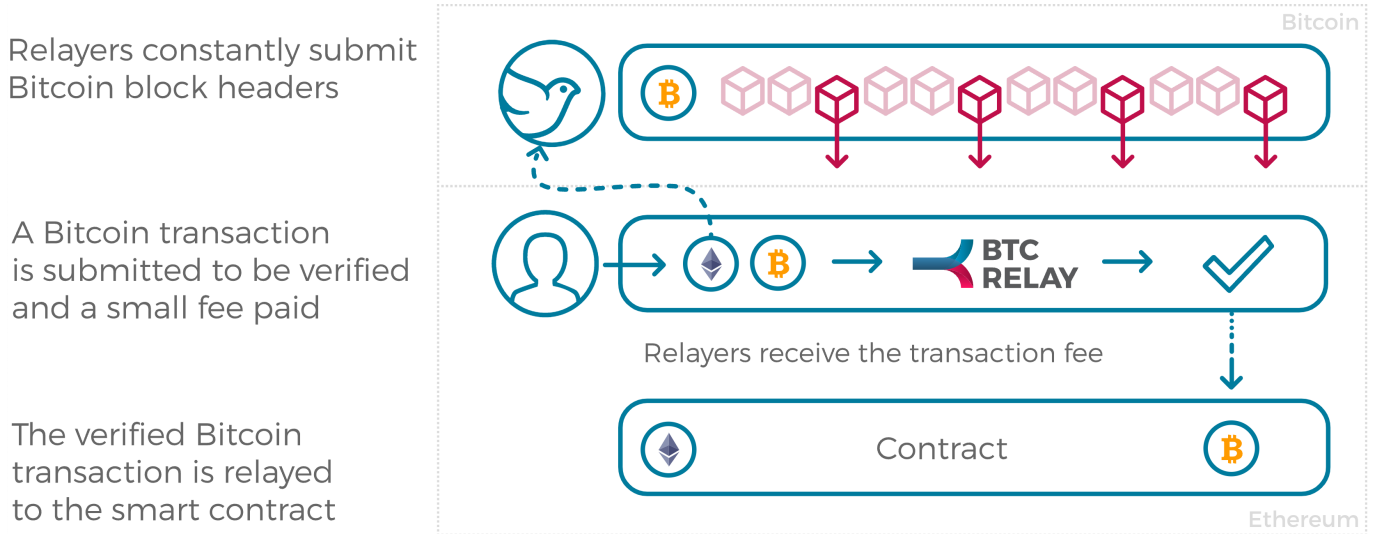
Simplified Payment Verification

- 待验证的交易和交易的Merkel路径证明
- 最长链block header



BTC Relay

- 基于SPV实现最早的轻节点合约
- 最早的原生验证跨链桥，实现从BTC到Ethereum的单向跨链



03

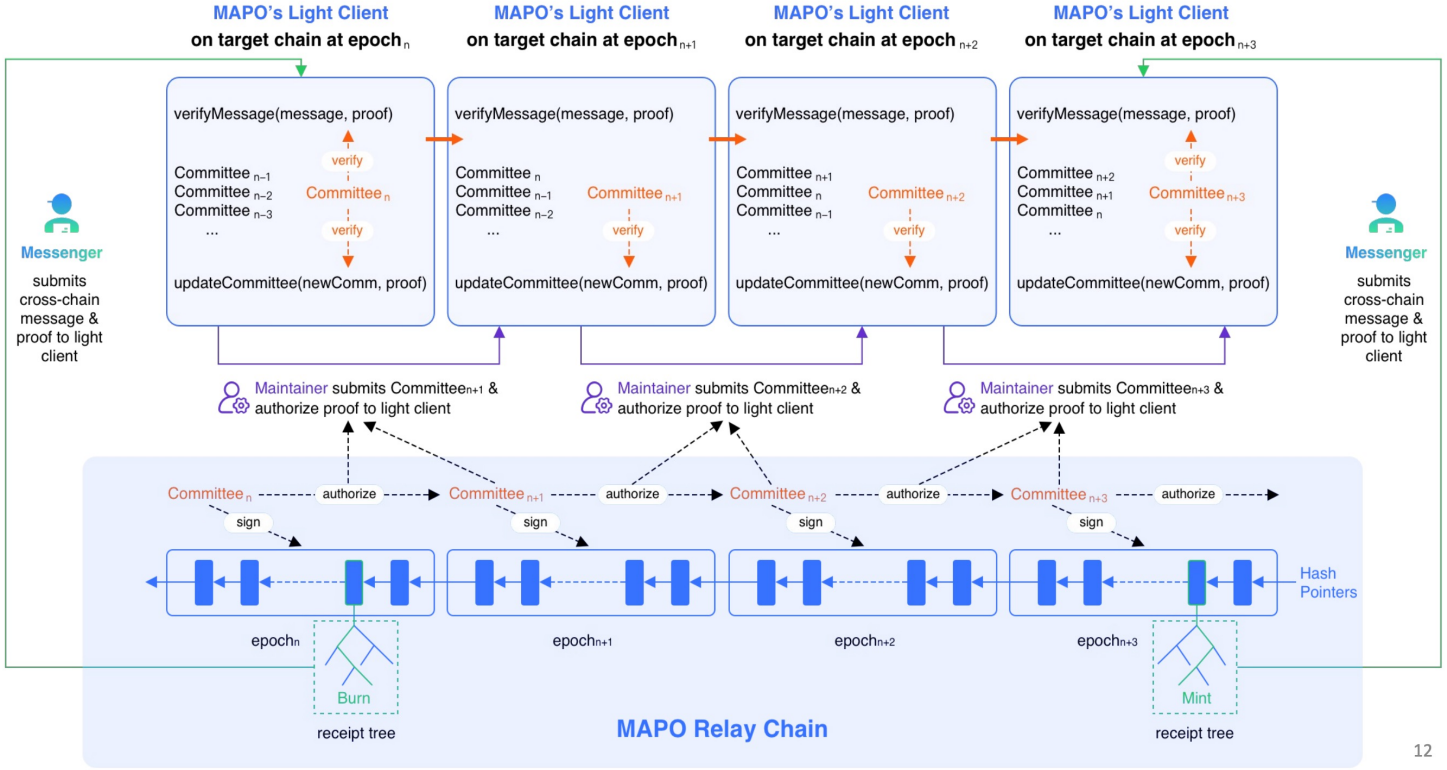
PoS轻客户端实现

共识算法

轻客户端实现

MAPO共识算法

- PoS算法
  - 基于staking数量选举一组validator
  - 在一个epoch内，validator保持不变
- 出块算法
  - validator基于BFT出块
  - 超过2/3 validator签名，并将签名聚合放入block header中



MAPO轻节点实现

- 获取创世或者某一个epoch的validator set的public key
- 获取新的区块头
  - 校验区块头基本信息
  - 根据区块头当中签名信息，验证是否由validator set 2/3以上签名
- epoch结束时，将区块头中包含的新的validator set信息保存
- 获取信息区块头验证

04 轻客户端状态验证

Merkle Tree和Merkle Paricia Trie

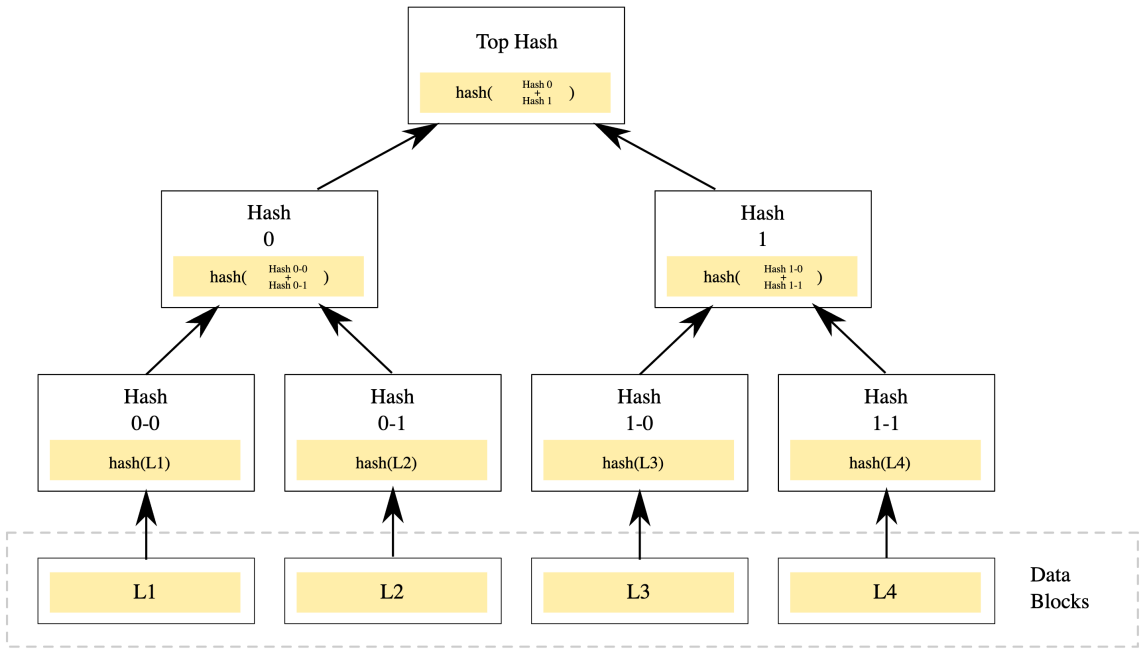
状态用户验证

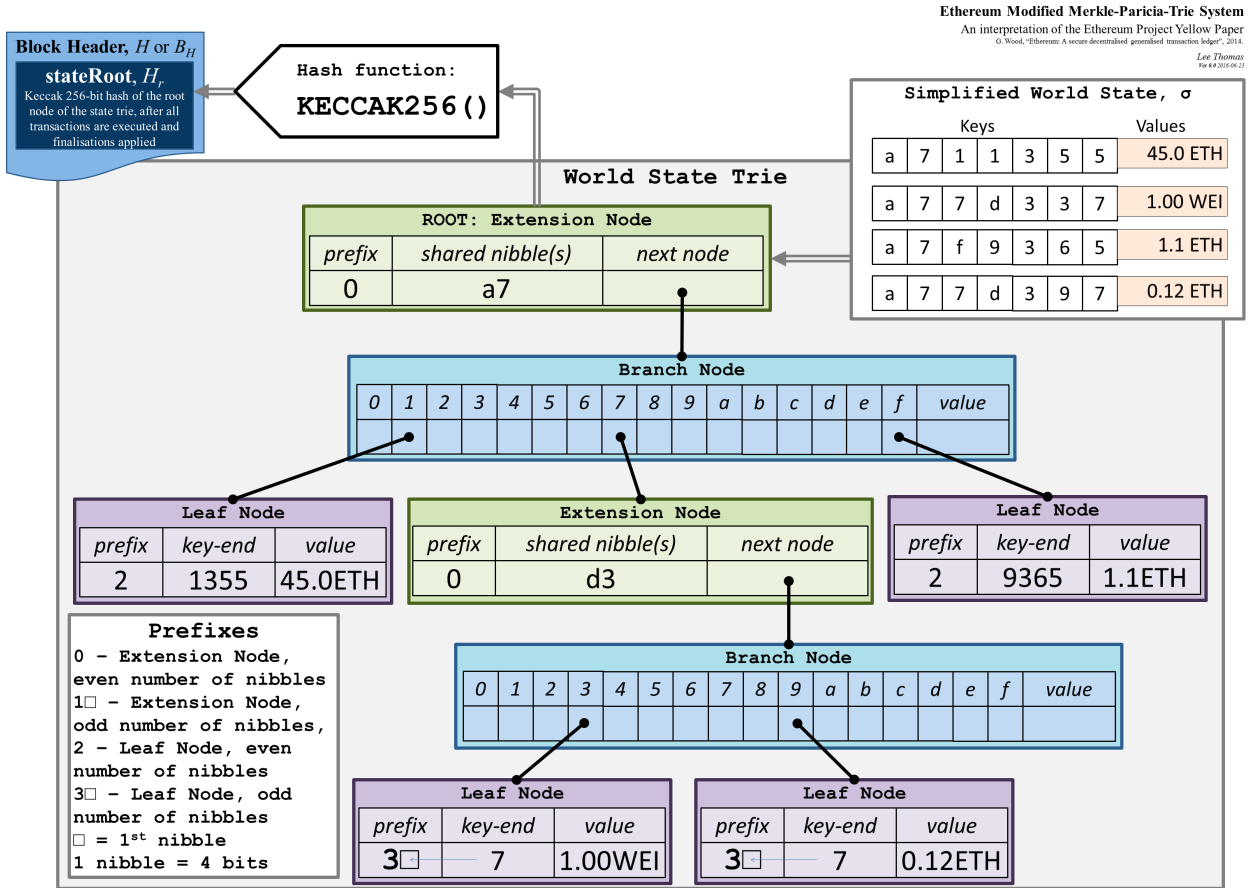
交易日志验证

EVM链轻客户端验证

- Block Proof - verifies the content of the BlockHeader
- Transaction Proof - verifies the input data of a transaction
- Receipt Proof - verifies the outcome of a transaction
- Log Proof - verifies the response of eth_getPastLogs
- Account Proof - verifies the state of an account
- Call Proof - verifies the result of a eth_call - response

Merkle Tree和Merkle Paricia Trie

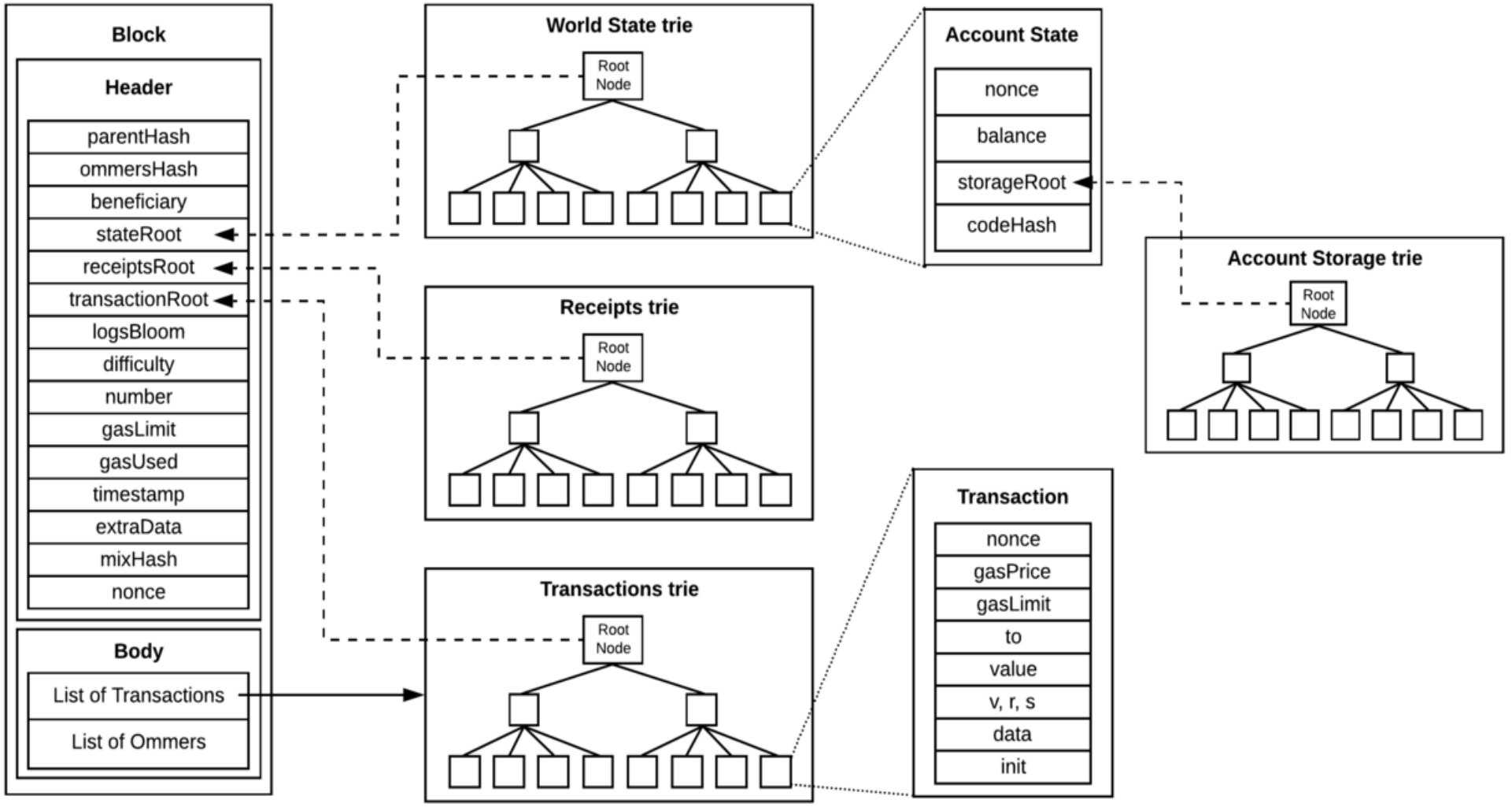




默克尔树，自底向上构建，相邻两个节点的哈希值合并成一个字符串，然后计算这个字符串的哈希，得到的就是这两个节点的父节点的哈希值

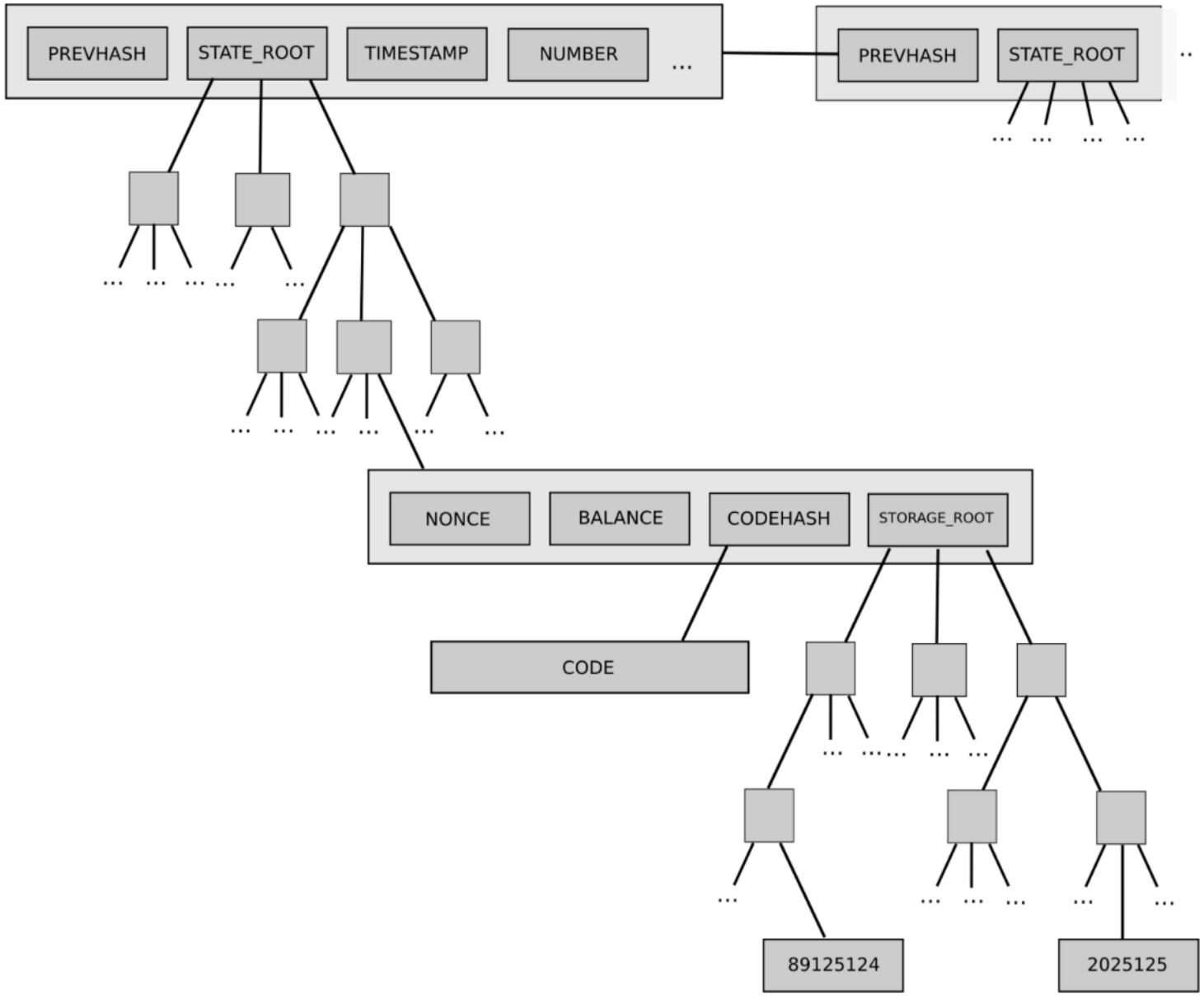
MPT树，经过改良的，融合了融合了默克尔树和前缀树两种树结构优点的数据结构，包括分支节点（branch node）、扩展节点（extension node）和叶子节点（leaf node）。

EVM链区块头结构

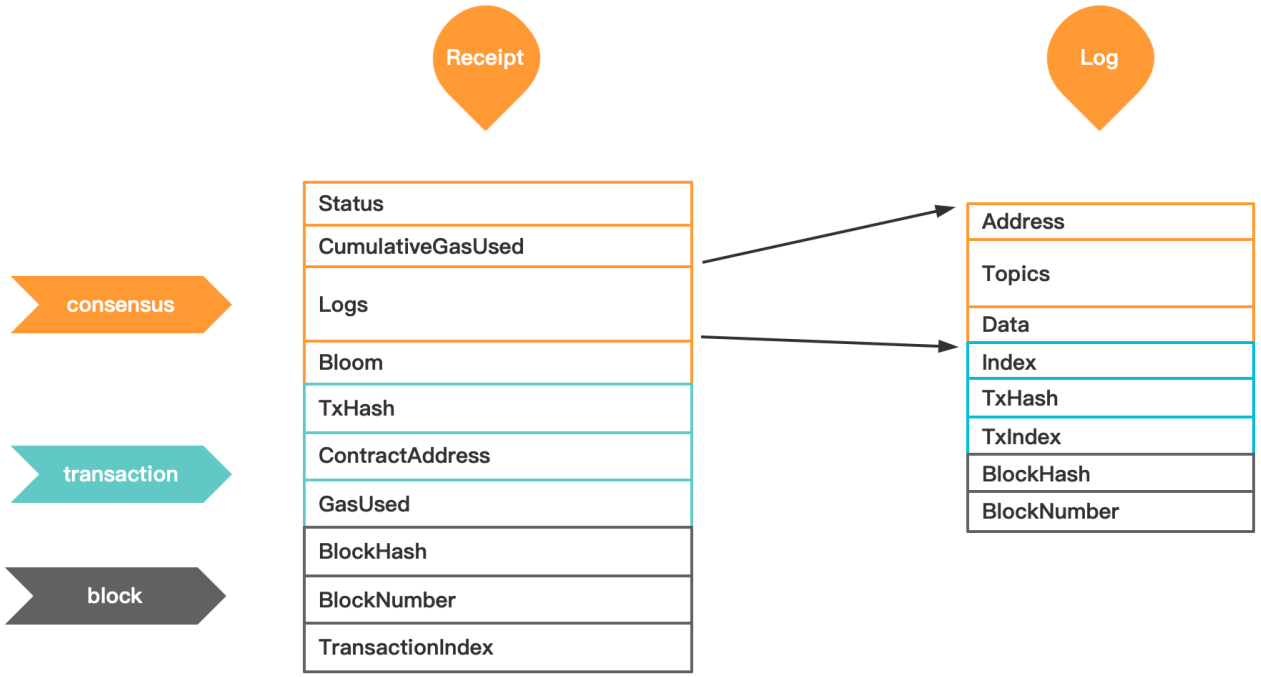

Block
Header
parentHash
ommersHash
beneficiary
stateRoot
receiptsRoot
transactionRoot
logsBloom
difficulty
number
gasLimit
gasUsed
timestamp
extraData
mixHash
nonce
Body
List of Transactions
List of Ommers
World State trie
Root Node
Receipts trie
Root Node
Transactions trie
Root Node
Account State
nonce
balance
storageRoot
codeHash
Account Storage trie
Root Node
Transaction
nonce
gasPrice
gasLimit
to
value
v, r, s
data
init

状态验证

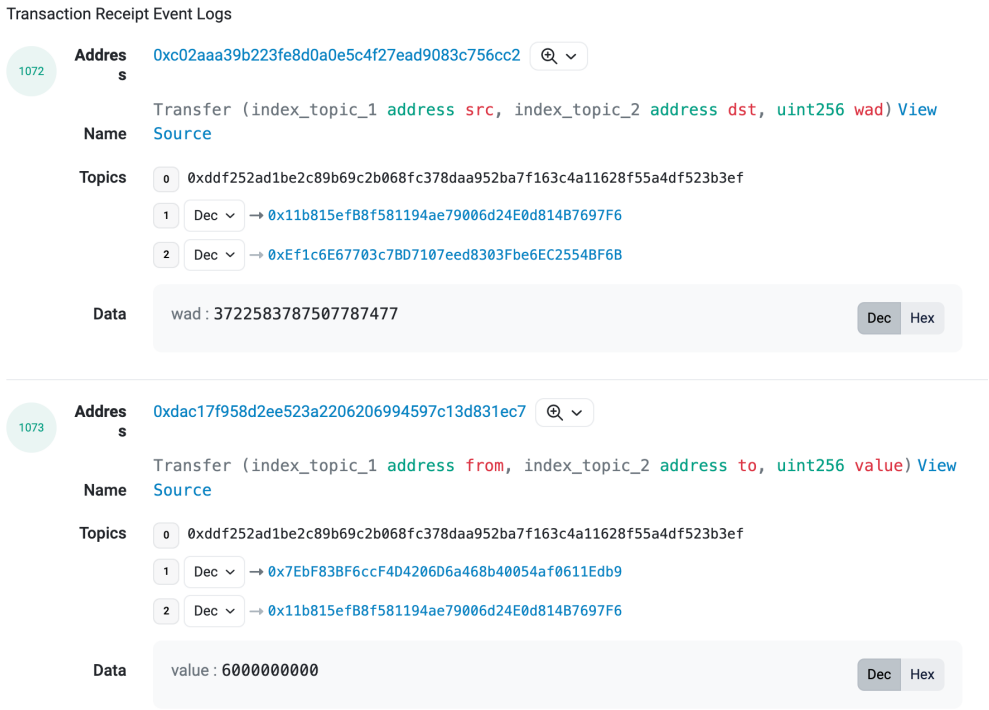
- 用于验证用户状态或者合约状态
- 用户状态
  - 基于state root，构建账户的MPT Proof
- 合约状态
  - 基于state root，构建合约账户的MPT Proof
  - 基于合约账户的storage root，构建合约值的MPT Proof



交易日志验证



- 每个Event对应一个Receipt的Log
- 每个Receipt是一个Transaction的执行收据
- 区块中所有交易的receipt构建为一个MPT



- Event是用于判断合约是否正常执行的依据

交易日志验证流程

- 区块头验证，基于轻客户验证验证区块头是否正确
- 获取区块头中的Receipt Root
- 针对需要验证的交易的Receipt构建MPT Proof
- 基于区块头中的Receipt Root验证交易的Receipt
- 解析Receipt，获取交易的Events，做后续处理

05

基于轻客户端的跨链

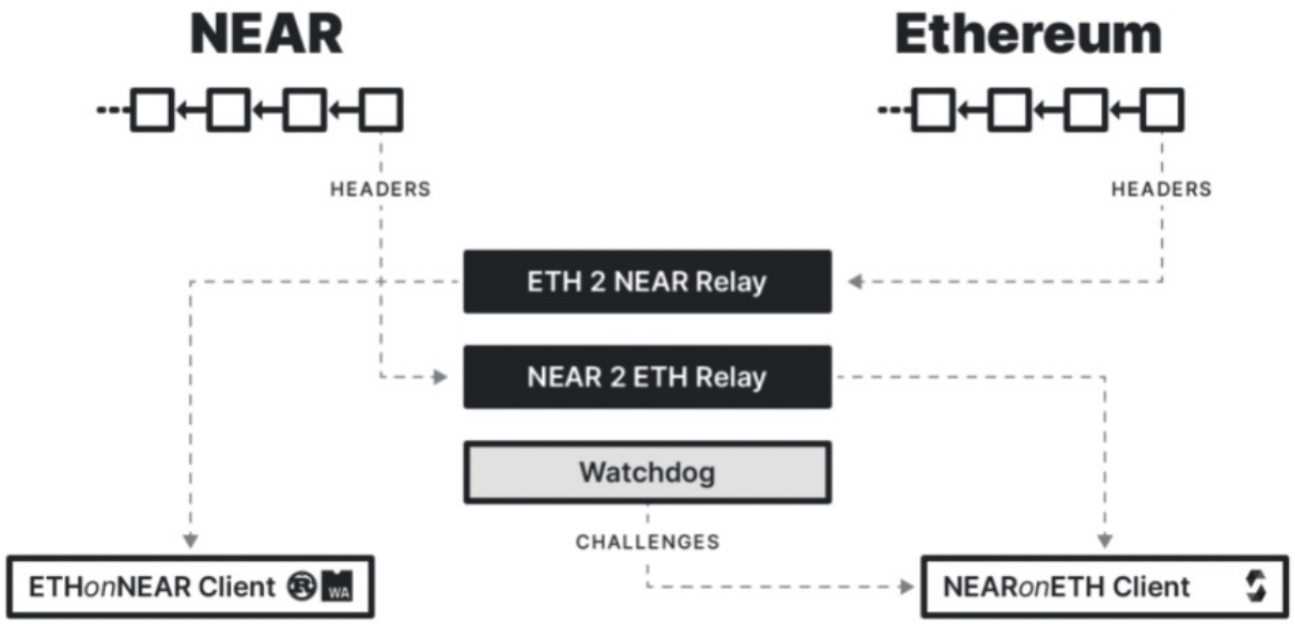
如何基于轻客户端实现跨链验证

同构/异构跨链架构

双链跨链

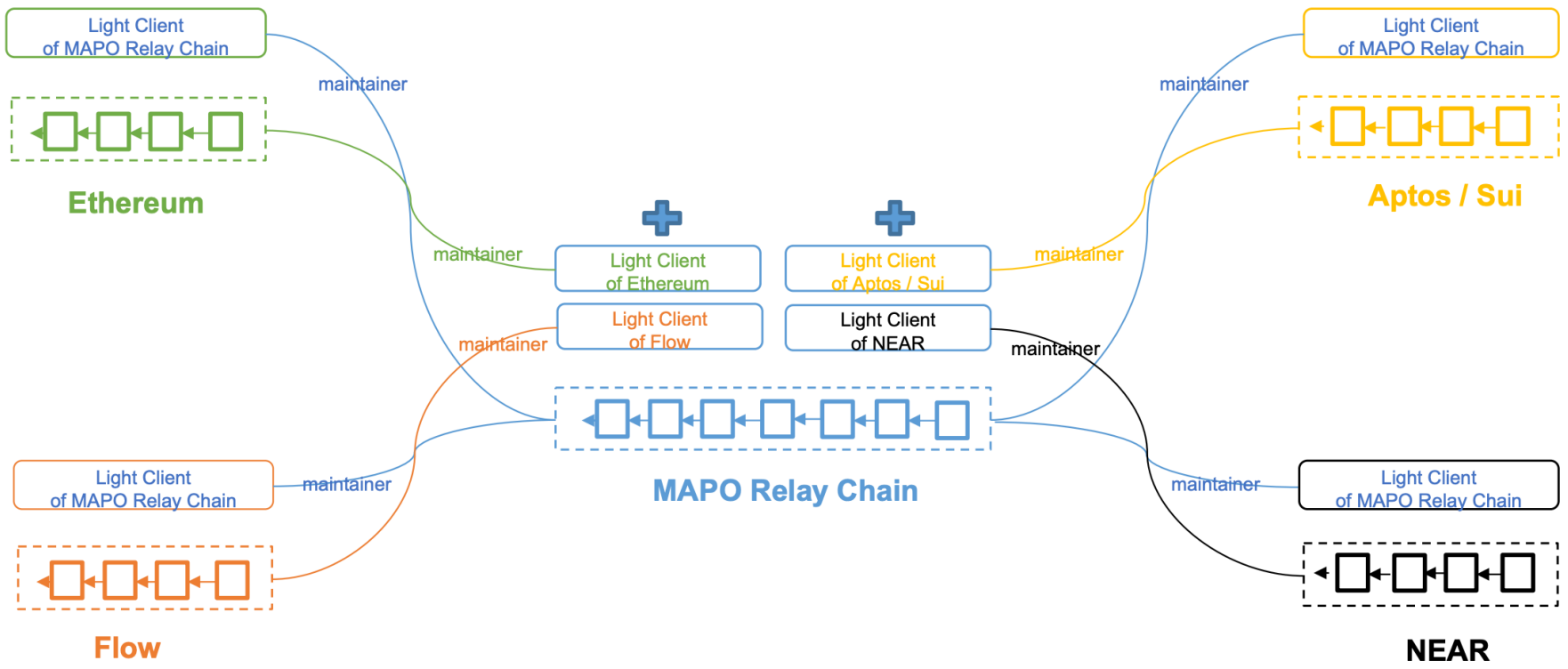
- 分别在两个链上部署对端轻客户端合约
- 由链下服务Relayer将最新区块同步到相应轻客户端
- Relayer监控跨链交易，构建跨链交易Proof提交到对端链上验证，完成跨链

- 多链的复杂性
- 链上合约指令支持

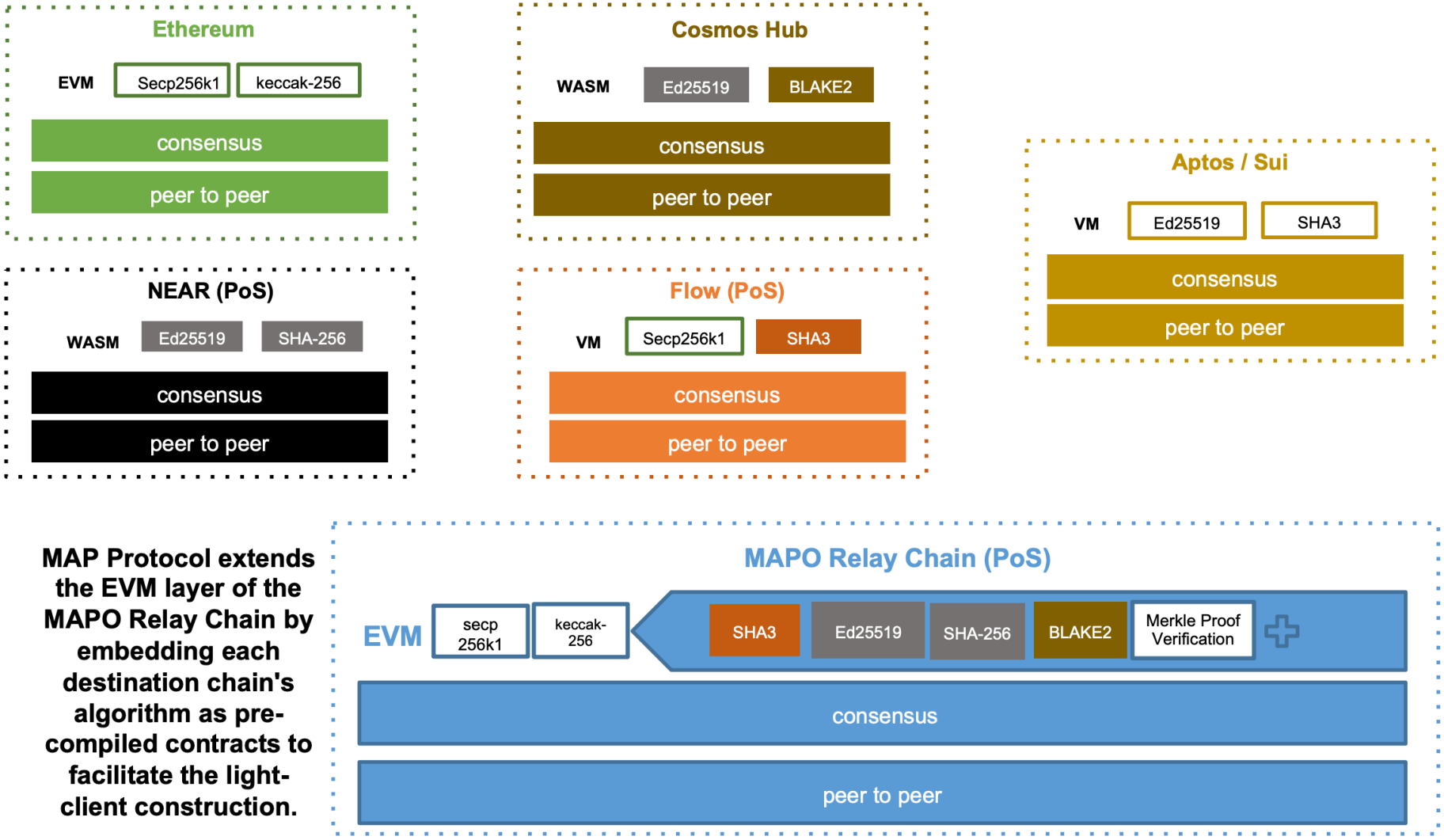


多链跨链

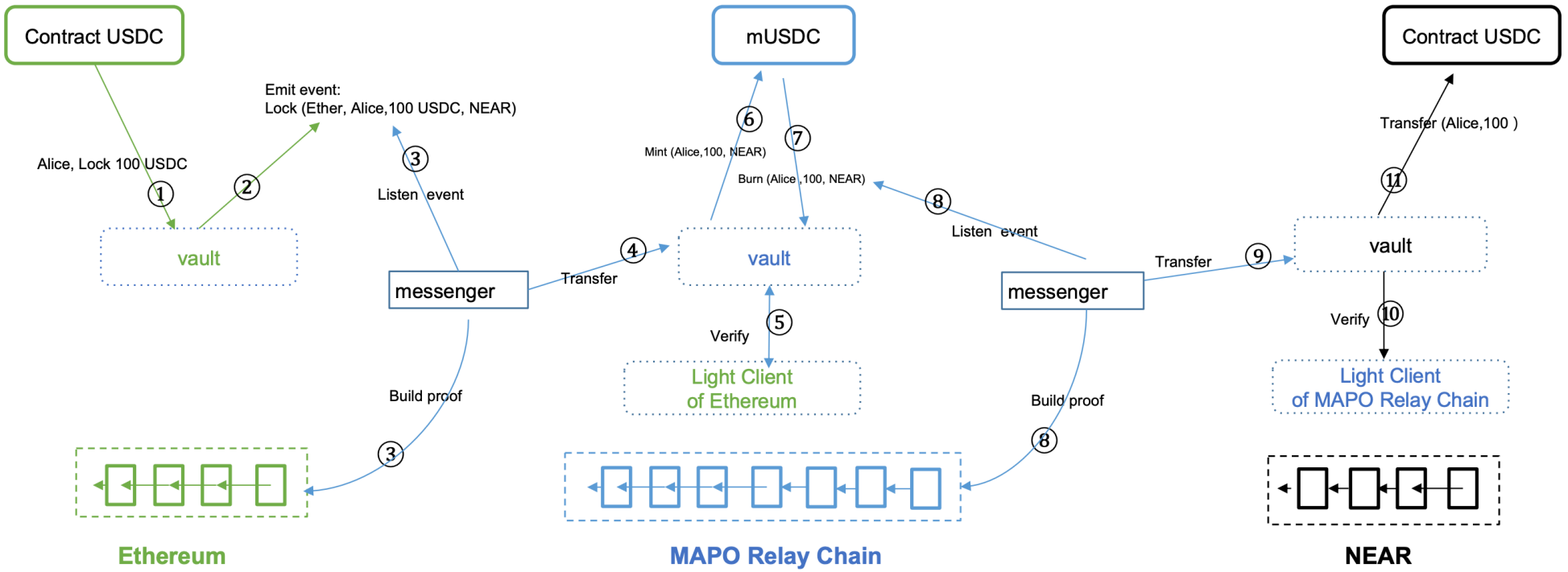
- 在N条链上分别部署对端轻客户端跨链，需要N*N个轻客户端
- 建立Relay Chain，由Relay Chain提供中继



轻客户端预编译合约支持



跨链流程


Contract USDC
Alice, Lock 100 USDC
1
Emit event:
Lock (Ether, Alice,100 USDC, NEAR)
2
vault
3
Listen event
messenger
Build proof
3
Ethereum
4
Transfer
mUSDC
6
Mint (Alice,100, NEAR)
7
Burn (Alice ,100, NEAR)
vault
5
Verify
Light Client
of Ethereum
MAPO Relay Chain
8
Listen event
messenger
Build proof
8
Transfer
9
Contract USDC
Transfer (Alice,100)
11
vault
Verify
10
Light Client
of MAPO Relay Chain
NEAR

06

轻客户端优化

轻客户端问题

基于ZKP的优化

基于轻客户端跨链方案的问题

- 目前链上轻客户端合约开发复杂，同时需要考虑预编译合约的支持
- 链上轻客户端合约需要存储大量的validator和区块头信息，消耗gas偏高
- 签名验证和MPT验证消耗gas偏高

基于ZKP的轻客户端优化

- 签名验证和MPT验证通过zkSNARK
- 链上轻节点合约不需要存储大量的validator和区块头信息，只需要保存最新的validator set或者最新区块头的commitment信息

THANKS