NetworkStatusBlockchain

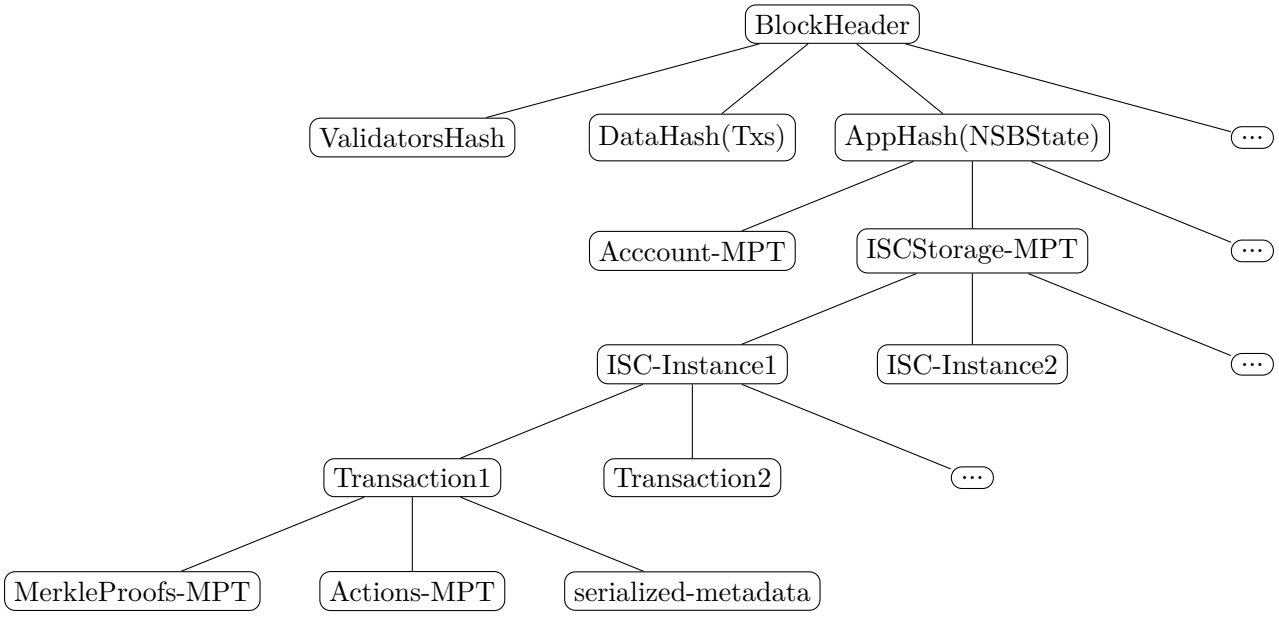
xyangxi5@gmail.com

1 Storage

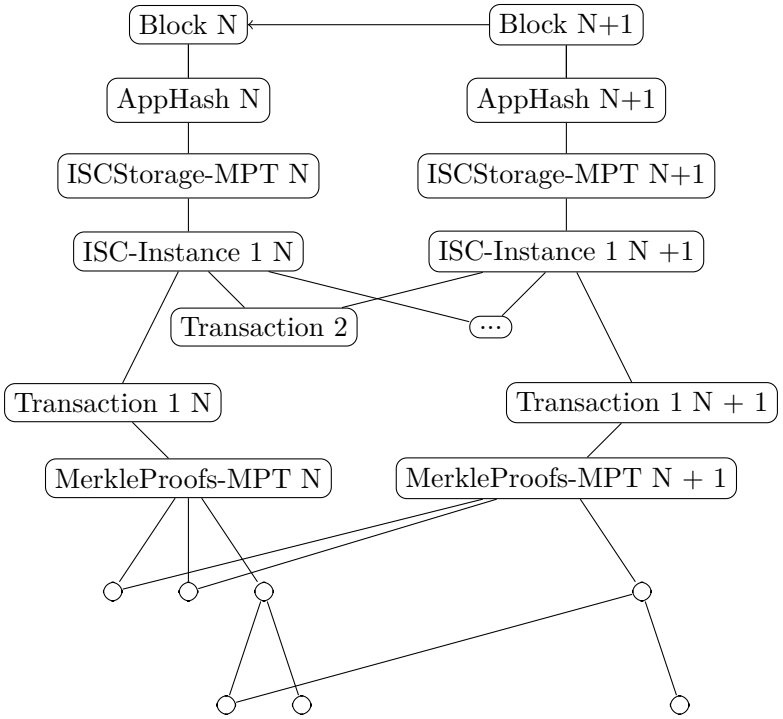
ValidatorsHash 是当前 validators set 的 root hash. 只有 validators set 中的节点才能对 Merkle Proof Pool 中的 merkle proof 进行投票.

DataHash 是 Block 中所有 Transaction 组成的集合的 MerkleTree 的 root hash.

AppHash 是左子树 Actions-MPT 和右子树 ISCStorage-MPT 组合成的 merkle tree 的 root hash.



NSB 不会删除任意一个已加入数据库的树结点. 这使得 MPT 具有持久化的特性. 这和 Ethereum 中 MPT 的特性是一样的.



2 Data Structure

2.1 MPT

MPT 定义如下, 和 Ethereum 中的 MPT 一样:

MPT := $\langle \text{node} \rangle$

node := $\langle \text{fullnode} | \text{shortnode} | \text{valuenode} \rangle$

fullnode := list[hash($\langle \text{node} \rangle_0$, hash($\langle \text{node} \rangle$)$_1$, ..., hash($\langle \text{node} \rangle$)$_{15}$, $\langle \text{value} \rangle$]

shortnode := list[$\langle \text{key\_slice} \rangle$, hash($\langle \text{node} \rangle$), 0]

valuenode := list[$\langle \text{key\_slice} \rangle$, $\langle \text{value} \rangle$, 1]

value, key_slice := $\langle \text{byte} \rangle^*$

byte := $\langle 0|1 \rangle^8$

2.2 MerkleProof

MerkleProof 的定义为 MerkleProof := tuple(type, meta),meta 总是 `[][]byte`, 即字节切片的切片.

如果是 Ethereum, 有 type = eth, meta = list[chainid, storagehash, key, value]

MerkleProof 中保存的 meta 可以验证 MerkleTree 上某一个结点值为 value.

2.3 Action

Action 的定义为 Action := tuple(type, meta),meta 总是 `[][]byte`, 即字节切片的切片.

如果是 Ethereum, 有 type = eth, meta = list[signature, msghash]

2.4 NSBState

NSBState 中包含能够描述某一时间 NSB 所有状态的信息, 现在只包含一个 Accounts-MPT 和 ISCStorage-MPT. Accounts-MPT 可以索引 blockchain 上某一个 account 的所有信息,ISCStorage-MPT 可以索引 blockchain 上某一个 isc 的所有信息.

2.5 Account

未定, 因为 Tendermint 的奖惩机制也是自定的.

2.6 ISCState

见以前的文档.

2.7 ISC

ISC 是一次跨链交易的合约实例.ISC 保存 TransactionIntents, 自身的状态和每一个 Transaction 的执行情况.

ISC 的 TransactionIntents 是有序的, 序号较小的 TransactionIntents 对应的 Transaction 一定被更早地完成.

2.8 TransactionState

见以前的文档

2.9 Transaction

Transaction := tuple (seq, state, intent, MerkleProof-MPT, Action-MPT).

seq 指示这个 Transaction 在当前交易中是第几个被进行的,state 指示这个 Transaction 已经进行到哪一个状态了,intent 指示原始的 TransactionIntent.MerkleProof-MPT, Action-MPT 之所以为 MPT, 是为了更好的进行存在性证明.

3 Storage

使用 leveldb, 只有键值存储逻辑.

所有的 mpt-node 都是 hash(prefix+value) → value.

nsb-state 是 hash(prefix) → value.

isc 任意一个元素 x 存储是 hash(isc addr + prefix + cal(x)) → x.

isc.transaction[i] 的 cal(transaction[i]) = i,isc 的其他状态只靠 prefix 防止冲突.

Account-MPT 的叶保存 account 信息,MerkleProof-MPT 的叶保存 merkle proof 的信息,ISCStorage-MPT 的叶保存 ISCStorage 的根 hash,Action-MPT 的叶保存 action 信息.

4 RPC

1 RPC method: 申请启动一个 ISC 实例

input: transaction intent, isc owners

output: isc address

isc owners 第一个总是 ves, 后面是此次参与交易的用户.

2 RPC method: 申请添加一个 Merkle Proof

input: isc address, tid, merkle proof.

output: response code

请求证明 isc address, 第 tid 个 transaction 的输出中包含 merkle proof. 返回一个 response code 值表示添加的状态

3 RPC method: 申请添加一个 action

input: isc address, tid, aid, action

output: response code

请求为 isc address, 第 tid 个 transaction 第 aid 个状态添加 attestation. 返回一个 response code 表示添加的状态.

4 function: insurance claim

input: isc address, tid, aid

output: response code

请求为 isc address, 第 tid 个 transaction 第 aid 个状态转移状态. 返回一个 response code 表示添加的状态.

5 RPC method: settle contract

input: isc address

output: response code

请求为 isc address 代表的交易清算结果.

6 RPC method: return/stake funds

input: isc address

output: fund

return 只能在 settle contract 以后进行,stake 只能在 settle contract 之前进行.

7 other RPC methods, 用于查询区块链状态.

5 todo

1 application

a nsb action 下周

b nsb merkle proof 下周

c isc upload 下周

d isc insurance claim 下下周

e isc settle contract 下下周

2 MPT 下下下周

3 HyperService wrapping 下下周.