HyperService 进展(190407-190414)

1.EOS 研究: 有一套复杂的区块发射和回溯机制. 分为内存数据库 ForkDatabase 和外存数据库 Chainbase. ForkDatabase 保存可逆区块, 这里区块只有轻量级的数据, 值得关注的是 transaction_merkletree_root 和 action_merkletree_root. Chainbase 中有合约信息部分, 最终都会映射到本地文件, 合约信息也是持久化的, 看起来这个持久化是可用的. 但是机制其实不完善.

1.1. 合约信息如何与区块联系？

1.2. 只有轻量级的数据是我下的判断.

1.3. 时间比较紧迫, 如果仔细研究太花时间了, 不值得.

2.Ethereum 部分:

2.1. Attestation 部分已经和 VES, DApp, NSB 接洽了.

2.2. 假设 a1 在链 A, a2 在链 B, VES 在链 C, NSB 在链 D.会有跨链处理的问题. 现在的处理方法是：

a1, a2, ves 应该都需要在链 D 上注册一个用来和 NSB 互通的账户.

假设 NSB 在链 B 和链 C 上都有合约部署，a1,a2,VES 应该在 intent 发送的时候互相约定一个 NSB 入口(NSB 当然不止一个合约，按理来说，NSB 之间的状态也可以需要同步)。

2.3. Action 和 Merkle Proof 污染问题.

在概率上，hashed data 是不用太考虑碰撞的. 可可以看见的是，有的时候相同内容的 Action 和 Merkle Proof 不止一个，甚至同一组 Transaction Intents 中也不止一个，这时候 NSB 就会发生误判。

处理方法是：在 Action 和 Merkle Proof 上增加一个 nonce 域.

或者将 Action 和 Merkle Proof 隐藏在 Transaction information 中，即: Transaction[isc_addr].txinfo[transaction_index].actionTree/merkleProofTree

两者开销差不多.

e.g. Action = (nonce, msg hash, signature)

nonce 有 16bytes, 8bytes(2^64)是 session id，表示当时约定的 session id, 8bytes 单纯是随机数据.

2.4. workflow test 过程：

DApp 向 HSL 提交 Op Intent 请求〇

HSL 返回 Op Intent √

Op Intent→Transaction Intent √

Transaction Intent→Transaction〇

DApp, VES send/receive/check Action √

DApp, VES send/receive/check Merkle Proof √/〇/Windows Only

2.5. Merkle Proof 何时加入 workflow test

Linux 那边还没有写脚本，所以第二条链(现在在 Linux)上的 Merkle Proof 拿不到.

2.6. TransactionIntent 转 Transaction 有什么问题？

以前是没问题的，现在需要增加一个 Data Parsing.

Data Parsing 目前来说, 应该交给 VES.