HyperService 进展

(190220-190227)

1. 找到一个Web3.py的包, 允许几乎所有的 JSON-RPC 方法, 比较实用.

   a. 用 web3 =Web3(httprovider(url)) 可以创建一个实例.

   b. 例如 js-console 中的想要查 eth.coinbase, 那么在 python 中使用 web3.eth.coinbase.

   c. 可以使用其中的库函数签名.

   d. 已经用 Web3.py 完成了合约的交互.

   e. 建议使用 Web3.py 重写以前请求的大部分方法, 一些最近添加的方法如 getProof 才考虑使用 request+JSON-RPC 请求.

2. 使用Go-levelDB, 可以连接上 Ethereum 的数据库 ethdb.

3. 自己写了一个 golang 版本的 rlp-decoder, 这样可以在 go 中与 ethdb 完成 VerifyProof.

   a. 比如想要证明 slot 为 0x0 的第 0x0 个位置中的值为"0x33", 那么即是验证:

$$\begin{aligned} \text{storage}[\text{keccak256}(&\text{"00000 00000 00000 00000 00000 00000 00"} \quad + \\ &\text{"00000 00000 00000 00000 00000 00000 00"} \quad)] \mapsto \text{0x33} \end{aligned}$$

4. 关于写 NSB 的思路

   a.ActionTree 应该有一个链上的版本 (在 StorageTrie 上?).

   b.ActionTree 可以有一个链下的版本, 可以采用plyvel(一个 Py-levelDB) 作为数据库.

   c. 验证一个 MerkleProof 或 Attestation 是否在链上:

$$\text{Mapping} : \text{keccak256}(\text{bytes32}(\text{keccak256}(\text{Merk or Atte})) + \text{bytes32}(\mathsf{slot})) \mapsto \text{Merk or Atte}$$

   d. 那个 Dependency Graph 具体怎么做?

   e. 合约的书写参考 broker.sol, 然后在 Hyperservice 中实现方便的方法操作合约.