Request Coin

Open-Source, Decentralized Oracle and Uptime Ledger

Austin Thomas Griffith

August 16, 2017

Abstract

decentralized oracle
uptime ledger
http to ipfs

what it is
how it works

Controllers

Head/Main
Provides a mapping to all the different active contracts.
This lets you upgrade a contract by deploying a new version, copying state, and updating head contract with new address

Auth
An auth contract that will store a mapping of addresses to an ENUM permission
Methods to get and set permission based on role of sender
Ability for trusted addresses to vote out or in new trusties?

Token
An ERC20 or similar token contract that will be used as the currency for requests
Even the token could theoretically be upgraded (freeze contract, transfer balances, set new address for token contract in head)

Requests
An open job queue of Request still waiting for more miners
Requests will be represented by a struct and can have many different forms
(Some requests will want consensus while others will want single calls)
Combiners
Once a request is fulfilled by a miner, it should be processed on the way back in
Meaning, the request struct will define a contract ENUM (looked up on head) to process the results
Processors could range from simple passalong, to hash, to different forms on

consensus

Responses
Stores references and other meta data for responses
This will include the original request id, status code, request time, and content hash
This could also store a consensus or anything else the processor ends up collecting and specifying

Request Response Lifecycle

End user sends Request struct to Requests contract along with some amount of RQC token to incentivize miners
Miners read Requests Events, pick the best job (incentives), and make http requests
Depending on request specifications, the Miner will at least submit the a response to the processor defined in the request
The miner may provide hash of the content, the response code, the response time, an IPFS address of response, location of miner, etc to the processor The processor will accumulate responses until the defined consensus is met and create a response
This response event is watched either on chain or off by either RequestCoin clients or third party apps
Public and decentralized history of all requests/consensus/responses

//best languages are C++ and AWK so far, maybe Scilab,

```
import "Token.sol";
//some comment
contract Requests is Freezable,Descendant {

  address public mainAddress;

  function Requests(address _mainAddress) {
    mainAddress=_mainAddress;
    //setAncestor(_ancestor);
  }

  function setMainAddress(address _mainAddress){
    Main main = Main(mainAddress);
    Auth auth = Auth(main.getContractAddress(0));
    if( auth.isOwner(msg.sender) ){
      mainAddress=_mainAddress;
    }
  }
}
```

Todo

Register requestcoin.eth, request.eth, and rqc.eth