Request Coin

Decentralized Blockchain Oracle

Austin Thomas Griffith

October 21, 2017

Abstract

Blockchain technologies, smart contracts in particular, are incredibly powerful and are poised to disrupt many aspects of finance, business, and law. However, due to the deterministic and decentralized nature of smart contracts, they are unable to communicate directly with valuable data sources on the internet.

An oracle provides that link by making off-chain requests and delivering results on-chain. Unfortunately, only centralized oracles exist today and contracts are forced to rely on a single point of failure which could easily be attacked, manipulated, or corrupted.

A decentralized, trustless network of incentivized oracles is needed to empower the next generation of smart contracts.

With the correct cryptoeconomics in place, a statistically adequate supply of miners will request information from multiple internet sources and, in combination with the staking of a token, reach a blind consensus off-chain. Then, efficiently inscribe the data onto the blockchain, available publicly for smart contract logic and execution.

The Pyth.io network will provide the raw middleware between Ethereum and the greater internet by allowing contracts to signal for data which will trigger miners to retrieve the information, reach a truthful consensus, and deliver it back to the blockchain. The miners will, in turn, receive a token reward originally reserved by the requesting contract.

Example

In the not-too-distant future, farmers all around the world could pay Ether into a smart contract that would provide agricultural insurance against hail or drought. Then, throughout the year, as request miners detect these specific weather conditions using multiple APIs and other internet sources, the contract would deterministically pay Ether back to the farmers in need. This incredibly efficient system completely sidesteps an entire field of insurance agents and adjusters, immediately helping the farmers in need without any unnecessary overhead.

Lets dive into an oversimplified (and insecure) example contract just to understand the mechanics of how such a system would work.

First, well need a way to signal miners that a consensus is needed for a particular data point:

```
mapping (bytes32 => string) public requests;

function addRequest(bytes32 _id, string _url) returns (bool){
    requests[_id]=_url;
    AddRequest(msg.sender,_id,requests[_id]);
}
event AddRequest(address _sender,bytes32 _id, string _url);
```

With the addRequest function we can store a request and trigger an event called AddRequest on the blockchain.

Miners, incentivized by a reserved token, then make requests to a number of internet endpoints, collect relevant data, and send it back to the contract.

```
contract.getPastEvents('AddRequest', {
    fromBlock: params.blockNumber,
    toBlock: 'latest'
}, function(error, events){
  for(let e in events){
    request(events[e].returnValues._url, function (error, response,
     body) {
       contract.methods.addResponse(events[e].returnValues._id,body
    ).send({
         from: params.account,
         gas: params.gas,
         gasPrice:params.gasPrice
       })
    })
  }
})
```

The addResponse method is used to store the final response that other contracts could then use to drive their logic.

```
mapping(bytes32 => string ) public responses;

function addResponse(bytes32 _id,string _result) returns (bool){
    responses[_id]=_result;
    AddResponse(msg.sender,_id,responses[_id]);
}
event AddResponse(address _sender,bytes32 _id,string _result);
```

This simplified example is the heart of a decentralized oracle network. We will build a more robust system around this idea in the following posts.