Blockchain, Ethereum, ClojureScript, Fleet

Erwin Rooijakkers

12-07-2017

Blockchain

What?

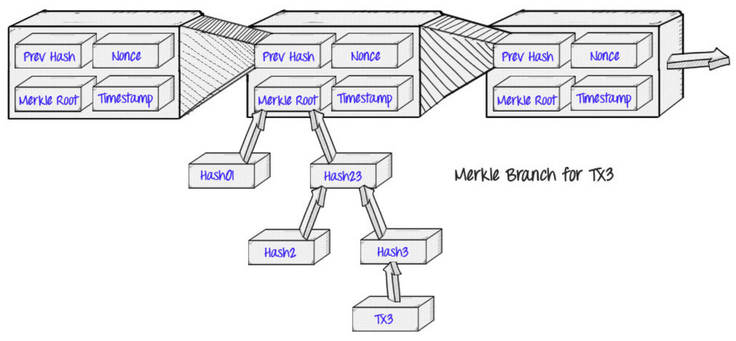
”A shared, programmable, cryptographically secure and therefore trusted ledger which no single user controls and which can be inspected by everyone.”

– Klaus Schwab (Chairman World Economic Forum)

Four pillars

- Cryptographic Tokens and Addresses
- P2P Networking
- Consensus Formation Algorithm
- Virtual Machine

This.



(Source: `https://blog.ethereum.org/2015/11/15/merkling-in-ethereum/`)

Consensus

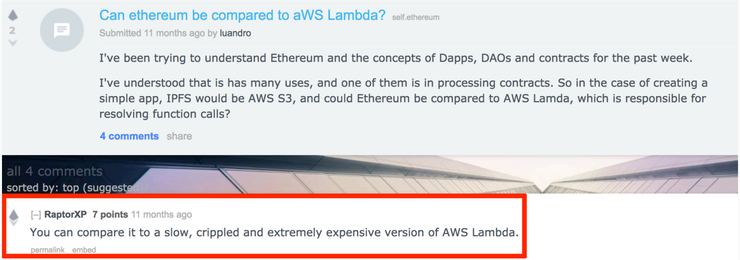
- Proof of Work (PoW): the current difficulty in the Bitcoin network requires miners to try quadrillions of times before finding a nonce that fits. (because hashing can provide vastly different outputs on minor changes)
- Proof of stake (PoS): mining is done by stakeholders in the ecosystem who have the strongest incentives to be good stewards of the system. (E.g., by setting coins aside for a longer period as stake.)

Smart Contracts

Model

- Stateless webservices
- Contract-oriented programming
- Gas fees

Stateless web services



2

Can ethereum be compared to aWS Lambda? self.ethereum

Submitted 11 months ago by luandro

I've been trying to understand Ethereum and the concepts of Dapps, DAOs and contracts for the past week.

I've understood that is has many uses, and one of them is in processing contracts. So in the case of creating a simple app, IPFS would be AWS S3, and could Ethereum be compared to AWS Lamda, which is responsible for resolving function calls?

4 comments share

all 4 comments

sorted by: top (suggested)

[–] RaptorXP 7 points 11 months ago

You can compare it to a slow, crippled and extremely expensive version of AWS Lambda.

permalink embed

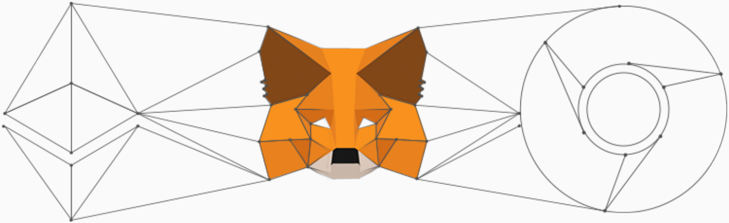
Contract-oriented programming

```
contract HelloSayerFactory {

  function create() returns (address) {
    return address(new HelloSayer());
  }

  function delete(address addr){
    HelloSayer(addr).remove();
  }

}
```

Tools

- geth: (Go Ethereum) cli for running full Ethereum node, exposes RPC
- web3.js: Ethereum compatible JavaScript API which implements the Generic JSON RPC spec
- solc: JavaScript bindings for Solidity compiler (creates ABI and BIN)

MetaMask (also: Mist; Parity)



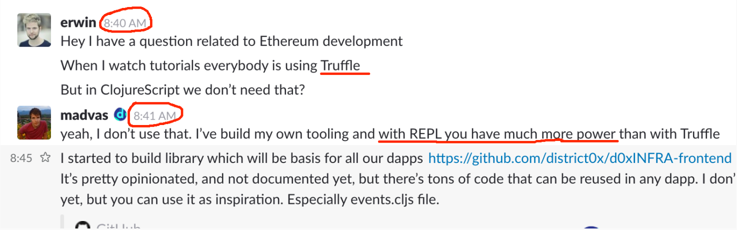
Fleet

Why ClojureScript + blockchain

- Blockchain
- ClojureScript!
    - Reagent
    - Figwheel

Code inspiration and big help

- `https://medium.com/@matus.lestan`
- `https://github.com/district0x/ethlance`
- `https://ethlance.com/#/job/128`



erwin 8:40 AM
Hey I have a question related to Ethereum development
When I watch tutorials everybody is using Truffle
But in ClojureScript we don't need that?

madvas 8:41 AM
yeah, I don't use that. I've build my own tooling and with REPL you have much more power than with Truffle

8:45 I started to build library which will be basis for all our dapps https://github.com/district0x/d0xINFRA-frontend
It's pretty opinionated, and not documented yet, but there's tons of code that can be reused in any dapp. I don'
yet, but you can use it as inspiration. Especially events.cljs file.

GitHub

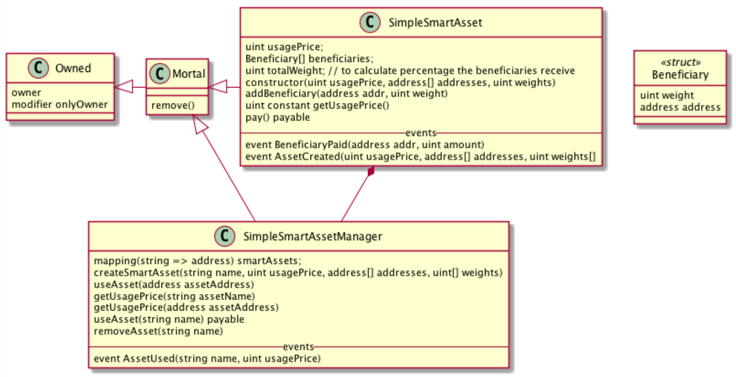
Idea

- Elisa Achterberg (Circle Economy and Sustainable Finance Lab) *et al.*

The idea

- Linear economy *designs for failure* and sells "almost-broken" products, creating waste
- When they are used, smart assets (a fleet of assets) pay parties involved in value chain (involved with design, commodities, creation, maintenance, et cetera)
  - Shift *from ownership to use* leads to Circular Economy
  - A circular value network in which materials and products are shared as well as risks and returns

Design



Owned

owner
modifier onlyOwner

Mortal

remove()

SimpleSmartAsset

uint usagePrice;
Beneficiary[] beneficiaries;
uint totalWeight; // to calculate percentage the beneficiaries receive
constructor(uint usagePrice, address[] addresses, uint weights)
addBeneficiary(address addr, uint weight)
uint constant getUsagePrice()
pay() payable

events

event BeneficiaryPaid(address addr, uint amount)
event AssetCreated(uint usagePrice, address[] addresses, uint weights[]

«struct»
Beneficiary

uint weight
address address

SimpleSmartAssetManager

mapping(string => address) smartAssets;
createSmartAsset(string name, uint usagePrice, address[] addresses, uint[] weights)
useAsset(address assetAddress)
getUsagePrice(string assetName)
getUsagePrice(address assetAddress)
useAsset(string name) payable
removeAsset(string name)

events

event AssetUsed(string name, uint usagePrice)

Fleet demo