Programming with Ethereum

William Harrell
12/4/19
LUG @ NC State

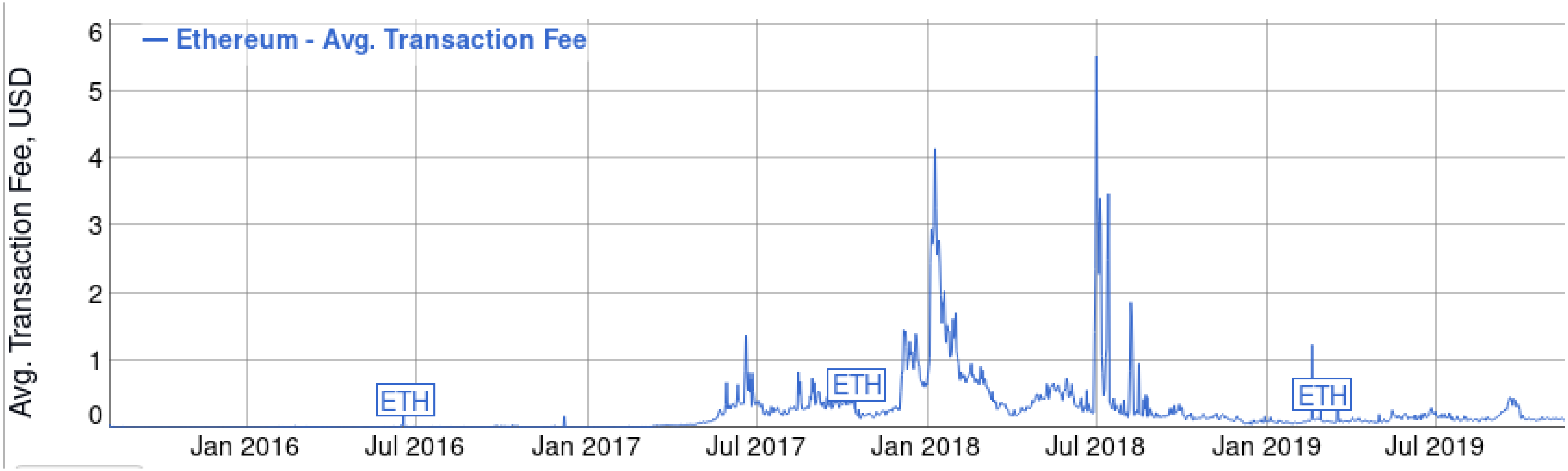
Blockchain

- Linear collection of cryptographic data
- Run by independent nodes in a P2P manner
- Decentralized – No single node controls the blockchain
- Transparency – Anyone can view the entire blockchain
- Immutability – Once something is added to the blockchain, it effectively cannot be changed or removed

Ethereum Virtual Machine

- **Ethereum programming languages compile down to Ethereum Bytecode which runs on the EVM**
- **Quasi-Turing complete language: computations are bound by a transaction fee which is paid to the miners**
  - Current fee is about $0.13 per transaction
- **No non-deterministic functionality (such as random())**

Transaction Costs over Time



Proof of Stake

- **Proof of Work method gobbles electricity – maybe more than Switzerland**
- **Ethereum is moving to Proof of Stake**
  - A stake holder is randomly selected to verify a transaction, with larger stake holders being favored
  - The transaction is forged and added to the network
  - The result can easily be checked. If the holder is caught falsifying the transaction, they lose their stake and can't forge transactions anymore
- **Easier for users with weaker hardware to participate, increasing network strength**

Smart Contracts

- Object-oriented programming adapted to the blockchain
- Centered around the exchange of currency, but can also store data in the blockchain
- Contracts cannot interact directly with the outside environment

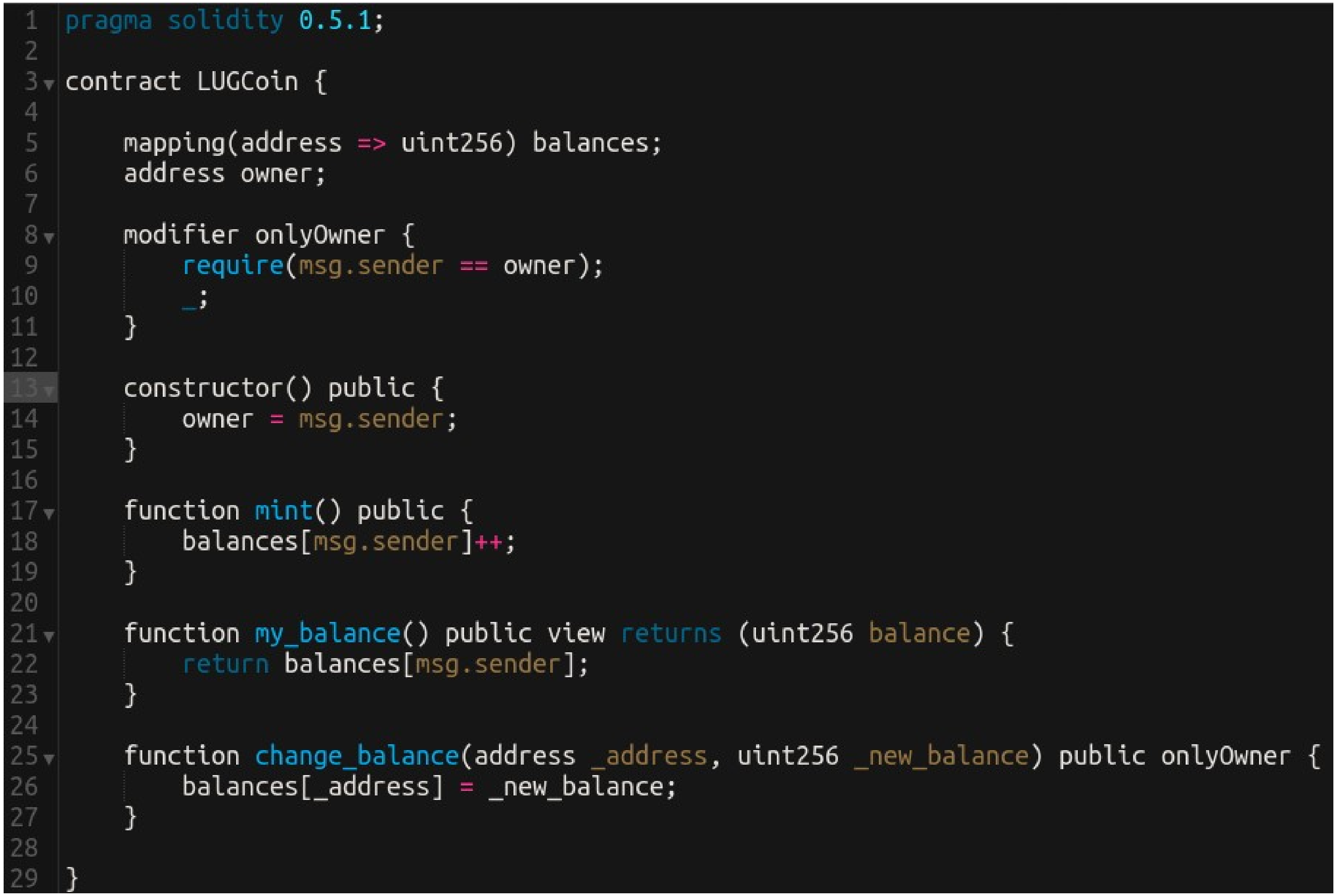
Oracles

- **Method for retrieving data from outside world**
  - A contract communicates with the oracle on the blockchain and requests data from it
  - Oracle retrieves the data from outside world
  - Oracle calls a callback function on the original contract
- **Provable, an oracle service, can retrieve:**
  - HTML/JSON/XML
  - Random numbers
  - WolframAlpha queries
  - Resources on IPFS

Solidity

- Contract oriented language
- Similar style to JavaScript
- A contract is constructed and deployed to the blockchain
- Other users can call public functions

LUG Coin



```
pragma solidity 0.5.1;

contract LUGCoin {

    mapping(address => uint256) balances;
    address owner;

    modifier onlyOwner {
        require(msg.sender == owner);
        _;
    }

    constructor() public {
        owner = msg.sender;
    }

    function mint() public {
        balances[msg.sender]++;
    }

    function my_balance() public view returns (uint256 balance) {
        return balances[msg.sender];
    }

    function change_balance(address _address, uint256 _new_balance) public onlyOwner {
        balances[_address] = _new_balance;
    }

}
```