Blockchain

Paolo Bettelini, Giacinto Di Santis, Gianni Grasso

Contents

1 Block

Each user owns a pair of private and public key.

All the transactions broadcasted to the network are grouped into blocks, which contain

- Markle tree root hash
- Timestamp
- nBits (PoW)
- Nonce (PoW)
- Previous block hash
- Number of transactions

With each block being confirmed, the blockchain is created.

2 Proof of Work

Proof-of-Work (PoW) is a cryptographic proof that a party has spent a certian amount of computational effort.

When a miner solves the puzzle the current block is archived, a new block is generated and all the transactions in the previous block are confirmed. The miner is then rewarded by the system.

3 Proof of Stake

4 Smart Contracts

Smart contracs are programs associated with an address and run on the blockchain. The nodes run code from the contract program at a relevant event, such as a received transation.

Users can interact with the contract via transactions. Contracts can often interact with other contracts and some of them are Turing-complete.

4.1 Deployment

A smart contract is deployed by sending a transaction to the blockchain which includes the compiled program as well as a special receiver address.

The program is added to the current block. When the block is added to the blockchain, the contract will execute one time to set its initial state, at which point the smart contract will now be valid and running.

5 Requirements

Req-01	
Name	BlockChain
Priority	1
Version	1.0
Notes	none
Description	It is required to create a Blockchain based on PoS interacting with each node of the chain.
Subrequirements	
Req-01_0	There must be implemented the Proof of Stake (PoS).
Req-01_1	There must be implemented a peer discovery algorithm.
Req-01_2	There must be implemented a consensus algorithm.
Req-01_3	There must be implemented a P2P algorithm.
Req-01_4	There must be implemented a Gossip algorithm.

Req-02	
Name	Node
Priority	1
Version	1.0
Notes	none
Description	It is required that a computing machine can become a node of the blockchain..
Subrequirements	
Req-01_0	Each node must have the blockchain saved locally as a SQLite database.
Req-01_1	There must be implemented an algorithm that allow al nodes to perform the proof of stake.
Req-01_2	Every node must be capable of executing a transaction.
Req-01_3	Any node must be able to make a smart contract.

Req-03	
Name	Transaction
Priority	1
Version	1.0
Notes	none
Description	On the blockchain it must be able to make transactions, both for cryptocurrency and for smart contacts.
Subrequirements	
Req-01_0	There must be implemented an algorithm that validates the transaction and adds it to the block.
Req-01_0	There must be a deflation system, which functions using gas fees.

Req-04	
Name	Smart Contract
Priority	2
Version	1.0
Notes	none
Description	There must be a system that generates a smart contract that is self-executing, containing the terms of the agreement between buyer and seller written directly in lines of code.
Subrequirements	
Req-01_0	There must be an algorithm dedicated to defalcation system for the transaction of the smart contract with gas fees.
Req-01_1	The code and the agreements in it must exist on the decentralized blockchain.
Req-01_2	The code controls execution and transactions are traceable and irreversible.

Req-05	
Name	WebApp
Priority	2
Version	1.0
Notes	none
Description	It is required that from a web application any user can request and see the basic information of the blockchain (transaction, block, ...).
Subrequirements	
Req-01_0	There must be a section on the site dedicated to information about blocks, transactions, gas fees.
Req-01_1	There must be a section on the site that allows the user to generate a wallet.
Req-01_2	There must be a section on the site that allows the user to carry out and control their transactions.
Req-01_3	There must be a section in the site that allows the user to carry out the proof of stake, that is to block a determined quantity of coins in order to be chosen as validator.

Req-06	
Name	Info Website
Priority	3
Version	1.0
Notes	Required by the mandator, but optional.
Description	It is required a website that explain how the blockchain works and how we've done it.
Subrequirements	
Req-01_0	IDK.

6 Design change

This design change is related to how a user interacts with the blockchain.
Our initial idea was to develop a *client* software alongside the *node* software. The *client* would connect to a single node as an entry to the blockchain to broadcast its transactions. The user would then be able to broadcast transactions to the blockchain using a web-based application.
This is a poor design choice since it has a number of problems.

- **Problem 1** Dual-functionality: Each node would need to be able to handle and distinguish both a connection from a node and a connection from a client.
- **Problem 2** WebAssembly: Both the *client* and *node* software would share the same protocol implementation, written in Java. Since the application is web-based, we would need to compile the *client* code to WebAssembly (WASM) to run on the browser.
- **Problem 2.1** Sockets: It would be challenging to allow for *socket connections* from a Java-based WASM.
- **Problem 3** Peer discovery: The client software can't store the addresses of other nodes since it is a web-based application. Even by caching nodes using *cookies*, clients will generally need to rely on *seeders* more than they should.

The solution is to completly remove the client. Only nodes can broacast transactions throughout the network. The web application will send requests to a server, which is also a node. This server will route the user transactions through its node. No WebAssembly is needed. Connection to the seeder will be established (possibly) only the first time the web application server's node connects to the blockchain. Users can still host their own node and use them as an entry to the network, and multiple users can host their own web application for the same purpose.