Blockchain - Diaries

Paolo Bettelini, Giacinto Di Santis, Gianni Grasso

Contents

1 Diaries

2022-01-27

- Created GitHub repository
- Added Readme
- Created Gantt
- Initial analysis
- Documentation
- Gradle init

Today we started to take a look at the blockchain technology. We started planning which pieces of software we will need to write in order to make everything work. We also initialized the porject using Gradle.

2022-02-03

- Created node module
- Created seeder module
- Imported SQLite dependency
- Tested SQLite database
- Added LaTeX files to `.gitignore`
- Seeder packets
- Byte utils classes

Today we created the node and seeder module using gradle. We focused on defining the protocol for the seeder application (to allow for peer discovery). We wrote che code for the peer discovery packets. We also tested SQLite, creating and interacting with a test database.

2022-02-05

- Design change
- Written seeder service
- Tested connection to seeder

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.

2022-02-10

- Designed peer discovery protocol
- Implemented peer discovery protocol
- Designed logo
- Designed GUI
- Blockchain analysis
- Fat Jar using Gradle

Today we have design the peer discovery protocol and implemented it (not tested). We continued to study blockchain algorithms and also design the logo and the GUI for out web application.

2022-02-17

- Initial Web GUI design
- Initial PoW computation implementation
- Worked on peer discovery and communication protocol
- Database structure discussion
- Initial smart contract language commit

Today we mainly focused on implementing the peer discovery and communication protocol.

2022-02-24

- Finished communication and peer discovery protocol
- Started defining and implementing the high-level blockchain protocol
- Continued to learn about cryptography and blockchain technology

Today we finished the base protocol and started implementing the high-level protocol. The outline for the next session is to solve the problem with elliptic curve cryptography (ECDSA).

2022-03-10

- ECDSA keypair generation
- REST API
- Blockchain consensus algorithm draft

- Some protocol packets
- Website design and css.

The ECDSA signature still doesn't work.