LINFO2345 - Project

1 Introduction

This project will make you discover the mechanisms behind blockchain and especially the blockchain structure and the consensus algorithms behind Proof of Stake blockchains.

This project is composed of two parts:

- Implementation of a distributed consensus algorithm and message broadcast system,
- Implementation of a blockchain structure and an authentication mechanism.

2 Context

A blockchain is a decentralized list of record which use cryptography to guarantee the integrity of all transactions store in it. A blockchain is composed of several lists of record named blocks. Each block is constructed using the previous blocks to guarantee all the transactions stored in it. To construct a block, each peer of the system attempt to resolve a cryptography puzzle to create the new block by brute force. The first node which solves the puzzle will push the new block in the blockchain, communicate it to other peers and be rewarded for this in cryptocurrency. This method to construct a blockchain is the Proof of Work (PoW).

The PoW has some disadvantages for the creation blocks:

- If an attacker wants to modify a record in a block, he needs to recalculate all the successor blocks. But during this time, the honest nodes continue to create blocks. So, an attacker needs a too big amount of resources to recreate all blocks faster than the network.
- The creation of block by brute force uses a huge amount of resources (CPU/GPU power, electricity, ...). Today, Bitcoin [1] uses an amount of electricity equal to the consumption of Hungary.

To solve the problem of PoW, we create the Proof of Stake (PoS) which use an elected group to decide which block add instead of using a challenge. The creation of the consensus group is pondered by the proportion of the blockchain cryptocurrency a node owned (not implemented in the project). The more of it a node owned, the more chances he have to be in the consensus group. Because a node that has a lot of currency doesn't want the blockchain to collapse so he will take decisions in the interest of the blockchain.

3 Project

The goal of this project is to implement a network that shares data via a proof of stack blockchain.

The project is cut into two parts:

- Implementation of a basic PoS Blockchain
- Improvement of the solution

3.1 Implement of a basic PoS Blockchain

The goal of this part is to implement a PoS blockchain without all the security mechanisms used to verify the identity of a node and verify the validity of a block.

This part is divided into 4 sub-parts:

3.1.1 Creation of the network

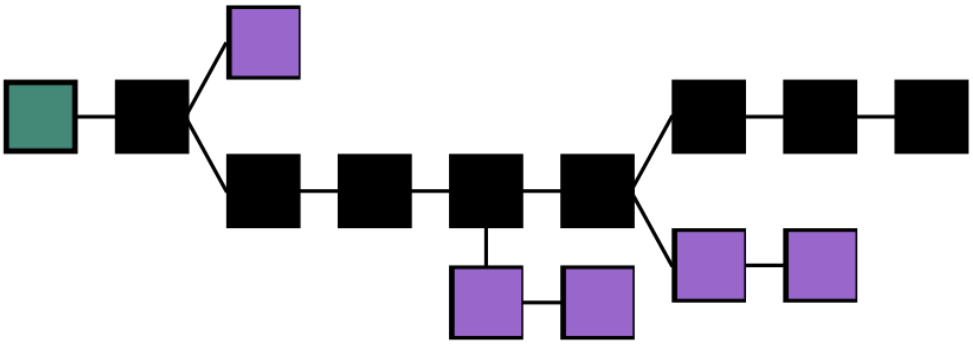
To simulate a Blockchain network, we will implement a network of Erlang Process. You will implement a master process that takes the number of nodes of the network (noted N) in argument. This process will start and stop all the nodes used for the experience. The master process will proceed in two steps to create the network:

- Creation of all nodes of the system and store in a list the PID of each process
- Send the list of PID to all nodes to let them know the network.

The experience will begin after these two steps of initialization. When you want to stop the experience, the master process needs to kill all nodes of the systems. The master process doesn't participate in the experiment

3.1.2 Implementation of the blockchain structure

A blockchain is a linked list of tables containing transactions. In a real blockchain, we use Merkle tree to store hash of transactions in each block. A block also contains the hash of the previous block and the block owner. In the first part of this project, we will just implement a linked list where each block contains the ID of the last block , the PID of the node which push the block and a table with all transactions stored in it. For the project, each block contains a maximum of 10 transactions.



3.1.3 Implementation of the election consensus used to push a block

At each turn of the blockchain, we need to select a user who will push the next block of the chain.

You will use a simplified version of RANDAO [2], the election algorithm which will be used by Ethereum 2.0 [3].

This Algorithm proceeds in two steps:

- Each node will provide to the node which pushed the last block, a random number.
- When the last block pusher receives all the random numbers, it sums all the random numbers and modulates the result by the number of nodes in the network to choose the next node which will push a block.

  You also need to implement a round mechanism to synchronize the consensus election algorithm.

3.1.4 Block push

When a node is elected to push a block. It needs to broadcast the block to the network. You are free for the implementation of broadcast, but you need to ensure each node receives the update of the blockchain.

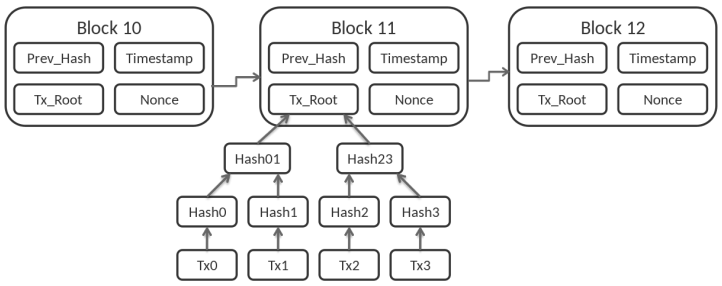
With this step, you have implemented a basic blockchain system. The next step will provide more security to the system.

3.2 Improvement of the solution

In this part, we will improve the security of the application especially against blockchain modification and Sybil attacks.

3.2.1 Implementation of the bitcoin block structure

One of the advantages of blockchain is the robustness to modification. To modify a previous block, you need to recreate all blocks between the block you modify and the actual head of the chain. To protect against modification, each block stores transactions in a Merkle tree. The transactions are hashed before placing in the Merkle tree. The ID of each block is the root of the Merkle tree which contain the hash of all transaction. Each block store the ID of its previous block. This solution facilitates the verification of a transaction and so detection of chain modification by other nodes.



You need to rework the blockchain structure to match the block structure shown in section 7 of Bitcoin paper [1].

With this step, you have implemented the real structure of a blockchain.

3.2.2 Protection against Sybil attack

During a Sybil attack, the attacker subverts the service's reputation system by creating a large number of pseudonymous identities and using them to gain a disproportionately large influence. With an over-representation in the system, an attacker has more probability to have one of its nodes choose to push a block. To counter this type of attack, we will create two types of users. Normal users which can only store and read the blockchain and validator which participate in the election consensus and can push block. In real Blockchains, validators need to pay with cryptocurrencies to authenticate. This limit the Sybil attack range.

Each validator node receives a key from the master process to sign its message. Only the validator nodes can participate in the election consensus and push a block.

Your goal is to implement the authentication system. At the master process starts, you need to add a new argument with the number of validator nodes in the system (noted V). At the experiment start, you will create V validator process.

4 Quotation

For this project, you will provide the Erlang program and a documentation report which will contain the detail of your implementation. All the Erlang files need to be in a zip file and the report in PDF format. You will submit the project on moodle.

You need to complete the first part of the project before implementing the second part of the Project.

The project is on 20 points and counts for 5 points of the final mark. The first part counts for 14 points and the second part counts for 6 points.

You have until Friday 10/12 at 6pm to submit your project on moodle.

Good luck and feel free to send me an email if you have any questions.

References

[1] S. Nakamoto, "Bitcoin: A peer-to-peer electronic cash system," 2008. [Online]. Available: https://bitcoin.org/bitcoin.pdf

[2] "Randao: Verifiable random number generation." [Online]. Available: https://www.randao.org/whitepaper/Randao_v0.85_en.pdf

[3] "Ethereum 2.0." [Online]. Available: https://ethereum.org/en/eth2/