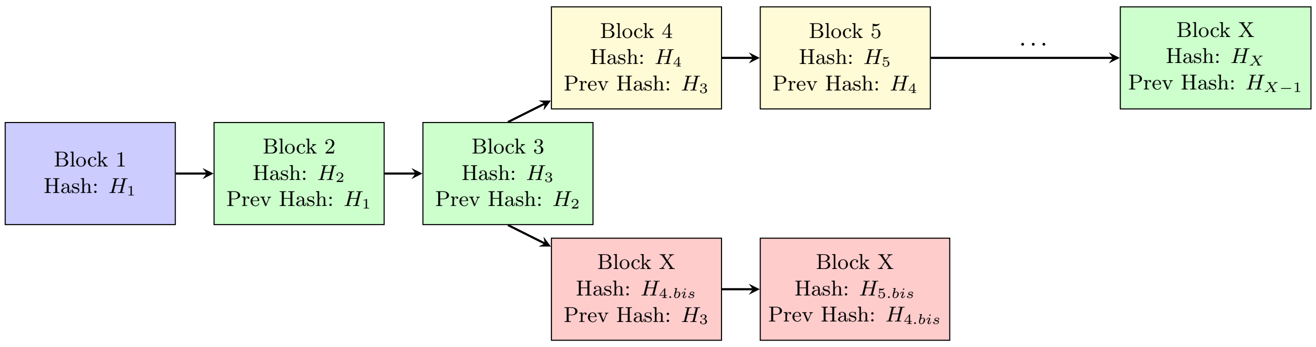
LINFO2345 - Projet
Proof of Stack

2023

Author:
Matthieu PIGAGLIO



Representation of a Blockchain

1 Introduction

A blockchain is a distributed ledger storing a growing list of records, called blocks. To guarantee the integrity of the blockchains, each block is cryptographically signed using the hash of the previous block. This mechanism protects previous blocks from malicious modifications. Several large-scale blockchains are deployed, the famous ones are Bitcoin, Ethereum or Cardano. Each of these blockchains is run by over 10,000 nodes each.

To construct a block, each peer participating in the system attempts to resolve a cryptographic puzzle to create a new block by brute force. The first node that resolves the puzzle pushes the new block in the blockchain and communicates it to other peers. This method to construct a blockchain is the Proof of Work (PoW). PoW was first introduced in Bitcoin and adopted by Ethereum at its creation. The major inconvenience of Proof of Work is its energy consumption. The creation of a block by brute force uses a huge amount of resources (CPU/GPU power, electricity, ...). Today, Bitcoin uses an amount of electricity equal to the consumption of Hungary.

To solve the problem of PoW, Proof of Stake (PoS) was created. PoS uses a group of elected peers that decide the content of future blocks. To have a chance to be elected, a node needs to lock a specific amount of cryptocurrency (32 ETH for Ethereum, approximately 60k$). This prevent sybil attacks where multiple users are created by the attacker to be overrepresented in the system. The group of elected peers uses a consensus algorithm to decide if a transaction will or not be incorporated in the next block. PoS was introduced in Ethereum with The Merge update on 15/09/22. It helps to reduce the average energy consumption of the Ethereum blockchains by 99%.

You're mission, using Erlang, is to create a PoS blockchain.

2 Project Summary

Your contribution to this project consists of 3 parts:

- Implement in Erlang a distributed ledger storing blocks.
- Implement a consensus algorithm to elect a group of nodes.
- Implement and analyze a proof of stake consensus for the blockchain.

3 Part I: Blockchain Distributed ledger

As explained previously, you will implement a blockchain-distributed ledger in Erlang.

3.1 Transaction

A blockchain contains a list of transactions. A transaction is an exchange of information between two users. For the project, we limit the information exchange to cryptocurrencies but other types of data can be exchanged in real blockchains. A transaction contains three fields:

- The address of the transaction emitter
- The address of the transaction receiver
- The amount of money transferred

To be valid a transaction needs to have these three fields none empty. During the project, we will not verify if the transaction emitter possesses the amount of money it wants to send.

3.2 Blocks

To scale transaction validation and dissemination, blockchains use blocks to store multiple transactions. For this project, the maximum size of the block will be 10 transactions. This means a block can store a maximum of 10 transactions. Blocks are constituted of several elements:

- Block number
- A Merkle tree of the hash of each transaction stored in it.
- The address of the node that constructs the block
- The hash of the last block published on the chain

- The list of transactions

Merkle Tree:

A Merkle tree is a tree where each leaf is the hash of a data block. For blockchains, the data block is a transaction. Farther nodes of the tree are the hash of the concatenation of its children. This type of tree is useful to rapidly verify the data of a large data structure, in blockchains, a list of transactions.

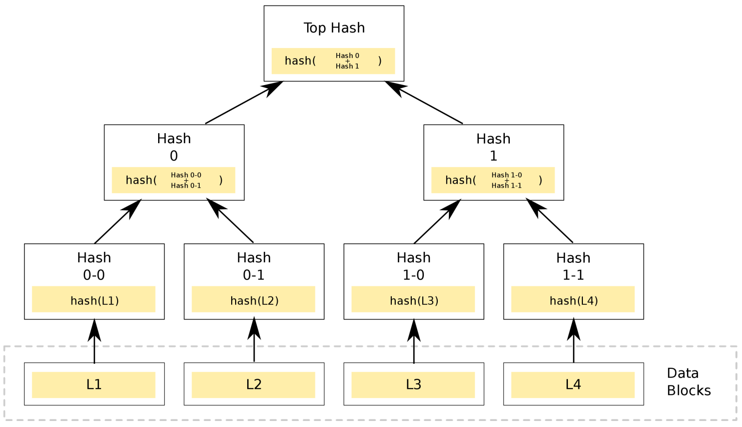


Figure 1: Representation of a Merkle tree

Hash of the previous block:

To link each block and ensure a previous block hasn't been modified each block contains the hash of the previous block.

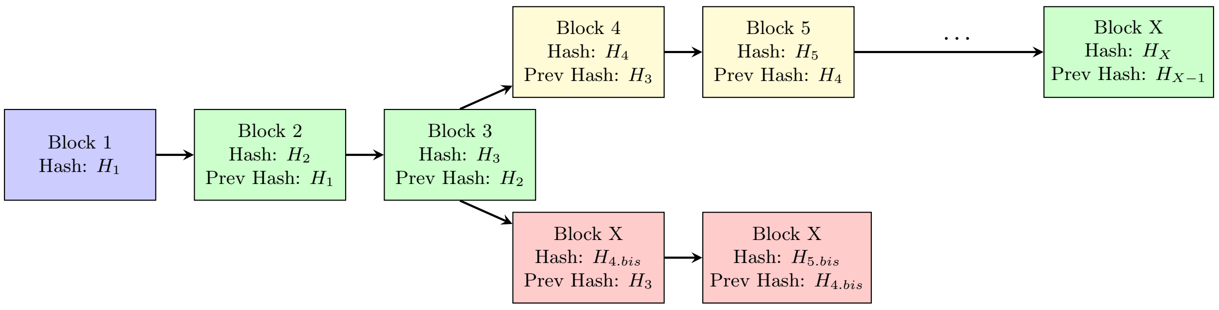


Figure 2: Representation of a blockchain

3.3 Blockchains nodes

For our blockchain, we define three types of nodes:

- Builder
- Validator
- Non validator

The address of a node takes the form of Type_X where Type is *Builder*, *Validator* or *NonValidator* and X is the Xth node of this type launched.

Builder:

The builder is a user who will forge a block and broadcast it to other nodes in the system. To construct a block, the builder take transactions in a distributed pending list of transaction. You will not implement the pending list of transactions in the project, but use a CSV file given with the project which contains a list of valid and invalid transactions.

Validator:

In proof of stake blockchain, Validators will validate if a block is valid or not before disseminating it in the network. This user will be implemented in parts II and III. In this part, the builder directly broadcasts the block to the network of nodes.

Non validator:
Nonvalidator nodes are nodes giving storage to the chain to store blocks but don't participate in the block building or validation.

3.4 Your task

- Implement a node that sends and receives messages.
- Implement a builder that can create blocks from a pending transaction CSV.
- Implement a broadcast step to disseminate a block to all other nodes.

Each node will store the blocks it has in a CSV file, where each line contains:

- Block number
- The root of the block Merkle tree
- The address of the node that constructs the block
- The hash of the last block published on the chain
- The list of transaction IDs present in the block

You are free to use the hash function of your choice but you need to mention the function used in your code **and** in the report.
You will test your system for 10, 20 and 100 nodes and provide the log in a zip archive file. The report on this part will contain, the technical choice you made for each step from creating a block to broadcasting it on the chain. You will explain the broadcast you use in your distributed ledger and the network topology of your ledger.
This part weight for 50% of the project grade

4 Part II: Consensus algorithm to elect validator group

Proof of stake uses a group of validators, called the proposer group, to validate blocks created by builders. For security reasons, the group needs to be changed every 10 blocks, we call each period 10 blocks an epoch.
To populate the proposer group, we will use a simplified version of Whisk, the Secret Leader Election algorithm used by Ethereum to elect its proposer group.
Validator
For the project, you will give each validator, a unique ID and the list of all validators at the bootstrapping of the system.

4.1 Secret Leader Election

At the end of each epoch, The first node of the proposer group, broadcast the beginning of an election step to stop block creation. Then first node of the proposer group will create a shuffled version of the list of validators. It will send it to the next validator in its list of validators. When a validator receives this shuffled list, it will reshuffle it and send it to the next validator in its validator list. These two steps are repeated until the list returns to the first node of the proposer group. When the head of the proposer group receives the shuffled list of all validators, it will select the 10% first nodes of the list. These validators will constitute the new proposer group. The first node of this list becomes the new head of the proposer group. Then the old head of the proposer group broadcasts the new proposer group to all nodes. Finally, the new head broadcasts a message to begin a new epoch and allows the builder to send new blocks.
At Bootstrap, you will provide a list of validators to be the first proposer group.

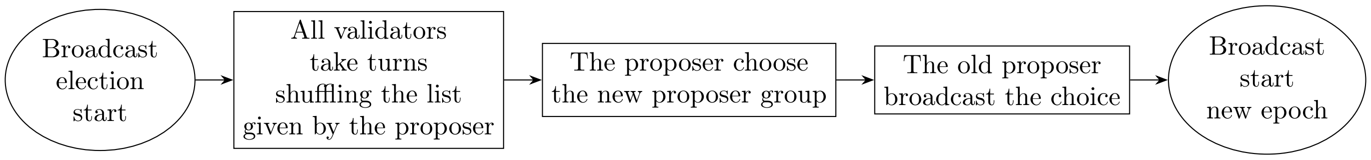

Broadcast election start
All validators take turns shuffling the list given by the proposer
The proposer choose the new proposer group
The old proposer broadcast the choice
Broadcast start new epoch

Figure 3: Election of a new proposer group

4.2 Your task

- Implement the election protocol to the system built in part I. You will test and analyze your system with different proportions of validator and size. Especially, you will monitor the time to elect a new proposer group.

This part weight for 30% of the project grade

For each validator, you will log in a text file all the operations corresponding to the election.

Hints

- *Construct the protocol and test it outside of the blockchain before integrating it.*

5 Part III: Proof of Stake

Now you have all the parts to construct a proof of stake blockchain.

To publish a block on the chain, you will follow these steps:

- The first validator of the proposer group becomes the builder.
- The builder constructs the block
- The builder sends the block to the other validators of the proposer group
- Each validator verifies the block is correctly forged (each transaction is valid)
- Each validator answers the builder if the block is correct or not
- If the majority of the validator validates the block, the builder will broadcast the block to all the nodes of the system. If the block is not valid, It discards it and constructs a new one.
- A new iteration of the process begin

Every 10 iterations, a new proposer group is elected and so, the builder will change.

5.1 Your task

- Implement proof of stake in your system.
- Evaluate the time to publish a block.
- Evaluate the limit of this protocol if the builder is a malicious node or if somes validators are malicious.

This part weight for 20% of the project grade

6 Grading

The project can be done by a group of two or alone.
You will find the weight of each part below:

- Part I: Blockchain Distributed ledger: 50%
- Part II: Consensus algorithm to elect validator group: 30%
- Part III: Proof of Stake: 20%

The project will count for 5 points on your final grade for the course.
You will provide a zip file containing 3 directories, one for each part of the project, and a report in PDF format. In each part directory, you will provide your Erlang code and the log files you produce. The report will contain all the results of your experiment and analysis. You need to complete all the previous parts of the project before beginning a new one. You have until Friday 08/12 at 6 pm to submit your project on Moodle.

Good luck and feel free to send me an email, **matthieu.pigaglio@uclouvain.be**, if you have any questions.