Design and Implementation of a Proof-of-Stake Consensus Algorithm for Blockchain

Final degree thesis

By

Eric Garcia Ribera

January 2018

Director: Albert Cabellos

Co-Director: Jordi Paillissé

Specialization: Information Technologies

UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH
Facultat d'Informàtica de Barcelona

Abstract

There is a great deal of talk about Bitcoin[1] and the blockchain among a lot of sectors in the security community as the system works without a central repository or a single administrator and without the use of digital certificates. Blockchain technology is said to be a new solution to a well-know problem in fault-tolerant distributed systems, the Byzantine Generals problem.

This document analyzes and describes a consensus algorithm implementation that will be a core part of a blockchain prototype whose main purpose is to allocate, delegate and bind IP addresses in a decentralized and secure way.
The objective of the algorithm is to allow the prototype determine which participants are able to add new blocks on the blockchain in a totally random and unpredictable manner.

Resumen

Últimamente se esta hablando mucho de Bitcoin[1] y la tecnología que utiliza, llamada blockchain o cadena de bloques, en distintos sectores relacionados con la seguridad de la red, ya que dicha tecnología proporciona una descentralización total de los sistemas y además, ofrece un funcionamiento seguro sin la necesidad de usar certificados digitales. De echo, se cree que la tecnología blockchain proporciona soluciones al problema de los generales bizantinos en sistemas distribuidos.

Este documento analiza y describe la implementación de un algoritmo de consenso que será una parte fundamental en un prototipo de blockchain cuyo principal objetivo es el de asignar, delegar y vincular direcciones IP de un modo descentralizado y seguro.
El objetivo del algoritmo es el de permitir al prototipo determinar que nodos, participantes en la red, pueden añadir nuevos bloques a la blockchain de una forma aleatoria e impredecible.

Resum

Últimament s'està parlant molt de Bitcoin[1] i la tecnologia que utilitza, anomenada blockchain o cadena de blocs, en varis sectors relacionats amb la seguretat de la xarxa, ja que aquesta tecnologia proporciona una descentralització total dels sistemes y a més, ofereix un funcionament segur sense la necessitat d'utilitzar certificats digitals. De fet, es creu que la tecnologia blockchain proporciona solucions al problema dels generals bizantins en quant a sistemes distribuïts.

Aquest document analitza i descriu la implementació d'un algorisme de consens que serà una part fonamental en un prototip de blockchain on el principal objectiu és el de assignar, delegar i vincular direccions IP de forma descentralitzada y segura.
L'objectiu de l'algorisme és el de permetre al prototip determinar quins nodes, que participen a la xarxa, poden afegir nous blocs a la blockchain de forma aleatòria i difícil de predir.

Contents

List of Figures

List of Tables

1 Context and Motivation

1.1 Introduction

Blockchain technology (**Section 1.1.1**) is gaining popularity as it is showing a new way of exchange information without the use of a coordinator reviewing every change in the system.
Despite the fact that people associates blockchain and Bitcoin[1] for financial purposes, blockchain is being used more and more on the networking field such as Blockstack[2], a global naming and storage system or IOTA[3], a cryptocurrency for the Internet-of-things (Tangle[4] technology).

The main goal of the project is to make a blockchain prototype for a network where the delegation, allocation and binding of IP addresses is decentralized and totally secure, and the aim of the document is to explain the steps taken to develop, understand and integrate a Consensus Algorithm and to argue why it is being chosen in our prototype.

This project is divided in six parts:

- **OOR Interface:** Part for the mapping and delegation resolution using LISP[5]
- **Peer to peer protocol:** Protocol used in order to communicate the multiple nodes in the network.
- **Consensus Algorithm(Focus of this document):** Algorithm used to decide, depending on the inputs, who signs a block.
- **User input:** Module to allow the communication from the user with the blockchain. Typically a file with transactions to be added.
- **Keystore:** Storage of the keys.
- **Chain persistence:** Database structure in which all the data from the Blockchain will be stored.

Figure 2 provides a diagram with the project division.

The entire project is a proposal where researchers from UPC and Cisco have tried to secure Internet routing. An introduction can be seen on Paillissé draft[6].

1.1.1 Blockchain

As a concept, a blockchain is a distributed, secure and robust database. Its main function is to store blocks, which at the same time stores transactions. Users of the blockchain communicate each other trough a peer to peer network, where transactions are the messages sent. Normally, the inserted data in a transaction

is something unique that can not be duplicated, such as coins in Bitcoin. In order to make notice of a done transaction, users broadcast the transaction to other nodes of the network, sometimes called participants. Those nodes will store temporarily the transaction and, at some fixed interval of time, a miner node will group an amount of transactions and will put the set in a block. When the block is performed and signed, the miner of the block broadcasts the block back to the network. Once a new block request arrives to other participants, they verify it and add this block after the previous block, creating a chain of blocks called blockchain. Once a block is written in the database, it is immutable.

Figure 1 presents an overview of the common structure of a Blockchain.

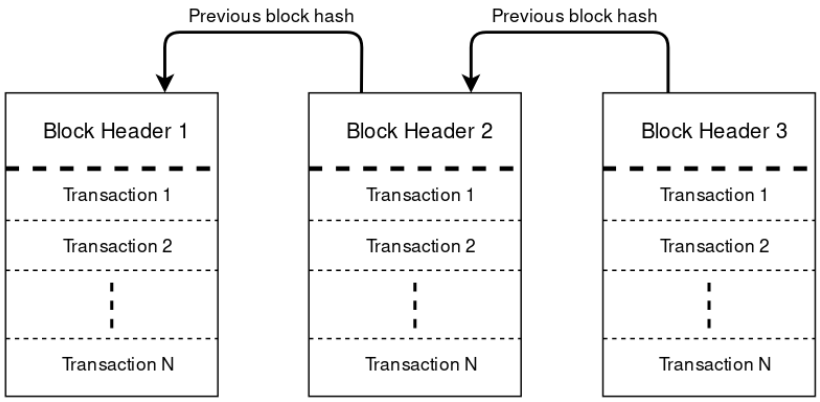


Figure 1: Blockchain structure

1.1.2 Inter-domain routing

One of the main objectives of the project is to secure the Inter-domain routing. In the Internet of today, BGP protocol is in charge of exchanging routing among autonomous systems (AS). This exchange of information is not totally secure as it is not totally verified for the autonomous systems. For example, a "corrupted" AS can inform others of a wrong routing, as this AS is well know until today, others will believe in it and will add this routing information on their tables. This situation is difficult to occur because there are policy based routing and every AS knows each other. But, as explained above, if some day one of this AS gets malicious, the routing information of this AS will be propagated and the network will be corrupted.
This same behavior occurs in the LISP protocol, the one that we will be using for the entire prototype. This protocol divides the identity of the device using LISP and so it stores IP pairs called <EID,RLOC>. EID corresponds to the endpoint identifier and RLOC is the routing locator. As it is important to separate this two pairs, LISP send packages with EID and RLOC information.

The database with the information of the EID and the RLOC is called DDT (Delegated Database Tree). LISP DDT is a database which acts similar to how it does BGP in the whole Internet, so in some manner it is insecure too.
In order to improve security in this protocols, the blockchain prototype will harden the security as the information in it can be totally trusted as it is immutable.
Moreover, DDT is hard to configure and its security is based in PKI (Public Key Infrastructure) certificates. So, with our prototype, autonomous systems and other nodes in the network will be able to avoid the use of PKI in order to trust in the certificates. This will be possible due to the use of a totally trusted database, in which the information can be retrieved, i.e. the blockchain.
Last but not least, the whole prototype infrastructure will be decentralized, so it will not belong to an entity and so it will be more secure.

By using the blockchain to secure this part of the Internet, users will achieve the same level, or even more, security, it will be easier to set up and to use and the hierarchical structure will disappear. Finally and the most important, we will learn about blockchains and its uses in the networking world.

1.2 Stakeholders

The stakeholders involved in this project are described below.

Developer The main function of the developer in this project is to do research on the multiple available algorithms for consensus purposes, develop the selected algorithm and integrate it with the rest of the parts in the blockchain.

Project partners Peer to peer protocol, database structure and mapping are other pieces of the whole project that are going to be implemented by the project partners. They will assure, as the developer, the correct integration with the other core parts of the project.

Director and co-director Their role will be the guidance and supervision of the work done by the developer and the project partners. They will also help on the comprehension of the technology and the implementation on the network.

End-user As the main goal of the project is to secure the Internet, anyone who uses Internet nowadays, will be beneficiary of our project, i.e. companies, users, government entities, etc.

2 Scope of the project

As explained in the Introduction, the architecture of the blockchain is huge (as can be seen in Figure 2), so the project is divided in six core parts where I will be developing the Consensus Algorithm. It consists on researching, finding, developing and integrating a Consensus Algorithm in a blockchain in order to let the other parts of the prototype work together to fully implement a new blockchain to secure the Internet. The consensus algorithm needs to be well defined, as it is the core part of the blockchain. The main responsibility of the algorithm is to select a new signer of a block each round. This selection must be totally random, with the objective of being impossible to predict and manipulate beforehand. For this reason, it needs to be studied carefully.

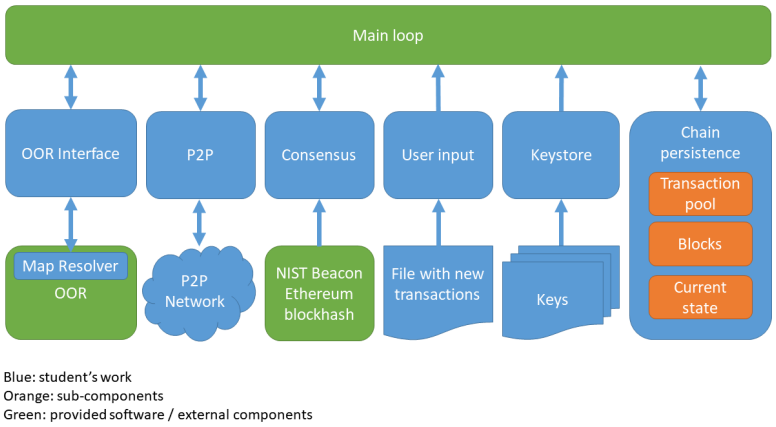


Figure 2: Blockchain prototype modules diagram. Consensus algorithm is the focus of the document.

2.1 Objectives

The part of the project explained in this document, the Consensus Algorithm, should be able to decide, depending on various variables, who signs the block at some point of time. This decision must be equal for all the participants and also it must be totally random. Moreover, it is expected for the algorithm to be fast enough in order to support a high number of transactions per minute, specifically the objective is to be faster or at the same level than Bitcoin. Lastly the module should protect the blockchain from network attacks.

Firstly, in order to understand what a consensus algorithm should do, I will be studying and comparing other algorithms to collect enough information for the completion of a custom consensus algorithm. It is expected to decide a

final algorithm by the end of October 2017. After that, the development of the consensus algorithm will take place.

Secondly, once the algorithm is well defined and developed, correction of code errors and testing should be done in order to corroborate the proper functioning of the blockchain.

Finally, by the end of the final degree thesis, the consensus algorithm should allow the prototype to find randomly a block signer in a reasonable time and also it must protect the network from the most common attacks.

2.2 Challenges and risks

As we are working with state of the art technologies, such as blockchain and proof-of-stake algorithms (explained in **Section 4.2**), there can be a lot of obstacles and difficulties during the development of the project. Some of them are listed below:

- There is no clear algorithm in order to fully secure the correct operation of a blockchain based on proof of stake. This is very important, as the system should be able to correctly handle attacks received by corrupted nodes, otherwise, the system could be compromised and lead by malicious nodes.
- The algorithm selected in order to handle the consensus part, should be efficient enough to support a high number of transactions created per second. As each block groups a set of transactions, the creation of blocks must be fast.
- As the election of block signer is decentralized, some organizations such as IANA, the Internet Assigned Numbers Authority, could see the entire project as a loss of their power, so they could be skeptic on the objectives of the project.
- The time destined to develop the project can be an obstacle too, as it is limited and should be used correctly.
- Bugs on the code are always an obstacle, and considering the development of a Consensus Algorithm non trivial, there can appear some of them during the implementation. In order to avoid a lose of time trying to correct code bugs, the development should be accurate enough to use the time efficiently.

3 State of the art

Bitcoin and the Blockchain technology have started at 2009 with a lot of controversy around different areas of study, e.g., security, environment, efficiency, etc. One of the most discussed aspects of Bitcoin is the proof of work or PoW. As Bitcoin uses PoW as its way of choosing a new block signer, nodes in the network have to mine, i.e., to compute a complex mathematical problem in order to add a block, thus demanding more and more computational effort.
Nowadays, the main problem of Bitcoin is how to solve this problem efficiently, as it has to find a hash starting with a fixed amount of zeros, which can only be solved through brute force. This can be controversial due to the amount of energy "wasted" trying to mine a block.

Regarding the prototype, we have opted for a different block signer election, it is called proof of stake or PoS. The basics of this Algorithm is that participants or nodes in the network with more stake in the Blockchain, will be able to add blocks more often. Normally, for each new expected block, a new signer of the block will be selected, through random criteria, from the list of participants, given its weight, i.e., the amount of stake they have. With this in mind, the power effort destined to mine a block is much less compared with PoW. More details about PoS can be found in **Section 4.2**

About PoS, Ethereum[7] community is developing, at the time of this writing, an algorithm called Casper based on PoS. There has been considerable interest in this algorithm as Ethereum is, following Bitcoin, one of the most popular cryptocurrencies in the market. Although Casper is expected to be secure, efficient and reliable, we have decided to move forward to other PoS Algorithms, e.g. Ouroboros[8] or ALGORAND[9] , as it is not finished yet.

Looking at the entire project, not only the Consensus Algorithm part, the objective is to secure the allocation, delegation and bindings of IP addresses. So the project is related with improving the Internet by the use of a blockchain, where there are other projects that tries to do so, few of them are Namecoin[10], which aims to improve the Internet by decentralizing and improving the speed of DNS entities, and the previously mentioned Blockstack[2].

In terms of blockchain technology, there are other implementations that should be considered in the future. For example the Tangle[4]. This technology is similar in idea as the blockchain but not in practice. It is a directed acyclic graph (DAG) which grows as new transactions are added. Basically it offers scalability, no fees and more security by changing the technology from a chain of blocks to a DAG. In **Section 6.2.3** a larger explanation about the Tangle and its possibles features for our prototype is done.

4 Consensus Algorithm research

4.1 Proof of work

The main idea behind proof of work is to work hard in order to demonstrate the commitment a node has with the network and thus, receiving a reward for the work done. This is often known as mining.
In Bitcoin, for example, a reward is given to the first node that finds a specific hash by hashing some information. The correct hash is a SHA-256 string made by the transactions of the block, the previous block hash and the nonce, which consists on having a leading number of zero bits. Two of the three parameters of the SHA-256 hash are static, but one of them, the nonce, is selected by using brute force, as it is the only known way to find the correct hash.

Proof of work is used nowadays in blockchains because it provides security to the chain as it is very complex to modify old blocks due to the fact that the valid chain for the vast majority of the nodes is the longest one, i.e. the chain with more accumulated computing power. This property ensures that data in old blocks will not be modified, as it requires a hard computational work in order to generate "by your self" a blockchain longer that the one that is shared with the rest of the network.
Despite proof of work seems to be secure, there are some drawbacks that should be mentioned:

- 34% attack: Some research has been done[11] about the problem of having a group with at least 34% of the hash mining power over the network. Such group could modify the chain and thus turning the blockchain insecure.
- Increasing complexity: One of the properties of proof of work blockchains is that the complexity of mining a block increases over time. This can be seen as a strength of the protocol, because it protects the network against new hardware, but it is not. As the complexity increases, there are only few mining farms[1] that can spend enough money to achieve new specific hardware in order to mine a cryptocurrency. This can lead do a centralization of the mining power, which is contrary to the idea of decentralization of blockchain.
- Energy waste: As the complexity of mining a block increases, it is needed more hashing power, which can be seen as more energy used. This is nowadays the main problem to solve on blockchains, as there are a lot of cryptocurrencies using proof of work, i.e. wasting a lot of energy.

[1] Group of facilities where its main objective is to mine cryptocurrencies by the use of specific hardware.

4.2 Proof of stake

The basics of this Algorithm is that participants or nodes in the network with more stake in the blockchain, will be able to add blocks more often. Normally, for each new expected block, a new signer of the block will be selected, through random criteria, from the list of participants given its weight, i.e., the amount of stake they have. With this in mind, the power effort destined to mine a block is much less compared with PoW, thanks to the no hash power needed.
Although PoS give us the advantage of wasting much less energy than PoW and some more decentralization, it has some drawbacks:

- Monopoly: The main drawback of PoS algorithms is that a potential monopoly from the major stakeholders of the network could be possible. This possibility puts those algorithms at the same level of PoW, situation that should be avoided.
- Nothing at stake: Regarding any proof-of-work blockchain, in front of a double-spend attack[2] or in a chain fork[3], nodes will have three options to choose:
  - Follow chain 1 and probably get benefit if chain 1 wins.
  - Follow chain 2 and probably get benefit if chain 2 wins.
  - Do nothing and get no benefit

  Only one of these three options can be chosen, otherwise you would waste energy mining on the wrong chain, costing you a lot of money in terms of electricity.
  Using the proof of stake protocol, this situation changes. Having in mind the small cost of energy on creating a block in a PoS environment, you could easily follow the two created chains using less power than a proof-of-work blockchain miner do.
  If this situation occurs, the blockchain will have no order and thus be a chaos. Moreover, if there is more than one unique chain being followed, it is probable to have more forks, which could cause more chaos.

On the other hand, proof of stake is suitable for us in this aspects:

- Nothing at stake is not worth it in our environment as the major stakeholders are the most interested in the proper functioning of the chain. In order to succeed in a nothing at stake attack, a high percentage of the nodes in the network should follow more than one chain when a fork occurs. As the nodes with more stake are the ones interested on a healthy chain, it makes no sense to follow the attack.

[2]This attack occurs where an attacker spends a set of tokens (coins in Bitcoin) in more than one transaction, winning tokens maliciously.

[3]Situation where a new chain is created from another one. This causes serious problems as nodes must choose one or the other chain to follow.

- Energy is not wasted as it is in proof-of-work blockchains. Typically, the election of a new signer is made by the use of random criteria and weighted depending on the stake a node has. So making more attempts per second do not increase the possibility of being selected as the new signer. Hence, a reduction in time and energy can be noticed.
- Faster chains are possible by using proof-of-stake algorithms. Without the need of computing a lot of hashes to find the correct one, is possible to create a chain with a higher rate of block generation. Nowadays, taking Bitcoin as an example of a proof-of-work blockchain, one block is created every 10 minutes. This situation is not scalable, so proof of stake offers a faster way for block creation by reducing the time to chose a new block signer.

4.2.1 Well known attacks

Proof of stake is the focus of this writing and so the algorithm to be developed should follow the idea of a PoS algorithm. For this reason, is interesting to comment the most well known attacks in PoS:

- Stake grinding: This attack consists on, little by little, take advantage of a manipulated consensus algorithm in order to have more stake. This is possible when an attacker has the opportunity to sign a block. When her/his turn arrives, the attacker creates more than one block and analyzes the future probability of signing again a block depending on the block chosen. So progressively, the attacker will gain more stake and thus sign more blocks, with the objective of be the major stakeholder inside the network.
- No participation: In PoS, the election of a new block signer should be random enough to avoid repeating the same signer for a long time. The direct consequence of that is the no interest from the nodes to stay on line, as they will not get benefit from the network. If this situation continues for long periods of time, the same stakeholders will sign blocks more often, obtaining more and more stake. Having a decentralized blockchain.
- Range attacks: The attack consists on creating a sub-chain (fork) at some point of time with the idea of be the selected chain for other nodes in the system. With this, the attacker will have the option of modifying the history of the chain, obtaining a benefit from it.

4.2.2 Ouroboros

Ouroboros[8] is one of the firsts approaches for a securely proof-of-stake algorithm for blockchain purposes. So its main focus is to find a way for a randomized leader election which incurs in a complete consensus through the network.

To achieve this, the algorithm uses time, slots and synchrony as main features. Time is divided in slots, and for each slot it is supposed to be minimum one block. At all time, nodes or participants are synchronized among them, thanks to high precision clock time.
At the beginning, where the system has started recently and so there are not yet a lot of attackers, a genesis block is created, distributing the stake among the first stakeholders. This first block creation is totally hardcoded into the code at the beginning. At this time, there is low probability of receiving an attack, because these new major stakeholders want the network to grow instead of break it.
From now on, nodes will communicate each other in order to determine the epochs. For each epoch, participants should achieve consensus on whose nodes will sign the blocks inside the next epoch. This communicative process is done by the use of coin tossing. This property ensures a verifiable random election between two parties, so each other ensures that the other party is not cheating the network, and it works as follows.

- Alice generates a random vector called S, which can be a string, a number, etc. This vector is encrypted through a $COM(r,S) = C$ function. Where r is a random parameter and C is the encrypted message.
- Alice sends the encrypted message to Bob.
- Bob generates a random vector called S' and sends it to Alice in the clear, i.e. without encryption. S' is of the same size as S.
- Alice sends the encryption key to Bob in order for Bob to know S.
- At this time, both Alice and Bob knows S' and S respectively.
- Both parties, XOR S and S'. ($\hat{S} = S\ XOR\ S'$), resulting on the same $\hat{S}$ for each party. This is the new random vector generated by two parties.

With this in mind, stakeholders implicated in an epoch will work together in order to determine a new election leader. Moving forward to new epochs.

Like others proof-of-stake algorithms, Ouroboros stands for a 2/3 of honest parties in order for the blockchain to stay secure and reliable in front of attacks. They argue this statement by using random walks probabilities, a common used probabilistic area in blockchains. And it is based on the fact that if we work with a probability of strictly less than one half, the random walk will move its position to one side because it is biased. So for an attacker of less than one half of the power, will be very difficult to end the attack successfully. More information can be found on the paper[8] and in a video[12] explained by the author of Ouroboros.

Although this algorithm is proved in terms of security and network attacks, it has some drawbacks that will make us move forward on to the creation of a custom algorithm:

- In order to achieve total consensus, all the nodes should be synchronized with high precision, it is a requirement for Ouroboros in order to work as expected. Although nowadays clock servers have a good precision, there can exist some latency in the network that can prevent the algorithm to work as expected, so in our situation we will avoid that.
- As the signer of a block is known some time before being elected, there is the possibility to launch a DDoS attack to that server, if it is known, and prevent it from signing the block. Slowing down the network.
- As a complete algorithm, Ouroboros is difficult to understand and to develop in a few months and considering the time we have in order to develop and integrate the algorithm in the blockchain prototype, we agree on developing a new and easier to implement consensus algorithm.

4.2.3 Algorand

Algorand is another alternative to consider when we search for an efficient and secure protocol to follow when developing a consensus algorithm for blockchain purposes.
Despite being complex and difficult to understand, it offers a good alternative to improve our blockchain prototype.
A brief summary of the algorithm will be explained in this section, but more information can be found on the original paper[9].

Algorand is based on following a number of steps in order to decide the block signer. Each step can be understood as an elimination round.
Each round starts when finishes the previous one, i.e. when we have found the new block signer. Once this signer broadcasts the new block, a new round starts.
At the beginning of each step, a hash (sig(r,s,q(r-1))) is calculated and converted into a decimal between 0 and 1, i.e. "p". Where "r" is the number of the block, "s" is the current step (at the beginning is 1) and q(r-1) is a random value taken from the previous block. If the value extracted from the hash, and converted into a decimal, is lower than "p", this node should participate in the step "s".

Each participant has a master key which purpose is to generate new temporal keys. Every new generated private key is destroyed after signing a message with it.
Every two years approximately, the master key must be renewed.

Step 1, Leader election: The node should prepare a candidate block, sign it and then send it to the network with sig(r,1,q(r-1)).

Step 2, Graded Consensus step 1: As long as we receive sig(r,1,q(r-1)) messages, the node must calculate the hash of this messages and then the candidate block will be the one with the minimum hash calculated. Then the new

candidate block is signed with a temporal private key and send it to the network with sig(r,2,q(r-1)). Finally the temporal private key is destroyed.

Step 3, Graded Consensus step 2: As long as we receive sig(r,2,q(r-1)) messages, the node should verify the following: If more than 2/3 of the arrived signatures belongs to the same bloc, the node signs this bloc and proceeds by sending the block signed and sig(r,3,q(r-1)). Otherwise an empty string is signed and send it to the network with sig(r,3,q(r-1)). Finally the temporal private key is destroyed.

Step 4, Graded Consensus output: If more than 2/3 of the new received messages belong to the same block, this is the candidate and a new variable "g" with value 2 is set.
If more than 1/3 but less than 2/3 of the new receiver messages belong to the same block, this is the candidate and a new variable "g" with value 1 is set.
Otherwise there is no consensus so candidate block is an empty block and new variable "g" value is 0.
Then, if "g" = 2, "b" = 0, otherwise "b" = 1. Finally the node signs "b" and sends it with sig(r,4,q(r-1)).

Step 5, BBA[4] with output 0: If more than 2/3 of all received "b" messages are equal to zero, the node stops the search and sets the definitive block the one received.
If more than 2/3 of the received "b" messages are equal to 1, the node stops the search and sets a void block because of the no consensus situation.
Otherwise "b" is equal to zero and "b" is sent with sig(r,5,q(r-1)).

Step 6, BBA with output 1: Same as Step 5 but if there is no consensus, "b" equal to 1 is sent with sig(r,6,q(r-1)).

Step 7, BBA with output different than 0 or 1: Same as Step 5 but if there is no consensus, "b" is equal to the lower hash of the sig(r,6,q(r-1)) messages received. Then "b" is sent with sig(r,7,q(r-1)).

Step 8, Last step (low probability): Execute again BBA and Step 5 but if there is no consensus, the node stops. No consensus found so the block remains void, "b" equals to 1 and sig(r,8,q(r-1)) is sent.

Algorand presents a secure and efficient way to implement the consensus algorithm of a public and distributed ledger. As seen trough the eight steps to arrive to a consensus, this algorithm fits perfectly for most of the proof-of-stake blockchains. However, this requires a lot of time on understanding completely the algorithm and then implementing it to fully work together with the other parts of a blockchain.
For this reasons and more, the Algorand algorithm is probably a good choice in order to achieve consensus through all the nodes in the network. Nevertheless, we have some time restrictions that prevent us from implementing this solution

[4]Protocol that is based on the honesty of 2/3 of the participants. It is started in the step number five of Algorand.

in our prototype. Moreover, as we can see in the paper[9], Algorand uses a lot of patents that should be carefully read and understood in order to use the algorithm, requiring more time to decide if it can be used or not.

4.3 Pros and cons of developing a new algorithm

As seen in this section, developing and understanding new algorithms is not easy. Moreover, there are algorithms that are focused on only one objective which may not help us in order to develop our prototype. For this reason, we are going to implement and develop a new algorithm that fits with our blockchain prototype.
However, this is not trivial and needs a lot of time to understand how other algorithms work in order to use some ideas as improvements.

First we are going to see the cons of developing a new algorithm:

- Developing a new algorithm often requires starting from zero, so the road to go is much longer than if we start with something already developed as Algorand or Ouroboros. However, creating something new, will allow us to integrate something that fits perfectly on our prototype.
- Although creating something from zero can help us on being more specific and more accurate in our algorithm, we will need to spend more time on testing the new implementation. This is because the other mentioned algorithms have been tested and corrected for a long time, so they are better prepared for stay in production than us.
- Due to a lack of time in the development, new algorithms fail more frequently than old ones and also tend to be less efficient.

On the other hand, new algorithms can be a good option depending on the situation:

- Not all the algorithms are generic, so sometimes we can find a good solution for one kind of blockchain but this may not fit for our prototype. For this reason we are going to develop an algorithm that completely fits our implementation.
- As long as we have seen, most of the well proved algorithms are difficult to understand and to implement because of the well performed they are. So sometimes it can be hard to fully adapt this algorithms to our prototype. We have opted for a specific solution but easy to implement in order to test our blockchain.
- Despite most of the algorithms proposed in this section are well explained, they are not easy to understand, so they require a lot of time to study the structure of the possible implementation, to study the idea and so on. Nevertheless, an algorithm developed for our own purpose and of course,

noticeably simpler than the ones presented here, is easier to understand and faster to implement.

- Although nowadays we can see an increase of the free software movement, not all the algorithms are free to use. Sometimes they present a list of patents used, copyright protection and other issues that can forbid our usage or implementation on them. On the other hand, creating a new algorithm will allow us to be hundred percent secure that we are not committing any law infringement.

5 Development of a new algorithm

As we have seen in the previous section, there are protocols that can be used as a consensus algorithm for our blockchain prototype. However, they are difficult to understand and to develop in a few months, so we have decided to develop a new algorithm that fully fits in the prototype.

Proof-of-stake protocols do not verify the block signer election through the computation of complex mathematical problems. Instead, verification is done by the use of the stake and with some random criteria. This behavior will allow the whole blockchain to run without the use of many quantities of energy, but achieving a random consensus through all the network.

5.1 Random signer election (why it is important?)

Randomness in a proof-of-stake protocol is crucial because of its security implications. Apart from the weighted selection depending on the stake a node has, all participants must achieve consensus in which is the new block signer. As this selection must be random, in order to be as much democratic as possible, honest participants should have the quality to detect when a malicious node is trying to cheat the network. With this in mind, we can think in an algorithm in which each node decides, by randomness selection, one node from the network. Then, depending on the number of times a node is selected, it can be chosen to sign a block or not. This can be seen with more detail in the first consensus algorithm approach in **Section 5.2**. One of the main problems of such approach is that the network can not verify the honesty a node has, so it is possible to have malicious nodes and that they select their winning node directly instead of doing it through a random process.

In a proof-of-stake algorithm, randomness is important to forbid participants cheat the network by choosing a malicious node in order to damage the current blockchain. If malicious nodes are selected continuously, they can behave in a way in which the new created blocks are incorrectly created, slowing the network each time they became selected. For this reason is important to achieve a random consensus selection to hinder the option of slow down the system by malicious participants.

5.2 First algorithm approach

In order to make participants agree on a block signer, random choice is a must. So our first approach was focused on a round based selection.
The main idea of this algorithm is to randomly find an IP address and so its owner, who will be the winner for the selection of the block signer.

In the first place, the block signer of the previous block (n-1) will randomly choose a list of participants public keys to be candidates to sign the block "n". For example we will have the list [pk24,pk125876,pk3765,...,pk4] where the length of the list is equal to the number of participants in the network.
In IPv4, as we have 2^{32} addresses, we will need to have 32 bits to choose an address. For this, the list is divided by 32, having a total of thirty-two groups. Each group will select a bit, being this one bit of the address.

Secondly, each member of the group will choose randomly a bit, one or zero. For each member of the group the algorithm will do a XOR with each bit (0 XOR 1 XOR 1 XOR 0 ... XOR 0 XOR 0 XOR 1). So in the end there will be a bit for each group. For example, if the 32nd group selects the bit one, the address will look like X.X.X.yyyyyyy1 where "X" represents 8 bits and "y" 1 bit. If the 31st group selects the bit 0, the address will be X.X.X.yyyyyy01.

Lastly, once each group has selected its own bit, a message will be issued to the rest of the groups saying that, for example, group 32nd has found its bit. When all the groups have the bit selected, each group will exchange their chosen bit, creating the final address whose owner will sign the block. It is a must to wait until every group in the network finds its chosen bit before broadcasting their bit selection. Otherwise, a malicious group would be able to modify the final result.

The algorithm presented above could work as expected in a real network, but there are some problems that change our mind in the implementation of the algorithm:

- Too many messages needed. In order to work as expected the network will need to broadcast a lot of messages at the same time. At the beginning of the prototype is possible for this to work, but once the network has millions of nodes, this would be a chaos.
- An "army" of malicious nodes can try to bring closer the selection of the address to one of its owned addresses. This can be done if all the members of a group are malicious, so they can choose always the same bit. Of course this is very difficult to happen, because at the beginning of the round, the previous block signer randomly mixes the list in order to prevent all the members of a group being malicious, but if unfortunately the previous signer is malicious (because this can happen), they can alter the selection.
- There is no way to control that each participant selects its bit randomly. This means that a group of participants can achieve a consensus and cheat the network. For this reason we should use a unique random source, this is explained in the next section.

Despite the mentioned problems above, coin tossing can be used in order to achieve truly random election. This is explained in the **Section 4.2.2** and basically it would allow the network to verify all the random numbers selected by all the nodes. This solution could have been the savior of this implementation,

but unfortunately, inn order to achieve the coin tossing consensus between all the network, a high quantity of messages should be sent and the network should be synchronized with every node. This requirements complicate the implementation, so we have decided to move forward into another algorithm.

5.3 Unique random source

As it is difficult to create an algorithm to allow the network detect when a node is malicious or not, our algorithm will use only one unique random source. This will force participants to achieve a consensus by using only one source of randomness.

As long as we have seen, if we achieve consensus in a random number, an address can be extracted from it, giving the participants the answer to the question of "Who is the next signer?". But to find a randomness source is not easy, because it needs to have enough entropy to not be predictable, otherwise an attacker can predict the election beforehand.
For this purpose, after discussing a lot, we arrive at the conclusion of using the entropy of a blockchain.

5.4 Blockchain entropy

As can be seen in *On Bitcoin as a public randomness source* paper[13], 32 bits of entropy can be enough to verify something that changes fast, like the block creation of a blockchain. And taking into account that Bitcoin blocks have a lot of transactions on them, i.e. a lot of information, we can think correctly that the hash of a block can have enough entropy to fit in our requirements. However, the information of the transaction can be known time before the creation of a block by the mining pools, so in some manner, this bits of entropy should not be counted in order to agree on taking Bitcoin as a source of entropy.

In the paper mentioned above[13], it is argued that thanks to the proof-of-work complexity, there exists a minimum of entropy that is given by the timestamp (seconds), the Merkle tree hash of transactions and the nonce.
The nonce gives us a lot of entropy, as it is very difficult to create and even more to predict the nonce long time before the creation of a block. In fact if we consider the difficulty "d" of finding a nonce as 68, the possibility of guessing the nonce is around 2^{-d} so 2^{-68}.

Moreover, we have the entropy of the timestamp of when the block was created. This time can only be predicted if we can guess the nonce, so it is very difficult to do so.

In conclusion we can say that, after the research done by the authors of [13], the hash of a Bitcoin block can be used as a source of entropy as it is, a priori,

unpredictable.

5.5 Final algorithm implementation

In order to use a unique source of randomness for every participant in the network, we decided that, as said in **Sections 5.3 and 5.4**, actual blockchains such as Bitcoin or Ethereum can be used for this purpose. In fact, as Ethereum's blockchain is faster, in terms of block creation, than Bitcoin's one, we will use Ethereum blocks to have a different hash block approximately every fourteen seconds.

5.5.1 Ethereum block hash

To argue the decision of using Ethereum instead of Bitcoin, an explanation of current blockchains entropy has been made in **Section 5.4**, so this applies for almost every blockchain that uses a proof-of-work algorithm. Moreover, as explained above, the block creation rate is much faster in Ethereum, so we will use this to extract the entropy needed for our algorithm.

5.5.2 NIST beacon generator

Furthermore, considering other sources of entropy, it is possible to mix different hashes by using an extractor function[5] to achieve, if possible, more entropy to the hash. So a good choice for that can be the NIST beacon hash generator[14]. This beacon issues a string of 512 bits every 60 seconds. In theory, this bit-string is unpredictable, so attackers can not predict the result beforehand. It is also autonomous, meaning that the source of entropy is totally unexposed to possible attempts to alter the random information, and it is consistent, so each participant that retrieves information from this beacon, will receive the same output as any other participant in the system at some time fixed.

As explained before, an extractor function will be used to add entropy to our hash string. This function will consist on using a SHA256 hash to add more entropy.

5.5.3 Extraction of the random hash

First we will take the 256 bits hash of an Ethereum validated block. Once we have this hash, it will be mixed with the 512 bits hash from the NIST beacon by concatenating each bit of the hash through an XOR bit-a-bit function.
Then, this 512 bits mixed hash will be used as a parameter for the extractor

[5] An extractor is a function that, given an input with some minimal entropy, it will output something with much more entropy than before.

function, which will give an output of 256 bits.
It is important to mention that the request made to the Ethereum API and to NIST beacon is depending on a timestamp in order to ensure the unique value for every moment of time. Also it may happen for our blockchain to create blocks faster than Ethereum, so at some point of time we will get the same Ethereum or NIST hash. To avoid this, the algorithm will wait until a new Ethereum block is created.

5.5.4 IP address extraction

Once we have a high entropy hash to use, an IP address from it should be extracted. By using a similar behavior to our first implementation algorithm, the functionality can be divided in two scenarios.
If the signer belongs to the IPv4 protocol, the algorithm will divide the 256 bits of the hash by 32. Then for every eight positions of the 256 bits list, will concatenate each bit, resulting on only one bit of output for each group. In the end we will have 32 bits resulting on an address.
As above, if the signer belongs to the IPv6 protocol, the algorithm will divide the 256 bits of the hash by 128. Then for every two positions of the 256 bits list, will concatenate each bit, resulting on only one bit of output for each group. In the end we will have 128 bits resulting on an address.

In the end what we have is a completely random address that will belong to an owner. With this in mind, we achieve one of the main points of a proof-of-stake algorithm, which is a random selection weighted depending on the stake an owner has. As much addresses an owner has, higher is the probability to sign the block.

5.6 Algorithm drawbacks

The main drawbacks of this algorithm are going to be described in this section.

Besides the algorithm presented above fits correctly in the prototype, there are some aspects that could be improved and also new functionalities could be added.

- Fork handling and sub-networks attack: Our custom algorithm protects the chain from having forks in some situations. As there is always a unique signer for every timestamp, the consensus is achieved by all the participants.
  However, as the algorithm is totally democratic, there is the possibility for a malicious node to be randomly selected to sign the new bloc. If this occurs, the node can create more than one block and send different blocks to the network with different timestamps on them. This behavior will lead in a catastrophic situation for the chain, as each block sent by

the malicious node will mean a different signer, creating sub-networks depending on the first block that arrives.

The situation explained above could be avoided in two ways: First way is to wait confirmation for other nodes about this new block. After receiving a block, a node can randomly ask the other nodes for the hash of the received block. If the hash is the same to the one that arrived to you before, means that the network is synchronized in terms of blocks. Otherwise, the node will reject the block and will wait for a different signer.
Another way is to use the number of block instead of the timestamp. Nodes waiting for a new block to be added, will wait for an specific number, so they will not be cheated.

All the nodes in our prototype are honest, so there will be no protection against these attacks.

- External dependencies: As the main goal of the algorithm is to achieve a unique source of randomness, we have opted to use external sources such as NIST or the Ethereum blockchain. This is not the ideal situation, because if fore some reason Ethereum or NIST stops working, our algorithm will not achieve consensus.

6 Project management

6.1 Methodology

The methodology used in this project should be agreed with the members of the project in order to work together efficiently and in consonance. The method used for this project has been Agile software development (Figure 3), where communication among team members is crucial, working on code is supposed to be mandatory and responding to change is supposed to be rapidly applied.

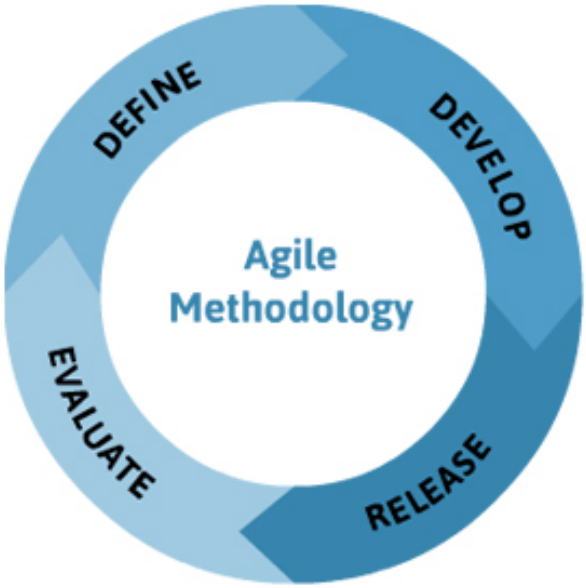


Figure 3: Agile methodology. (source [15])

Complementing the Agile methodology, there will be weekly meetings with the team members, the co-director and the director of the project. The objective of the meetings will be to summarize the work done during the week and to discuss relevant aspects for the good performance of the prototype.

To complete the communication and integration of the project with the different members of the group, we have opted to use a version control system to fill our necessities. Specifically, we will be using git. Git will allow us to upload and download recent versions of the project without hard work, simplifying a

lot the process of creating new versions, debugging and documenting the code programmed.

Once we have our prototype working, the next step will be to test it in a new beta environment. This environment is set with a network protocol called LISP[5]. Using this beta program will allow us to simulate our prototype in a very close to reality environment, which will give us the real conclusions.

6.1.1 Development tools

In order to achieve a correct integration with all the parts of the project, we will be using the following tools:

- Git: As explained above, we will use git version control as a repository for our code.
- Google Drive: This cloud sharing system will allow us to share important documents, write documents in a live format and to organize better the whole project.
- Skype: Skype platform will bring us the opportunity to communicate with all members of the project even though they are located in different places.

6.1.2 Validation

In order to validate our project, we will use a beta network to test the project. This network is very similar to the real environment we have in the Internet setup, so this will give us the real result of the entire project.

6.2 Temporal planning

6.2.1 Estimated duration

The TFG project is estimated to finish in 4 months. However, in order to improve the quality of the project, I've started with it two months earlier, being a total of 6 months estimating its completion in January 2017.

6.2.2 Considerations

It must be noticed that, due to the characteristics of this project, the planning is subject to modifications depending on how the project is being developed. Furthermore, it will be implemented by a team group, where each member will be responsible of its own part, but in the end, we should previously plan how things are going to be developed in order to stay up to date with the project.

6.2.3 Project planning

The coverage of the GEP course can be divided in this stages:

- Context and scope of the project (10h)
- Project planning (5h)
- Budget and sustainability (5h)
- First oral presentation (10h)
- Specification (5h)
- Oral presentation and final document (15h)

All this stages will be done by the Project Manager under the supervision of the Co-director and Director of the project.

By the end of the thesis the planning is the following:

- Project planning (50h)
- Research about blockchains (60h)
- Analysis of requirements (40h)
- Research on PoS (70h)
- Design of the algorithm (80h)
- Algorithm development (50h)
- API creation (20h)
- Module tests (40h)
- Documentation of the final document (50h)
- Defense preparation (10h)
- TOTAL (470h)

The resources used in order to complete this stages are a Personal Computer, lent by the university, a text editor to write the project and several Linux virtual machines for the test environment. All of this will be lent by the university.

As a human resources we have only one person for this part of the project, the Consensus algorithm. Nevertheless, regarding the entire project, we have more parts, so we will be 4 developers implementing the project and the director to manage it.

6.2.4 Project analysis

This phase will cover the analysis of the project and its possible implementations.

Before starting the development of a project, there is a need to understand what are the technologies involved in it, the objectives to achieve and the requirements to have. In the case of this scenario, the technology is the most important part, as it is new and hard to understand. This technology is called the blockchain. In order to achieve the state of the art knowledge of the technology, we must read a lot of papers to understand what a blockchain is, how it works and how it should be developed.

Once we know the technology, we will be able to design and analyze a plan in order to develop the project.

6.2.5 Project design

This part consists on creating the software architecture in order to get a working setup and the correct performance expected for the completion of the project.

As a reference we have a lot of blockchains developed in the last 5 years or so that can be helpful to have a general view of how a blockchain should be implemented. To filter the search of a good prototype, we have opted for Bitcoin and Ethereum. They are very different between them but they have what we need.

6.2.6 Algorithm implementation

The core part of the project is the Algorithm implementation. After the hard work involved on understanding each part of the Blockchain, it is time to shape our thoughts in the code.

First of all, we are going to create a repository where we will upload our code. It will be created in a way where each member of the team, can separate its own modules from the rest of the code. This allows the developers to implement their libraries and frameworks independently of the other parts.

Once we set up the repository, we will start developing our project modules in consonance with the team members.

It is important to stress that every week, all the members of the project will meet once a week in order to update all the parts and coordinate the development. This part is crucial, as it requires synchronization as the development goes further. This is so because all the parts are correlated, and can not be developed without all the information related to them.

6.2.7 API creation

In order to get all the project running, we need to group together all the modules and make them work together. This is one of the most difficult parts of the whole project, as it needs each developed part to work as expected.

To achieve the objective of good performance and communication between modules, we need to create an API. API stands for Application Programming Interface and it is a set of commands, functions and protocols that are used by developers to communicate between different modules in a program. In our case we have the Database module, the LISP module, the P2P protocol and the Consensus Algorithm. All of them should work using an API created by the developers.

Before creating the API, the team members need to meet up to define what are going to be necessary between the modules to communicate and pass the information through them. I should emphasize this stage, as everything depends on it.

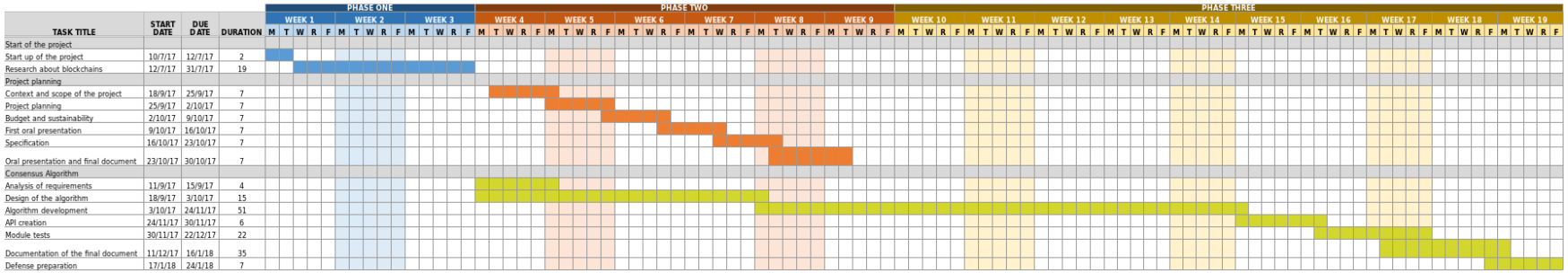
6.2.8 Action Plan

As explained in the methodology part, on the first delivery, we are going to use Agile as a development method so it will allow us to adapt the timings in a proper way if something requires more time than expected. Moreover, in the meetings, our Director and Co-director will check all the timings in order to decide whether we are in time or not.

If we see that we are in time, everything is good, so we will not take any additional action. If wee see that we are not in time, we will try to find which part is taking this extra time with the objective of reducing it to stop the lose of time. Moreover, if the action that is taken this extra time can be avoided, we will do it if necessary.
For this purpose, we will have a meeting each week, where we will analyze the situation and take actions if required.

6.2.9 Gantt Chart

In this section we will see two Gantt charts. The first one is the expected Gantt before the development of the entire project. The second one is the real and final chart.
As we can see there are a some differences during the Consensus algorithm development stage. At the beginning we thought that the most time consuming task would be the implementation of the algorithm into code. Nevertheless, in the end we have seen that by far, the most consuming task has been the research and design of the algorithm as we have changed our mind a lot of times, so it has been reflected in the time consumed.



TASK TITLE	START DATE	DUE DATE	DURATION
Start of the project			
Start up of the project	10/7/17	12/7/17	2
Research about blockchains	12/7/17	31/7/17	19
Project planning			
Context and scope of the project	18/9/17	25/9/17	7
Project planning	25/9/17	2/10/17	7
Budget and sustainability	2/10/17	9/10/17	7
First oral presentation	9/10/17	16/10/17	7
Specification	16/10/17	23/10/17	7
Oral presentation and final document	23/10/17	30/10/17	7
Consensus Algorithm			
Analysis of requirements	11/9/17	15/9/17	4
Design of the algorithm	18/9/17	3/10/17	15
Algorithm development	3/10/17	24/11/17	51
API creation	24/11/17	30/11/17	6
Module tests	30/11/17	22/12/17	22
Documentation of the final document	11/12/17	16/1/18	35
Defense preparation	17/1/18	24/1/18	7

Figure 4: Expected Gantt chart

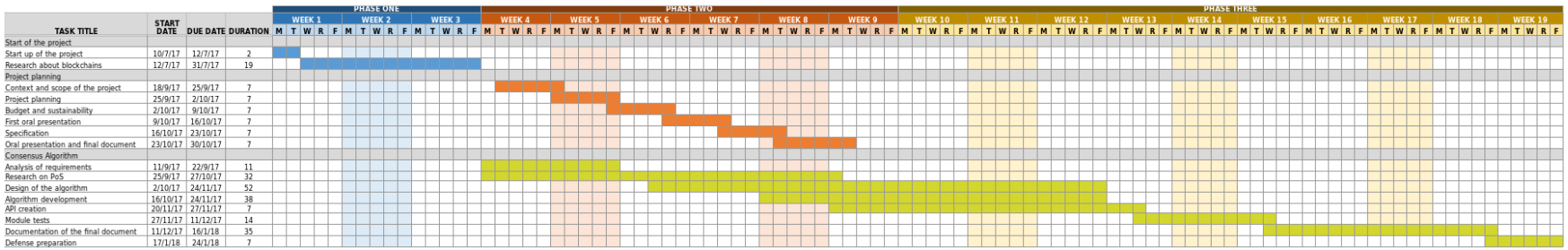


Figure 5: Real and final Gantt chart

6.3 Budget

In this part of the document we will cover the budget of the project. It contains a wide description of the estimated cost of the project, containing human and material costs.

6.3.1 Budget estimation

As for the development of the project we need software, hardware and human resources, we will divide it in three well defined sections. The hardware part, to determine the costs of the material needed to develop and test the project, the software section, to show the cost of used programs and the human cost, which contains a cost summary to know what is the total cost for a human to develop this project. At the end of the three sections, we will sum everything to have a total estimation of the cost.

6.3.1.1 Hardware resources First of all we will see the cost of the hardware resources. Table 1 contains the costs of the hardware that we are going to use in the development of the project and during the final test.

The Workstation and the Asus screen will be used to develop during the implementation time planning described in previous deliverables. This hardware will be used until the end of the project.

Two months or so before of the deadline, we will be testing the project in conjunction with the other members of the team, for that we will be using several Linux Virtual machines to analyze how the project fits into the expected result.

Product	Price	Useful life	Amortization
Workstation i5-7600 8GB 500G	1051.84 €	5 years	70.12 €
ASUS VX248H 24"	178.00 €	5 years	11.86 €
Linux Test VMs	40.00 €	2 months	0.00 €
Total	**1269.84 €**		**81.98 €**

Table 1: Hardware resources cost

6.3.1.2 Software resources Table 2 summarizes the costs of the software that we are going to need to develop our project. As it can be appreciated, the cost is zero since we will be using free software to develop everything.

As mentioned above, the software resources needed to develop the project are going to be the following:

Product	Price	Useful life	Amortization
Debian OS	0.00 €	Unlimited	0.00 €
SublimeText 2	0.00 €	Unlimited	0.00 €
LaTeX	0.00 €	Unlimited	0.00 €
Github public	0.00 €	Unlimited	0.00 €
Beta Network	0.00 €	Unlimited	0.00 €
Total	**0.00 €**		**0.00 €**

Table 2: Software resources cost

6.3.1.3 Human resources This project is going to be developed by only one person. So at the same time, the developer should do the work of a Project Manager and the work of a software developer engineer.

In order to count the human resources, we have made some assumptions. At Table 3, we can see the average pay for the roles used in the project.

Finally, at Table 4 we can see the costs of each task of the project, in terms of the human resources needed.

Role	Price per hour
Project Manager	20.00 €
Software Developer Engineer	10.00 €

Table 3: Human resources by role

Task	Project Manager	Software Developer Engineer	Cost
Project planning	55h	0h	1100.00 €
Project Analysis	25h	0h	500.00 €
Project design	0h	25h	250.00 €
Algorithm implementation	0h	110h	1100.00 €
API Creation	0h	30h	300.00 €
Testing	0h	30h	300.00 €
Measurement	25	0h	500.00 €
Documentation	60	0h	1200.00 €
Total	**140h**	**195h**	**4750.00 €**

Table 4: Human resources by task

6.3.1.4 Other costs The transport will be necessary as we will need to move up to the university in order to develop and share the progress with the other team mates. For this we will be using Bicing. The annual cost for this is 47.16 €. Divided by 4 (months doing the TFG) we have the total cost for transport as 15.72 €.

As we will be doing the project inside the university, we will take as costs the Internet connection provided by the UPC, the power consumption of the personal computer and the management cost to manage the room needed for the interns.

Internet connection is necessary in order to communicate with team members, research of the project and to upload the code to Github. As the interns room has many tables and many computers is fair to assume that we need a good Internet connection. So probably it will go around 30.00€/month, multiplied by four we have a total expense for Internet connection of 120.00€.

As another indirect cost we have the power consumption used by the computer and the room electricity. Moreover, at the end of the project we will have to test the final environment, so several Virtual machines will be used. So we can estimate the power consumption of the personal computer as 150W when working. Having a total number of hours of 360, we have a total power consumption of 54 kWh. During the final stage of the TFG we will use around of 3 or 4 virtual machines, all of them running in one or two computers, having a power consumption of 150W each one, we have a total consumption of 300W. If the number of hours dedicated to testing is 30 we have a total of 9kWh consumed by the virtual machines. In Barcelona, the price per kWh is 0.14€, so our project will spend 8.82€ in energy.

In order to have a computer and a table to develop the project, DAC department has lent to interns a place in the interns room. The cost of the manager person that manages this kind of situation can be 1400€/month, per 4 months we have 5600€. If there are twenty interns in the room the expense can be 5600/20, which is a total of 280€.

We can see the total in the following table:

Concept	Cost
Transport	15.72€
Internet	120.00€
Power consumption	8.82€
Room management	280.00€
Total	**424.54€**

Table 5: Other costs

6.3.2 Budget control

Due to some modifications during the development of the project, there is the possibility of reduce or augment the final cost of the entire TFG. In order to be as much closer to the final cost we should have a budget control.

The worst situation can happen if we do not finish the project in time. For this we will have an increment of the cost of around 1000.00 € per delayed month. So it must be a requirement to finish the project in time.

We have developed a contingency plan in order to have some funds saved in the case of a delayed project. We reserved a month in euros in order to be cautious for the case of a delayed project.

6.3.3 Total budget

Using the data shown in the tables 1, 2, 4 and 5 we can create the table 6 to show the total cost of the project, adding to it the contingency costs.

Concept	Cost
Hardware	81.98 €
Software	0.00 €
Human	4750.00 €
Other	424.54 €
Contingency	1000.00 €
Total	**6256.52 €**

Table 6: Total costs

6.4 Sustainability

6.4.1 Economic sustainability

Taking into account the amount of hours needed and the number of computers dedicated to develop the project, is difficult to do a similar project with a lower general cost. It is difficult to say beforehand if we will have the time to finish the TFG according with the deadline, but this is the objective, so no extra costs will be needed in theory. Finally we've arrived in time.

In order to reduce the cost of the project, we have used free software tools for the development of it. Moreover, with this kind of actions we are supporting the free software community, which is very important for the development of the software world.

This project is going to be awarded an 8 in the economical field. Its price is reasonable, we reduced the costs of the hardware and software parts.

6.4.2 Social sustainability

The aim of this project is to decentralize the assignment of Internet IP addresses in order to make the Internet more free and more accessible to people. With this project and the final arrival of the IPv6 protocol, we could dream in a world where each entity, person or whatever can have its own public IP without the need of third parties.

The only entity affected by our project is the IANA institution. They have the nowadays power to determine who have which IP address, so they will want to keep their power.

This project is going to be awarded a 9 in the social impact to our society, as it is open source and in the end, will help entities and users to have more security regarding the Internet.

6.4.3 Environmental sustainability

As we have seen in the costs of the project, we will use a computer and some virtual machines to develop it. So the total estimation is 63kW consumed for the implementation of the Consensus Algorithm part of the Blockchain.

Moreover, we will be sharing the interns room with other people, so it will reduce the environmental impact caused by the project.

As mentioned above, in the transport costs part, we will be using Bicing in order to reduce the CO2 emission.

This project is going to be awarded a 10 in the environmental sustainability.

6.5 Specific module TI

6.5.1 Specification of requirements

The main goal of the project is to make a Blockchain prototype for a network where the delegation, allocation and binding of IP addresses is decentralized and totally secure. As the project is difficult and extensive we've opted to divide it in four parts: **Peer to peer protocol** Protocol used in order to communicate the multiple nodes in the network. **Database definition** Database structure in which all the data from the Blockchain will be stored. **ODL** Part for the mapping and delegation resolution **Consensus Algorithm(Focus of this document)** Algorithm used to decide, depending on the inputs, who signs a block.

As each part is big enough to make a TFG on it, we've decided to implement a Consensus Algorithm for the Blockchain. So this will be our project inside the

other project. To summarize, a consensus algorithm should be able to decide, depending on various variables, who signs the block at some point of time.

Securing the Internet with Bitcoin and the Blockchain project, will be developed by the CBA research group of the Universitat Politècninca de Catalunya. Most of the topics covered by this research group are strongly related with broadband networks and broadband services and applications: switching, protocol development and performance, traffic modeling, network resource management policies and bandwidth allocation, traffic and congestion control, routing strategies, internetworking, network management, IP and optical networks convergence, quality of service management and transport protocols, as it said in its own website: *https://cba.upc.edu/*

My project, the consensus algorithm part, is based on a bigger project that starts from zero and as explained above, its main focus is to make a blockchain prototype for a network where the delegation, allocation and binding of IP addresses is decentralized and totally secure. This is important, as it brings Internet more credibility.

6.5.2 Architectural design

Regarding the whole project, the architecture will be as follows: - Core of the program: Consensus algorithm and validation - Storage: Transaction database with Patricia tree structure. This structure is very similar to the Ethereum[7] one. We've chosen this as it is very well tested and it is very powerful in terms of performance. - Communication: To send messages between peers, we'll be using a peer to peer protocol provided by a python library. For future work we can use more appropriated algorithms for the P2P protocol, but due to time restrictions we'll use just a python library. - Networking: We'll be using LISP[5] as explained in the first deliverable.

As known, our project focuses on the Consensus Algorithm part, so the main structure for this part is first, find a random known number, second, using this number select a block signer and finally, validate the block signed by the signer. In order to find a truly random number, we will be using the Hash of Ethereum[7] blocks as a source of randomness. This will put a lot of entropy in the search of a good random source. Moreover, to combine the Hash of the Ethereum blocks with something more, we'll use the Hash of a NIST beacon[14], which its only objective is to output every minute a random Hash.

6.5.3 Implementation

In order to implement the Consensus algorithm part of the project, we will be using different Hardware and Software tools.

As Software tools we have:

- Debian OS: The main operative system that we'll be using during the implementation is Debian OS, a Linux distribution. We've opted for this OS because of its free cost and its good performance in terms of consumption and friendly usage.
- SublimeText 2: As a text editor for the programming of the project, SublimeText 2 will be our choice. It offers a free to use version and it is very complete regarding Python Language.
- Github public: As the main public repository we chose Github, as it is very well known and very easy to use.
- Beta network: For the testing of the final complete version, we will be using the beta network of the university. There we'll be able to test performance and usability of our implementation.
- Virtual Machines: Once we achieve the final of the project in terms of implementation, we'll be executing the code in different Linux virtual machines.

As Hardware tools we have:

- Workstation: The computer that we'll be using is a workstation lent by the university. It is an HP computer with good performance.
- Screen: The screen used to develop the implementation will be an Asus monitor of 24".

6.5.4 Risk management

As we will be using open source software for the development of the project, we have the risk of being deprecated in the future. If the developers do not continue with the development we can be affected. This possibility is not so real as there are a lot of contributors working in the software that we've chosen for the implementation.

It is true that licensed software can be a good choice if we need support. But as we know the tools and there a lot of documentation, we won't have any problem for the development of the project with the software chosen.

As a risk we also have the management of more than one virtual machines. It is not trivial to manage them, as they are hardware simulated by software, so sometimes it can be tricky to make everything work as expected.

6.5.5 Relation of the project with technical competences of the specialization

1. CTI1.4 (To select, design, deploy, integrate, evaluate, build, manage, exploit and maintain the hardware, software and network technologies, ac-

cording to the adequate cost and quality parameters) This competence was chosen as we'll select, design, deploy, integrate, evaluate, build, manage, exploit and maintain the hardware and software of a network technology. This network technology is the Blockchain, and we'll create a new one with what this entails.

2. CTI2.3 (To demonstrate comprehension, apply and manage the reliability and security of the computer systems (CEI C6)) This competence was chosen because the main objective of the project is to secure the Internet in the end, so we need to understand and manage the security that is around the network system.

3. CTI3.4 (To design communications software.) Behind the Blockchain there is the communication protocol, the P2P mentioned above. This is a crucial part, so in order to develop the rest of the project, we need to fully understand how it works.

7 Conclusions

Regarding the Consensus Algorithm module, all the objectives have been met except one. The module is not capable of protecting the network against a sub network attack (explained in **Section 5.6**). At this moment this protection is not necessary as the prototype is still in development but, in the future, if the prototype goes in production phase, this must be solved. Despite this unmet objective, the algorithm finds the same signer among all the nodes in the network and it does it completely random, so the next signer can not be predicted beforehand. Also the block creation rate is fast compared to other important blockchains such as Bitcoin.
To sum up, the algorithm behaves as expected and provides a secure signer election that maintains the blockchain synchronized in terms of block creation.

In relation to the prototype, were all the modules should work in consonance at the same time, the result is not the expected. At the beginning, when the nodes of the network were in the same location, i.e. Barcelona, the prototype behaves as expected and the blockchain was able to create blocks in a reasonable time. However, when we tested the prototype with nodes that are geographically distributed, the result was not optimum.
The nodes where distributed on North California, North Virginia, Frankfurt, Ireland, Sydney, Tokyo and Barcelona, were the bootstrap node is located.
The main idea of the tests is to delegate an initial amount of /8 addresses to the nodes. Then, among the nodes, /16 addresses should be delegated. But the result of the test is not satisfactory as there is a problem of clock synchronization among the nodes.

In conclusion, there are some aspects that can be improved in the prototype in order to work as expected, but in general all the modules have worked properly. So if the synchronization problem is solved, the prototype will demonstrate to be a good alternative to the current security management of the Internet.

7.1 Results

During the preparation of the final results, the Ethereum API that we have used in order to retrieve the information from the Ethereum blockchain, stopped working. In fact, the information taken from the API was chaotic, and depending on the IP address that makes the call, the result was different. This behavior confuses the team as we saw that the blockchain was not working properly. At this point we opted to leave aside the Ethereum block hash information and work only with the NIST beacon.
When using only the information from the NIST beacon we expected a block creation rate of sixty seconds approximately, as this beacon changes every minute. So the tests consisted on measuring the time that takes for the consensus algorithm in order to find a new signer.

As can be seen in Figure 6, the average in one hundred and ten samples is very close to sixty seconds, so the consensus algorithm is behaving as expected with this configuration. Despite this good results, the main idea was to have a rate block creation of fifteen or twenty seconds, which could not be achieved using only NIST. Nevertheless, this rate is ten times faster than Bitcoin's blockchain, so it can be considered as a good rate.

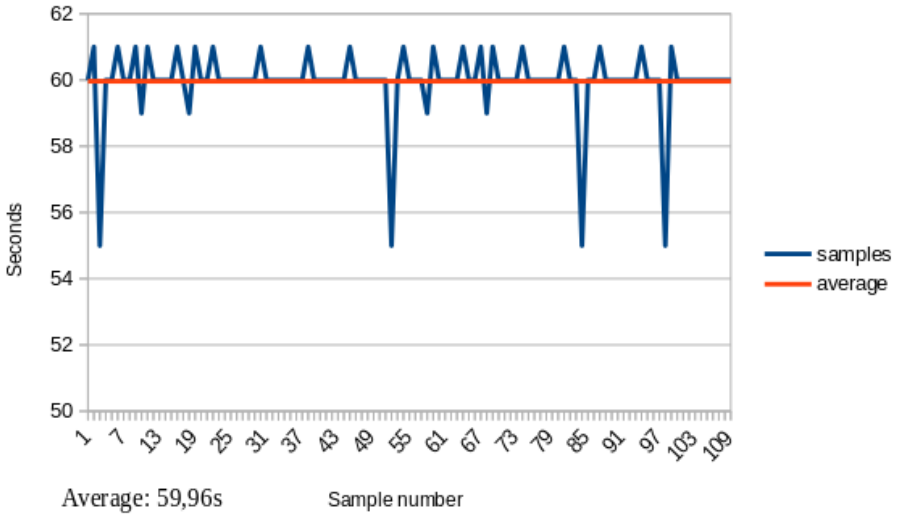


Figure 6: Sample using only NIST beacon

During the final stage, and so close to the deadline, we realized that the Ethereum API started working again, so we did the same tests as for the previous commented result. The graph can be seen in Figure 7 and it shows very interesting numbers.
As we can see, the average of seconds to find a signer is thirty tree seconds. A half of just using only NIST as a random source. Although there are blocks created in the expected time, i.e. fifteen or twenty seconds, in some situations the time goes up to eighty or seventy seconds. This is caused for a high delay between the API server and the real Ethereum chain. This can be solved if instead of using an external API, every node of the prototype runs an instance of the entire Ethereum blockchain. Nowadays this is not feasible because of the long time of bootstrapping at the beginning and for the amount of memory needed to run an Ethereum node.

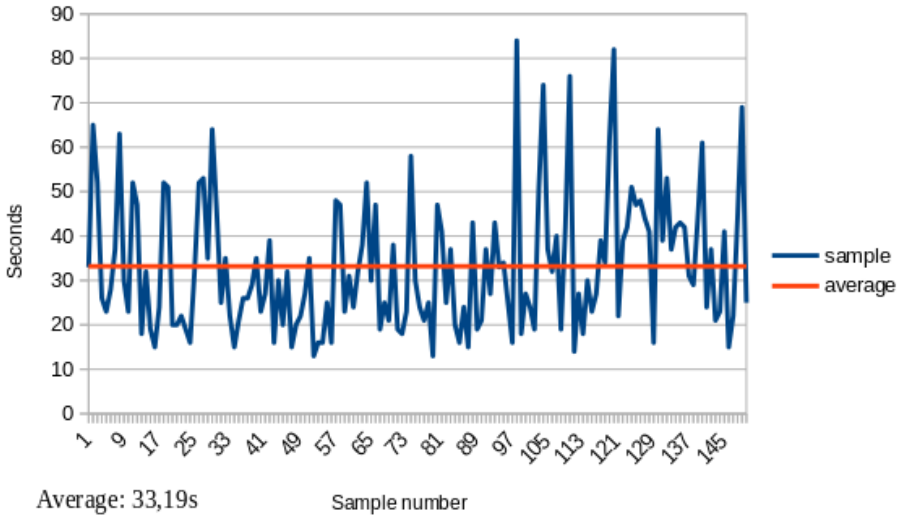


Figure 7: Sample using Ethereum and NIST

7.2 Future work

7.2.1 Blockchain fork handling

In the **Section 5.6** of this document, fork handling has been addressed as a drawback in the implementation of our algorithm. There are some specific situations where the chain has problems to support forks. In particular, when a malicious node has to sign the block, the chain can be affected by the creation of sub networks.
To avoid this problem, more work should be done on the consensus algorithm module and in the P2P protocol. The main idea to solve this is to verify the correctness of the block when receiving a new one. For example it can be done by asking other peers randomly, and listen to their response. If the hash of the new block arrived is the same that arrives to other peers, verification goes successful, so the block can be added to the chain.
By doing this, very small sub networks can be created, but they will stop working when realize that they are alone, i.e. when no signers appear.

The best solution in terms of fork handling is to use an algorithm that protects the chain of it. For example, in this document, we have seen Ouroboros and Algorand. Which can be good alternatives as consensus algorithms. However, it will take a lot of time to integrate in in the prototype.

7.2.2 Improve block creation speed

Theoretically, the block creation speed of our prototype should be on average five-teen seconds. It follows the speed creation of Ethereum, so it should go at the same rhythm.
Besides five-teen seconds of block creation is fast compared to other blockchains, it is important to slow down the time to have more transactions per second.

The only way of improving the speed is by removing the dependency on Ethereum and NIST, i.e. changing completely the consensus algorithm.

7.2.3 Tangle instead of Blockchain

Tangle technology[4] is gaining a lot of popularity thanks to IOTA[3] cryptocurrency. IOTA stands for the machine-to-machine currency of the future, being used for the bast majority of IOT devices.

The Tangle, differing from blockchains, uses a directed acyclic graph (DAG) as a data structure which aims to have an infinite scalability thanks to its disrupting consensus algorithm. Tangle structure can be seen in Figure 8.
When a new user issues a transaction, it should confirm and verify two previous non confirmed transactions, also called tips. This verification is the PoW necessary for the system in order to confirm a transaction. This operation is similar to the PoW in other blockchains such as Bitcoin, but as the software should be run in IOT devices in the future, the computational effort to verify the transaction is minimum. In fact it needs 3^8 operations to resolve the nonce. This behavior is supposed to allow the high scalability that the Tangle paper claims. Theoretically, as more transactions are sent to the network, faster it becomes, because the need of verify two previous transactions.

By the write of this thesis, IOTA is being tested by thousands of users and for the moment it is working as expected. So, as far as we understand, our prototype could work better and faster with the Tangle, as in the future IP delegation and allocation will be needed constantly, due to the amount of new devices that will appear thanks to the IOT massive deployment.

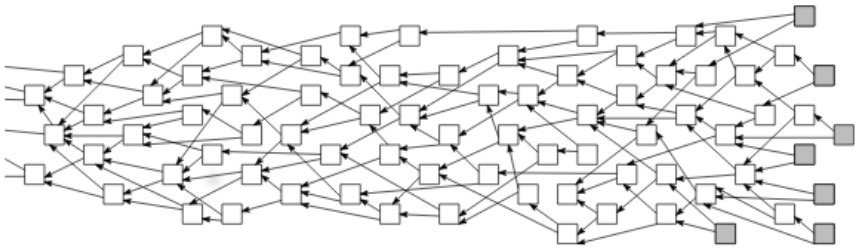


Figure 8: Tangle flow of transactions (source [4])

References

[1] S. Nakamoto. *Bitcoin: A Peer-to-Peer Electronic Cash System*. URL: https://bitcoin.org/bitcoin.pdf.

[2] M. Ali et al. *Blockstack : A Global Naming and Storage System Secured by Blockchains*. URL: https://blockstack.org.

[3] *Iota: a cryptocurrency for Internet-of-Things*. URL: https://iota.org/.

[4] *Tangle*. URL: https://iota.org/IOTA_Whitepaper.pdf.

[5] *Locator/ID Separation Protocol*. URL: https://tools.ietf.org/pdf/draft-ietf-lisp-introduction-13.pdf.

[6] J. Paillisse. *An analysis of the applicability of blockchain to secure IP addresses allocation, delegation and bindings*. URL: https://tools.ietf.org/html/draft-paillisse-sidrops-blockchain-00.

[7] *The Ethereum project*. URL: https://www.ethereum.org/.

[8] et al. Aggelos Kiayias. *Ouroboros: A Provably Secure Proof-of-Stake Blockchain Protocol*. URL: https://eprint.iacr.org/2016/889.pdf.

[9] et al. Silvio Micali. *ALGORAND: The Efficient and Democratic Ledger*. URL: https://arxiv.org/pdf/1607.01341.pdf.

[10] *Namecoin*. URL: https://namecoin.org/.

[11] Ittay Eyal and Emin Gün Sirer. *Majority is not Enough: Bitcoin Mining is Vulnerable*. URL: https://arxiv.org/abs/1311.0243.

[12] *IOHK | Cardano whiteboard; Ouroboros, with Prof. Aggelos Kiayias, Chief Scientist*. URL: https://www.youtube.com/watch?v=nB6eDbnkAk8.

[13] Jeremy Clark Joseph Bonneau and Steven Goldfeder. *On Bitcoin as a public randomness source*. URL: https://eprint.iacr.org/2015/1015.pdf.

[14] *NIST beacon*. URL: https://www.nist.gov/programs-projects/nist-randomness-beacon.

[15] *Agile product development*. URL: https://www.prometheusresearch.com/embracing-agile-product-development/.