uc3m | Universidad Carlos III de Madrid

University Degree in Telecommunication Technologies Engineering

Academic Year (2021-2022)

Bachelor Thesis

Indeliable Evidences

Timestamping service over Bitcoin blockchain and prove of ownership by ERC-721 standard and Ethereum database implementation

---

Jose María de la Cruz Sánchez

Daniel Díaz Sánchez

Madrid, 2022

ABSTRACT

Most reports about Blockchain focus on explaining why Blockchain technology is important and how it will change the world as we know it today in a very general way or focused in a business approach. The thesis will concretely explain how the Bitcoin and Ethereum blockchain network work and how they can be used to provide certain services allowing us to reach a completely secure world without having to rely on third parties.

The thesis proposes a service design that operates using a protocol, the OpenTimestamps protocol, built over the Bitcoin blockchain in order to timestamp any file and its contents in a totally reliable and incorruptible way without the need to rely on third parties. A parallel process has also been implemented by the use of smart contract on the Ethreum blockchain in order to provide the users with ownership of their timestamps.

In order to perform with the project, it will be first explained what is Blockchain, how the Bitcoin and Ethereum blockchains work for the purpose of understanding how is the OpenTimestamps protocol developed over the Bitcoin Blockchain and how the parallel process in designed and executed.

It will also be analysed how the OpenTimestamps (OTS), an open-source protocol that provides a standard format for blockchain timestamping that works over Bitcoin Blockchain. The thesis also presents a design and a prototype implementation of a service that implements the OpenTimestamps protocol which allows the user to timestamp file contents and that provides the possibility of validating the existence at any point of time of these files in a totally reliable way due to the characteristics of the technology on which it is built over, the Bitcoin blockchain. A quick technicall overview about tokens in the Ethereum blockchain is also given for the reader to understand the importance of ownership and how it can be truly achieve in a decentralized service by the generation of ERC-721 tokens which could act as an evidence or proof.

Keywords: Blockchain, Bitcoin, Ethereum, OpenTimestamps, smart contracts, ERC-721, tokens, ownership, distributed systems, cryptography, merkle trees, file timestamping, nodes, consensus mechanism.

DEDICATION

Me gustaría dedicarle esta tesis a mi familia, a mi hermana y mis padres por el apoyo incondicional que he recibido de su parte durante toda mi vida y por poner todo su esfuerzo en las oportunidades que me han brindado durante esta, ya que sin ellos no hubiera sido posible recorrer el camino que me ha llevado hasta aquí y lo que me queda por conseguir que estoy seguro les hará sentir muy orgullosos.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

1. INTRODUCTION

Blockchain technology is becoming increasingly popular these days, most people immediately associate it with cryptocurrencies but the potential applications of blockchain technology go beyond money transactions in a decentralized system. Blockchain provides the opportunity of developing countless services that can be built on the top of this technology to improve our lives in the area of decentralization. The number of people who don't trust institutions and centralized companies is exponentially increasing on those days and decentralized services are the solution for that. In ownership issues users and stakeholders also have to rely on centralized third parties and assume the risk that these third parties will behave in a non-malicious or corrupt manner as they have done so many times, therefore the creation of decentralized applications and protocols in terms of ownership is hugely required.

1.1. Motivation

The relationship between organizations and people has become increasingly problematic because of untrustworthiness episodes during the past decade. From 2008 onwards, when the financial crisis took place, financial institutions, mainly banks, have been atribuitted the failure of trust by the people [1]. Episodes of abuse of information and use of technology for monitoring people using XKeyscore that was revealed by Edward Snowden [2],[3]. Large centralized technology companies, such as Google and Facebook, have also admitted data breach that had affected millions of users[4]

Proving that something existed, such as a file on the internet, or that something took place at some particular point in history, in a deterministically way, is a very complex task. Since the beginning of the history, the task of proving that something existed was delegated to a forced trusting on a third party or centralized system. Trusting a third party or a centralized system has several implications.

A cryptographic solution for the game theory problem known as "The Byzantine Generals Problem" [5], where the problem of reaching consensus in a decentralized system withouth the need of trust in a third party is presented, was proposed in 2008. Satoshi Nakamoto, who released a white paper called "Bitcoin: A Peer-to-Peer Electronic Cash System" [6]. The potential applications of blockchain technology go beyond money transactions in a decentralized system. Blockchain technology provides us the opportunity of building applications and offer powerful services in terms of security, transparency, scalability, reliability and tracealibily.

Being able to prove that a particular document has existed at some point in time on the internet in an unquestionable way even if it has been deleted nowadays is a very compli-

cated task when we talk in terms of centralized databases as these can be manipulated at any time but by making use of the tools and benefits provided by Blockchain technology it is possible to create protocols and services that take evidence of the existence of any document in a decentralized, immutable and incorruptible way.

1.2. Objetives of the thesis

The main and general goal of the thesis is to propose a design and a prototype implementation of a decentralized service which will allow the user to timestamp and the content of the files of his choice, considering files from different sources, and generate an evidence of ownership. With the use of a cryptography-backed protocol implementation and design of the Bitcoin and Ethereum blockchain networks will make it unnecessary to rely on third parties. To achieve this main purpose, the designed and implemented service in this thesis must accomplish the following specific goals:

- Generate timestamps from the content of objects in an almost infinitely scalable method.
- Creation of a timestamp verification method by means of a previously generated hash.
- Creation of an ownership protocol that links the timestamps generated by a user to his own Ethereum address to provide evidence that he was the originator of the timestamp.
- In real time generation of ERC-721 token for the timestamp evidence generated by the user.

The implemented service will not only allow to timestamp file contents but also to verify the existence in any point of time of a previously timestamped file. A parallel process implemented on the Ethereum blockchain by the usage of smart contracts and the ERC-721 standar will be also implemented to provide ownership to the user of its own performed timestamps.

To understand the technology at the core of the implementation of the proposed service and how it can provide full security over the timestamps of the file contents, it will be first analyze in depth and in a technical context the Blockchain network systems, more specifically, the performance and different key aspects of the Bitcoin and Ethereum blockchains from its origin to the most technical aspects such as the architecture of its network and distribution of the nodes participating in it, the structure of the blocks that form its chain, the cryptographic methods used and its consensus mechanisms. It will also be analyzed the operation of the OpenTimestamp protocol that will provide the timestamping of the files and that is built over the Bitcoin network. A technicall analysis about tokens on the Ethereum blockchain and their properties is also given.

1.3. Outline of the thesis

The thesis is structured according to the subsequent points:

- **Chapter 1:** Main and specific objetives of the thesis and motivations are presented.
- **Chapter 2:** State of the art. A presentation about technical analysis about distributed systems, blockchain networks, Bitcoin blockchain and Ethereum blockchain in order to clearly understand the proposed service by the reader.
- **Chapter 3:** The protocol on wich the thesis proposed service is based is presented. Presentation and technical analysis on the performance, design and scalability options of the used protocol.
- **Chapter 4:** The thesis proposed service is presented. An in-depth presentation of the objective of the proposed service and a functional description is given.
- **Chapter 5:** Implementation of the previously proposed service and a service testing is given.
- **Chapter 6:** Conclusions of the thesis and possible future work is given in this chapter.
- **Chapter 7:** Socioeconomic and project management are presented. An explanation on social and economical points related with the thesis development is also given.

2. STATE OF THE ART

The main purpose of this chapter is to provide the concerned person with trivial concepts about blockchain technology and to explain in depth how Bitcoin blockchain works to enable the reader understand the analysis of OpenTimestamps and the rest of the thesis.

2.1. Introduction

2.2. Distributed systems

2.2.1. Definition

Blockchain core works as a distributed system, specifically as a decentralized distributed system.

"Distributed systems are a computing paradigm whereby two or more nodes work with each other in a coordinated fashion in order to achieve a common outcome and it's modeled in asuch a way that end users see it as a single logical platform". [7]

In a distributed system a node is defined as each individual contributor to the system. Nodes are connected to other nodes and they are able of communicate with each other. Nodes can be both, software processes or hardware devices. Nodes in an distributed system can behave in an arbitrary manner. Nodes can behave in a honest, faulty or malicious manner. Inappropiate behaviour of a node can be cause by disability of the node's function, a node can be switched off but it can also behave in a malicious way intentionally because of someone's intention to hack into the system. This nodes are also referred to as *Bizantines*

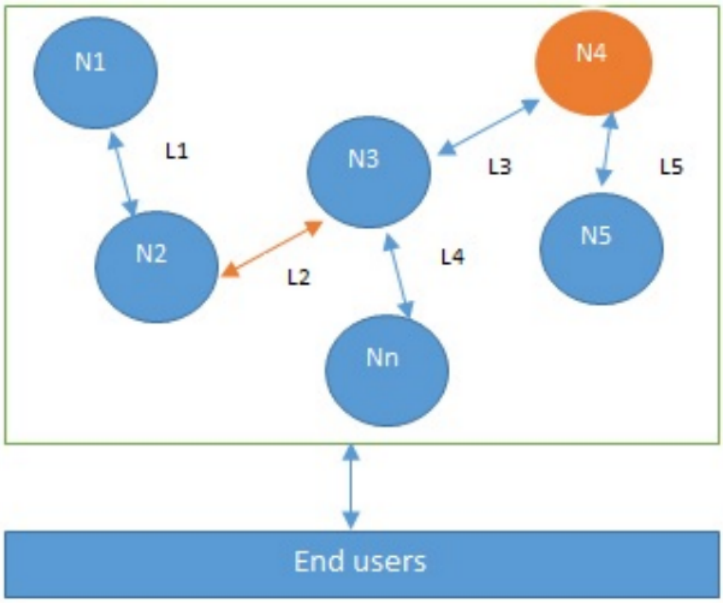


Fig. 2.1. DISTRIBUTED SYSTEM DESIGN [7]

The distributed network should work in order to achive the purpose or goal it is designed for even if a link on the network becomes broken or a single node or a group of them becomes maliciously operational. Several algorithms have been designed and proposed to manage node's faults or malicious behaviour during the time, this thesis will cover consensus algorithms in the coming chapters.

2.2.2. Features

Distributed systems have to fulfill two principal requirements in order to be considered as such:

2.2.2.1 Collection of autonomous computing elements

Nodes types can range from small plug computers to high-performance computers in modern distributed systems. Nodes can perform actions completely independent from other members in the distributed system but they are actually programmed to achieve common goals. Although the nodes can act independently, they are programmed to exchange messages with the other members of the system.

The notion of time of each node that takes part of the same distributed system does not match with the other members of the network. The design of an implement synchronization algorithms for the nodes to avoid uncoordination acting is necessary to deal with the absence of a common timing benchmark.
A distinction must be made between closed groups and open groups in distributed systems in order to qualify its group membership system. **Open group systems** allow new nodes

to join the distributed system so that the new members are able to establish a connection with the rest of members in the system interchanging messages with them, this can be seen in figure 2.2 b). In **close group systems** members in the group can communicate with each other but if a node inteds to join or leave the system a separete mechanism is need to be applied as it can be seen in figure 2.2 a).

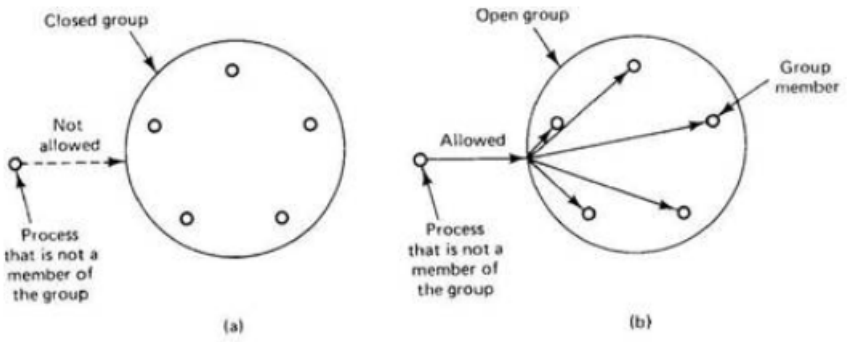


Fig. 2.2. CLOSE GROUP AND OPEN GROUP JOINING MECHANISM [7]

For an addmisision control to be successfully implemented, a node authentication method must be used, if not, scalability bottleneck can appear. Communications between nodes must be verified individually so that each of them can ensure whether they are in contact with a genuine node or a malicious member of the system.
If confidentiality is a feature on the distributed system, as system member nodes can easily communicate with non system member ones, trust issues must be handled.[8]

Distributed systems are typically organized in an overlay network design. In overlay network designs a list of processes is assigned to a node, which is normally a software process, to which it can directly send message to through TCP/IP or UDP channels.
Overlay networks can be clasiffied as **structured overlays** where nodes have perfectly defined the set of nodes to which they can communicate with as it is seen in figure 2.3, and **unstructured overlays**.[9]

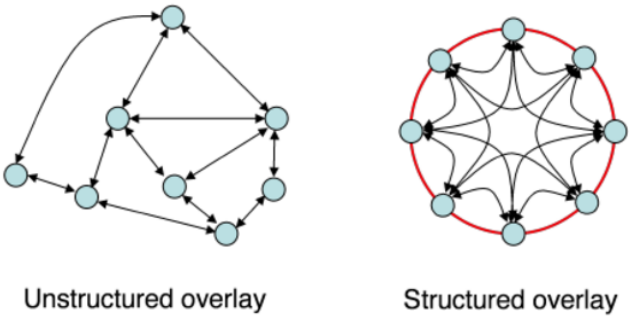


Fig. 2.3. UNSTRUCTURED AND STRUCTURED OVERLAYS [7]

Peer-to-peer (P2P) networks, such as Bitcoin Blockchain, are a well-known type of

overlay networks.

2.2.2.2 Single coherent system

Distributed systems should behave as coherent systems in order to be consdiered as so, this means that the user perceives it as a single system and not as different processes, information and management that are scattered throughout a whole computer network. **Distribution transparency** is a main goal of distributed systems to achive in order fulfill the requierement of beign percieved as a single coherent system by the final user. Distribution transparency refers to the idea of designing a distributed system where locating in a precisely manner on which workstation a task is being executed or to notice that a it was split up in two or more processes executing somewhere else becomes impossible for the end user [8].

"The distributed systems should be perceived as a single entity by the users or the application programmers rather than as a collection of autonomous systems, which are cooperating. The users should be unaware of where the services are located and also the transferring from a local machine to a remote one should also be transparent." [10].

2.3. Blockchain networks

2.3.1. History

In 1976 a paper under the name of "New Directions in Cryptography" [11] was published. This paper discussed the concept of distributed ledger from a cryptographic perspective. A first distributed system idea that aimed to create a cryptographically encrypted chain of blocks to generate digital documents tamper-proof timestamps was first presented in the year 1991 by Stuart Haber and W. Scott Stornetta [12], [13].

Hashcash was a proposal by Adam Back [14], a British cryptographer and cypherpunk in 1997 that tried to wipe out spam emails or reduce the number of them. Hashcash was a system that proposed that mails could incorporate proof that the sending computer had previously dedicated some time and resources to be able to send such mails, this cost of electricity will be minimal for people sending not many emails but whoever wanted to send millions of mailings would have to devote considerably more resources to it, thus imposing a cost [15]. This idea led to the creation of 'proof of work (PoW)' protocol used in Bitcoin blockchain.

A first attempt at developing an anonymous, distributed electronic cash system was made by Wei Dai. The concept b-money [16] was described in a white paper published in 1998 by Wei Dai which was a computer engineer by the University of Washington. Wei Dai described b-money as "a scheme for a group of untraceable digital pseudonyms to pay each other with money and to enforce contracts amongst themselves without outside

help" [17]. B-Money's paper describes some specific features of its network that match those of today's some cryptocurrencies blockchain technology. In Wei Dai's paper a computational work system was proposed in order to generate b-money and to reward the nodes that worked in the network, this would trigger a network formed by a community of independent nodes that would be in charge of verifying the transactions and remain them organized by using a bookeeping with cryptographic protocols for authenticate these transactions, in compensation for the aforementioned rewards [16]. This concept was inspired in Hashcash mechanism and it is almost identical to the proof of work protocol used in Bitcoin blockchain and others which will be detailed described in future chapters of this thesis.

Wei Dai was honored years later by attributing his name 'wei' to the smallest unit into which ether can be divided, the cryptocurrency of Vitalik Buterin designed blockchain, Ethereum [18], [19].

In 2005, a scientist on computer science specialized on cryptography, Nick Szabo, desgined a new decentralized mechanism for the build of a new digital currency, BitGold [20]. Proof-of-work was the basis of the mechanism designed by Nick and on which the foundations of his digital currency for cryptographic problem solving were built. Bit Gold was never implemented but it known as the Bitcoin precursor.

Today's most popular blockchain network, the bitcoin blockchain, was proposed in 2008 by Satoshi Nakamoto, whose real identity is still not known in the nowadays. A paper titled *"Bitcoin: A Peer-to-Peer Electronic Cash System"* [6] was published by Sathosi Nakamoto. In 2009 an implementation of the previous blockchain technology was created, Bitcoin. The double-spending attack was first solved by a decentralized currency with the creation of Bitcoin [21]. Satoshi drew inspiration from the system implemented by Hashcash, proof-of-work (PoW) and implemented it in the bitcoin network as consensus algorithm which is able to prove that a node has spent resources in the creation of a new block of the blockchain and validate it.

Over the years, new blockchain networks have been developed and blockchain technology began to be conceived as a separate concept from currency since 2014. This is the case of the most widely known network, Ethereum blockchain, which introduced the concept of *Smart Contracts* [22]. Users can engage with a smart contract by submitting transactions that execute the smart contract's declared function. Smart contracts, like conventional contracts, can set rules and have them enforced automatically through programming. [23].

New blockchain networks or decentralized applications are still being developed in the present and will be in the future but for the purpose of this thesis, the network to be analyzed will be the Bitcoin network.

Electronic cash schemes and distributed systems mechanism used to developed the previous technologies for the ideas of creation of decentralized cash was combined to create Bitcoin Network.

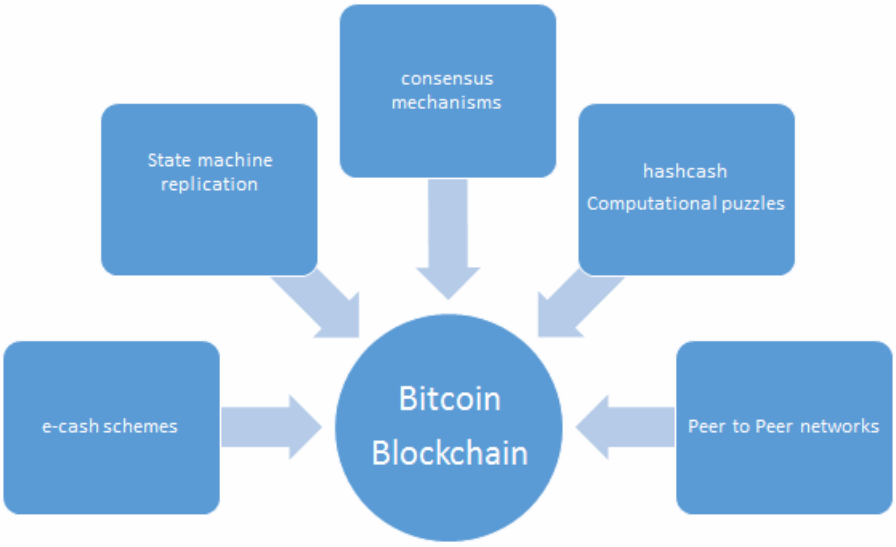


Fig. 2.4. CONTRIBUTIONS TO BITCOIN BLOCKCHAIN CREATION [7]

Among all the blockchain networks that exist at the moment, this thesis will focus on the technological analysis of the Bitcoin blockchain on which the OpenTimestamps tool is deployed, so that the reader can understand it more accurately in future episodes of the thesis.

2.3.2. Blockchain

2.3.2.1 Introduction

From a business point of view, blockchain technology could be described as a system of records of transactions of value in a peer-to-peer system, with no central authorities and intermediates. In such a symple way of explaining it is just removing the intermediate between two peers that are interchanging value from the equation, as it can be seen in figure 2.5.

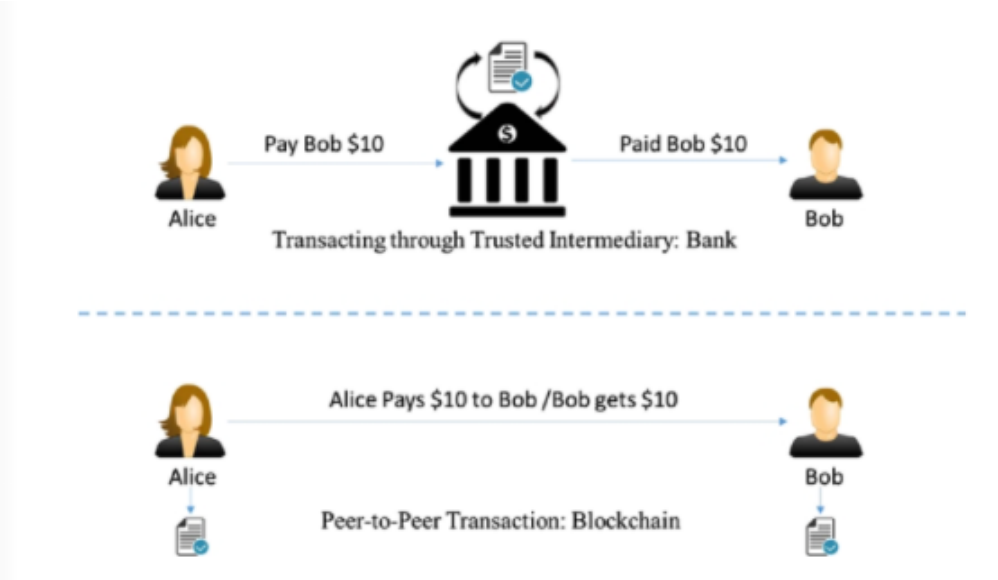


Fig. 2.5. INTERMEDIATE VS PEER-TO-PEER TRANSACTIONS [24]

In order to remove intermediates from peer-to-peer transactions trust is needed to be provided in those transactions between peers. Achieving trust from users who do not know each other is very difficult, almost impossible. Blockchain technology is the solution that provide trust in decentralized manner because of what its features imply.

Blockchain is a peer-to-peer distributed ledger, a data structure formed by blocks that are connected in a chain structure to generate a collection of records that uses cryptographic technigques to ensure its security. This ledger database mirrored in all the nodes that take parts in the network. Blockchain is an append-only ledger, immutable and it can only be updated by an agreement among peers mechanism also known as consensus. Bitcoin could be define as a distributed database of record of transactions which is validated by a network of nodes around the world.

Blockchain's core consists on a distributed system, exactly a decentralized distributed system. When a node in a blockchain network engages in malicious behavior it is easily detected by the other members of the network and ejected from it. Integrity in a blockchain network is reached by cryptography.

"Blockchain can be thought of as a layer of a distributed peer-to-peer network running on top of the Internet, as can be seen below in the diagram. It is analogous to SMTP, HTTP, or FTP running on top of TCP/IP." [7].

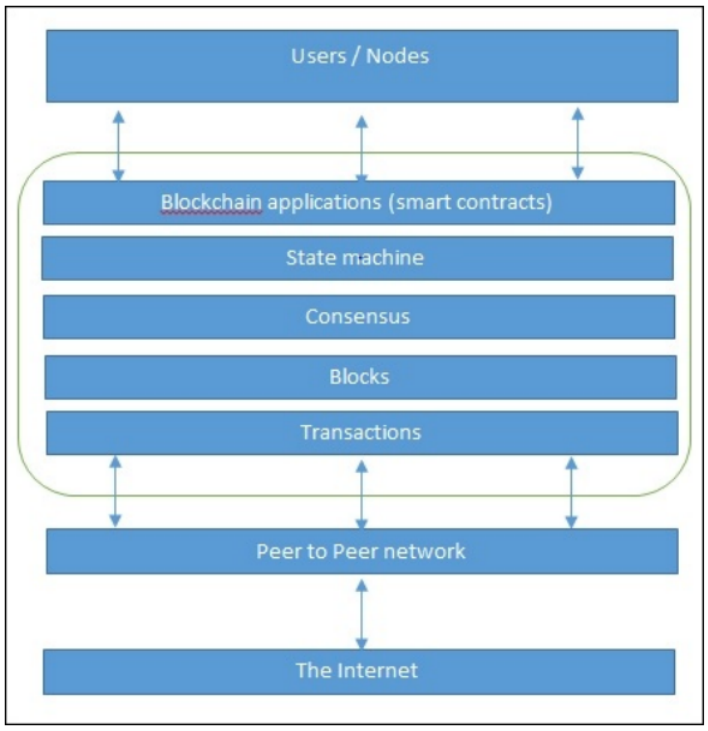


Fig. 2.6. BLOCKCHAIN NETWORK [7]

There are several blockchain networks and new ones are emerging every day, each one with its own defining characteristics but all of them sharing common fundamental principles that define them as such.

In the following sections of the thesis, the main characteristics of blockchain networks will be presented and explained in a technical format so that the reader can subsequently gain a general overview of blockchain technology in other to achive an in-depth understanding, in next chapters of the thesis, of how the network under study, Bitcoin blockchain, works.

2.3.2.2 Data structure

Blockchain networks are designed in such a way, as the name implies, as a chain of blocks which contain information about transactions and where sizes of the blocks differ according on how it was designed for the particular blockchain network we are referring to. The first block of any blockchain network is commonly known as genesis block, all blocks in blockchain networks have reference to their previous blocks forming a chain except obviously the genesis block for being the first one in the network.

Transactions can not be altered in blockchain networks because each block contanis a *hash* pointing to the previous block in the chain in its header section.

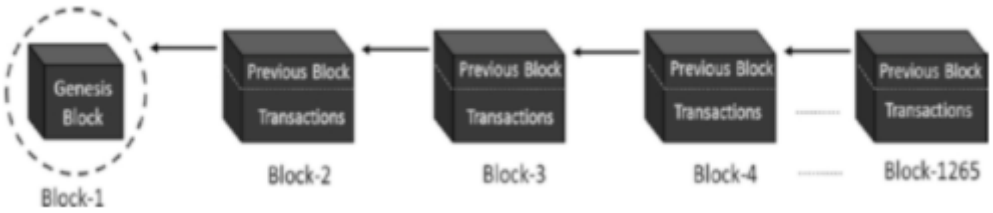


Fig. 2.7. GENESIS BLOCK AND BLOKCHAIN STRUCUTRE [24]

Blocks in a blockchain network are fundamentally data structures that are represented in a block pattern. Each blockchain network defines the structure of its blocks, including features that vary depending on the design of its network but there are some common features that are usually repeated and included in most of the blockchain networks as it is seen in figure 2.7.

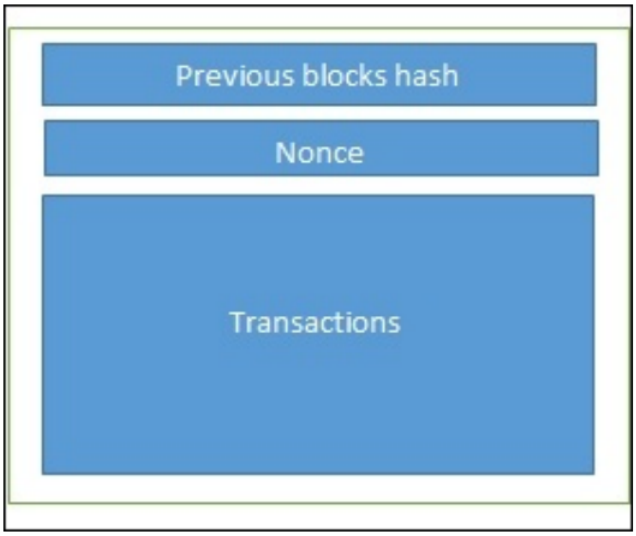


Fig. 2.8. BLOCK STRUCTURE [7]

Everynode in the blockchain has a copy of the transactions ledger which result in that it is impossible to reverse a transaction. Changes in blocks already added to the blockchain become impossible to execute because they would result into a new transaction invalidated by all the nodes in the blockchain.

2.3.2.3 Types of systems

One of the main feature of blockchain network is that they are theoretically designed to become decentralized networks. The number of nodes taking place in a blockchain network and its consensus algorithm, that we are analyzing in later sections, are directly related to the degree of decentralization a blockchain network can achieve.

It is very important to differenciate between distributed and decentralized networks.

Even wether a system is distributed if can be centralized or decentralized. In centralized distributed systems, it exists a *master node* which assigns tasks or data to the rest of the nodes as it can be seen in figure 2.9 [24].

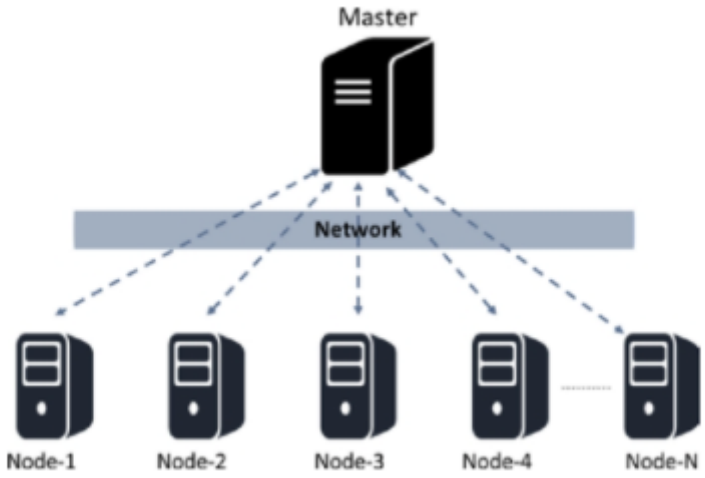


Fig. 2.9. CENTRALIZED DISTRIBUTED SYSTEM [24]

CENTRALIZED SYSTEMS

In centralized systems a central node, as its name suggests, is connected to all the other nodes in the network, thus defining the architecture of centralized systems as it can be seen in figure 2.10, which has control over the rest of nodes with all administrative authority.

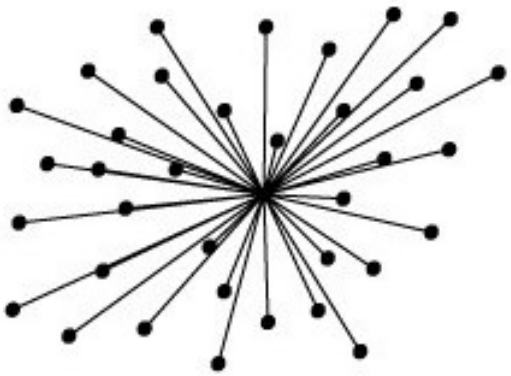


Fig. 2.10. CENTRALIZED SYSTEM

Characteristics of centralized systems

- **Common clock:** Since there is a central node to which the rest of the member nodes of the system are connected, they are synchronized with the 'clock' of the central one.

- **Unique central node:** The central node is the node in charge of coordination of the rest of nodes connected to it.
- **Failure dependency:** The rest of the nodes in the network are bound to the operation of the central node. If the central node is attacked, shut down or stops working for any other reason, all the nodes connected to it will also be attacked, and therefore the system as a whole will not be able to continue operating [24], [25].

Centralized systems are easy to design, maintain and govern but there are several limitations when it comes to this type of systems.

Limitations of centralized systems

- Scaling up vertically after a certain limit becomes imposible. After a limit, even if you increase the hardware and software capabilities of the server node, the performance will not increase appreciably leading to a cost/benefit ratio < 1.
- Traffic spikes can lead to bottlenecks. The number of open ports in the server which are listening connections from the client nodes is a finite number. When high traffic situations take places the server can essentially suffer a Denial-of-Service attack or Distributed Denial-of-Service attack (DDoS) [24], [25].

As every system, centralized systems bring with them a number of advantages and disadvantages due to their characteristics,

Advantages of centralized systems

- Physical security is easy to reach by virtualizing their location.
- Satisfying user experience.
- Updates can be excecuted quite fast.
- Easy removal of a malicious node from the network. As there is only one connection from that node to the central node, it is simply a matter of eliminating that connection. [24], [25].

Disadvantages of centralized systems

- Network connectivity is a very high attached to the network. Existance of nodes with a single connection to the central one implies the failure of the system if connection is lost.
- If a disconnection situation occurs, it would not take place gradually.

- Reduced data recovery possibilities. If a node is disconnected from the system and does not have a backup assigned to it, all data will be lost.
- Server maintenances become difficult to carry out [24], [25].

DECENTRALIZED SYSTEMS

The overall behavior of decentralized systems is the overall sum of the operations and decisions of the individual nodes as each node has a weight of authority at the same level.

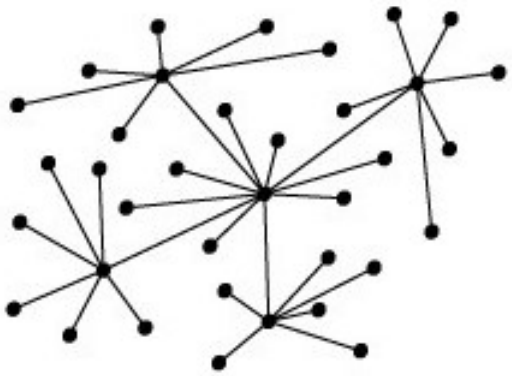


Fig. 2.11. DECENTRALIZED SYSTEM

Characteristics of decentralized systems

- **No common clock:** In contrast to centralized systems, in decentralized systems there is no common clock to synchronize all the nodes that are part of the network, rather, as they are dependent on each other, each one relies on its own internal clock.
- **Several central points** Connections can be listened from more than one central unit.
- **Failure dependency:** Failure of one node does not implies on the failure of the entire network but of a part of it [24], [25].

The design, maintenance and governance of centralized systems is complex. As well as in centralized systems, in decentralized ones limitations appear.

Limitations of decentralized systems

- Common goals are not easy to achive because of each node being independent so coordination must be designed in detail.

- Low cost/benefit ratio is not worthy in small systems that implement decentralized architectures.
- Regulation of single nodes is difficult. Nodes does not have superior nodes which are overseeing their behavior [24], [25].

Those characteristics lead decentralized systems to a list of advantages and disadvantages as well as centralized systems.

Advantages of decentralized systems

- Bottlenecks rarely appear.
- Huge availability is reached, as nodes are independent, some of them can be operating while others are not, so there are usually always active nodes running.
- Resources can be managed more efficiently as each node is responsible for managing its own resources [24], [25].

Disadvantages of decentralized systems

- Large global tasks are difficult to execute due to the lack of a command-and-control chain.
- Oversight does not exist in a regulatory manner.
- Difficulty on selecting wich node went done when failure takes place.
- Requests in a decentralized system are correctly addressed as a whole, but trying to select the specific node that has taken care of that particular task is complicated [24], [25].

COMPARISON BETWEEN TYPES OF SYSTEMS BASED ON THEIR CHARACTERISTICS

Once it has been established the difference between centralized and decentralized systems it is much easier to compare and differenciate between the feature of descentralization and distribution, explained in 2.2, in computing systems.

	Centralized	Decentralized	Distributed
Clock's type	Global	Each node has its own clock	Each node has its own clock
Units	Single central unit	Multiple central unit	Common decisions are reach by nodes by using consensus protocols
Failure dependency	Central unit fail causes entire system to fail	Failure of a central node does not imply on full system failure but on a part of it	Failure of independent nodes does not lead to a huge impact on the network as it can continue working normally

Table 2.1. COMPARISON BETWEEN TYPES OF SYSTEMS BASED ON THEIR CHARACTERISTICS

A blockchain network can be centralized or decentralized. It is very important to differenciate between centralization and distribution when talking about blockchain networks. While a blockchain is inherently distributed, as a copy of the ledger is stored in each node of the network, it is not inherently decentralized [26]. As mentioned in 2.3.2.3 the number of nodes that are members of a blockchain network is the main factor that defines the degree of decentralization of a blockchain network.

The goal of most blockchain networks it to achieve the highest level of decentralization as possible. Contributing nodes in decentralized distributed networks do not work on a piece of the work; instead, the interested nodes complete the full task. [24].

Peer-to-peer systems which are design under the principles of decentralization and distribution appear as shown in figure 2.12.

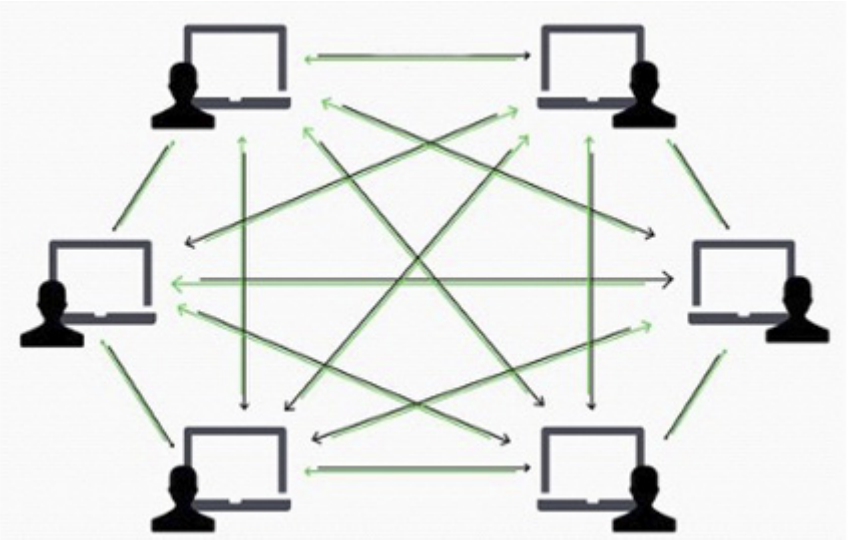


Fig. 2.12. PEER-TO-PEER DECENTRALIZED SYSTEM REPRESENTATION [24]

2.3.2.4 Layers of blockchain

Blockchain technology is the result of a mix of business ideas, economics, game theory, encryption, and computer science engineering. As a result, developing complicated applications on top of blockchain might be problematic, as real-world apps are typically highly complex. Developing blockchain solutions from the bottom up is a preferable option.

Abstraction of layers is useful for the built of products that helps the achievement of an open system. Having a separation of layers where blockchain solutions are being developed implies that changes made in one specific layer does not affect to the other ones. This concept of layering blockchain networks can be related with the TCP/IP communication protocol but this is a protocol used by blockchain as well as every internet application [24].

Although blockchain is a technology that is gaining a lot of popularity now, it is still a very new technology. New blockchain networks are emerging theses days and existing ones continues updating themselves. This is way there are still not global starndars that clearly divide blockchain into definited layers. In this section of the thesis, it will be presented one model of blockchain layers segregation, as shown in figure 2.13, that would probably match will the one adopted in the future for most blockchain and cryptocurrencies proyects.

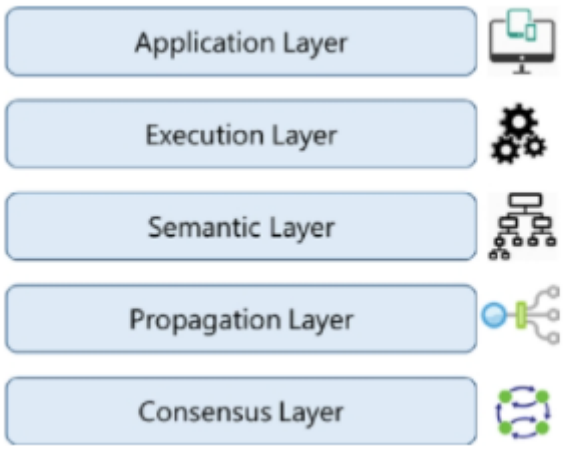


Fig. 2.13. BLOCKCHAIN LAYERS [24]

APPLICATION LAYER

Most applications that are design using blockchain technology are developed in the application layer, others are developed interwovening the application layer with any other layer of blockchain networks.
The application layer is the one that are located most outside from a high-level point of view, it is here where coding the desired functionalities of the applications takes place [27].

Traditional technology for software development is used in application layer, such us scripting, APIs, frameworks, etc. The aim of blockchain applications is to reach decentralization thus it is not used a client-server model to design them [24].

The concept of *off-chain networks* and *off-chain transactions* is gaining a lot of popularity because they allow transanctions of value from a blockchain network to the outside of itself [28]. The benefits of using off-chain networks is its zero/low cost of transacting value [29]. This allow developers to ensure that the application layer is used to perform the heavy tasks of the blockchain application so that bulky storage is performed outside the core of the blockchain which still remains light and effective.

EXECUTION LAYER

Instructions programmed in the application layer are executed in the execution layer, as it names implies. All the nodes that are taking part in the blockchain network do execute these instructions which can be simple instructions or complex ones, *smart contracts*.

All the nodes in a blockchain network have to execute programs/scripts independently. Deterministic execution of programs/scripts on the same set of inputs and conditions always produces the same output on all the nodes, which helps avoid inconsistencies [24].

The type of scripts that are execute in the execution layer depend on the type of blockchain that we are talking about and its goals or purposes it been design for. In Bitcoin blockchain, the type of scripts that need to be execute by every node in the network are simple ones that do not require a huge amount of instructions to be executed. On the other hand, in Ethereum and Hyperledger blockchains scripts are much complex, they are known as *smart contracts* which could be either a single instruction or a very complex coded script. In Ethereum blockchain, smart contracts are coded using *Solidity* as programming language which get compiled to Bytecode and executed on its own Ethereum Virtual Machine [24].

SEMANTIC LAYER

When a transaction is made, as we have explained before, a script of set of instructions is executed in the execution layer but it is required that the transaction gets validated, this validation takes places in the semantic layer.
In some blockchains, like in Bitcoin blockchain, when a user makes a transaction is it validated in the sematic layer by every node in the network by traversing previous transaction received by the user in order to see that the user has al least the amount it is speding in the current transaction. By traversing previous received transaction in order to validate current spending ones makes both, receiving and spending accounts, to get updated.

Rules in a blockchain network are defined in its semantic layer. Data models and structures, such us Merkle trees and their mekle root, which we are analyzing in next chapters of the thesis, are defined in this layer too. Usually, when receiving a transaction, a smart contract, is invoked where complex intructions are coded. Rules defined in the semantic layer also include how blocks are connected to each other in blockchain networks. Blocks are connected using hash functions that point to the previous one in the chain.
These are examples of general features of blockchain networks that are defined in the semantic layer but, as usual, each blockchain defines its own particular features that make them stand out from the other ones [24].

PROPAGATION LAYER

Connection between the nodes in the blockchain network is reach in the propagation layer. The propagation layer allows nodes to synchronize and communicate between them, it also defined as the *peer-to-peer communication layer*.
The propagation layer defines the broadcast to the whole network when one block is added to the network or when a transaction is made [30].

By design, most of the blockchains are designed such that they forward a transaction/block inmediately to all the nodes they are directly connected to, when they get to know of a new transaction/block [24].

Node's capacity and blockchain internal design are facts that directly affect on the propragation time of broadcasted transactions.

CONSENSUS LAYER

The base layer for most blockchain networks is the consensus layer. The consensus layer, as its name implies, is the layer where consensus in reach among all the nodes that are part of the blockchain. Consensus is the technical name attributed to the set of rules followed by everynode in the blockchain in other to stablish an agreement on one consistent ledger's state.
Each blockchain network defines its own consensus mechanism. Most known consensus algoright are Proof-of-work (PoW) which is currently used by Bitcoin and Ethereum that consist on minning or Proof-of-stake (PoS) which most new blockchains are using.

2.4. Bitcoin Blockchain

2.4.1. Introduction

Today's most popular blockchain network is Bitcoin whose white paper was proposed by Sathosi Nakamoto in 2008 as mentioned in section 2.3.1. Bitcoin is an electronic payment system which is based on cryptographic proofs. Referring to Bitcoin as a cryptocurrenciy it occupies the first place in the ranking with the largest market capitalization with a total of 916B dollars amount. Bitcoin blockchain is mostly used to transmit value between users using the bitcoin protocol but because the features implemented by its blockchain network, the transmission of monetary value between users it is not its only use. Bitcoin cryptocurrency is only the first applciation of the Bitcoin protocol.
Nodes configured as a decentralized distributed network, the consensus mechanism and the cryptographic algorithms implemented by Bitcoin enabled to the existence of fully distributed and trusted protocol which core can be the base for the creation of many applications.

In the next sections of the thesis, the main characteristics of the bitcoin blockchain will be presented and analyzed in a technical way.

2.4.2. Bitcoin network

2.4.2.1 Network architecture

Bitcoin network is design as a peer-to-peer (P2P) network in which every participant in the network, the nodes, are equal to each other. There are not a master node who control data or distribute taks to the rest of the nodes. Nodes are not designed to conform a hierarchical structure in which special nodes exist. Services are provided and consumed by

nodes in a peer-to-peer network and this serves as an incentive for participation.
Bitcoin blockchain, as every peer-to-peer network, is decentralized, resilient and open. Decentralization is the most significant core design principle in bitcoin peer-to-peer system and can only by achieved and maintained by a decentralized peer-to-peer consensus network. Since Bitcoin blockchain is running on the Internet, the TCP/IP protocol stack is used by the blockchain network as shown in figure 2.14 [24]

"The term "bitcoin network" refers to the collection of nodes running the bitcoin P2P protocol. In addition to the bitcoin P2P protocol, there are other protocols such as Stratum, which are used for mining and lightweight or mobile wallets. These additional protocols are provided by gateway routing servers that access the bitcoin network using the bitcoin P2P protocol and then extend that network to nodes running other protocols" [31].

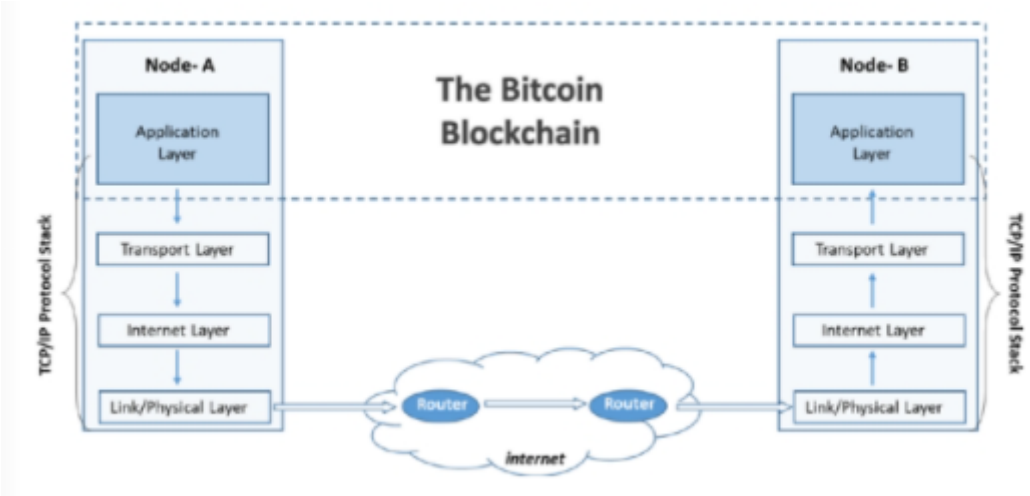


Fig. 2.14. BITCOIN BLOCKCHAIN NETWORK ON THE INTERNET [24]

2.4.2.2 Nodes

Despite every node is equal in a peer-to-peer network and there is not a hierarchical structure among nodes, different roles are assigned to each node, that means that the functions of a node or the tasks it is performing will differ depending of its role in the network. A *full node* contains a wallet node, a miner, a blockchain data base and a routing node as it can be seen in figure 2.15.

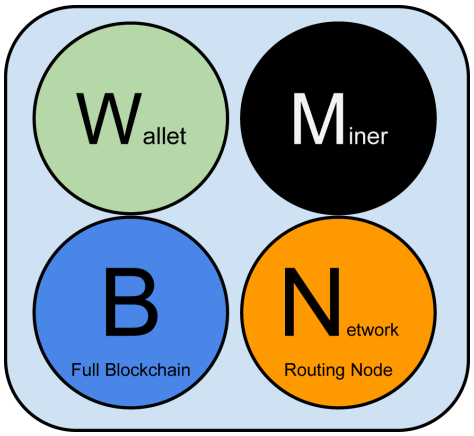


Fig. 2.15. BITCOIN FULL NODE [31]

Every full node contains a routing noe in order to participate in the bitcoin network, the rest of functionalities can be implemented by a specific type of node or not.

Full nodes in bitcoin blockchain, as the one represented above, contains an inmmutable up-to-date copy of the blockchain. Transactions are verified by full nodes without the need of external references.
Another type of node called *SPV* or *Lightweight nodes* exists which does not contain a copy of the entire blockchain but only a subset of it. Lightweight nodes also verify transactions by a method called Simplified Payment Verification (SPV). SPV or lightweight nodes are normally represented as in figure 2.16 [31], [32].

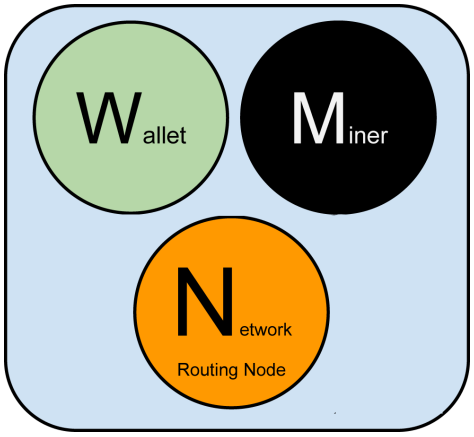


Fig. 2.16. LIGHTWEIGHT NODE [31]

Downloads and storages requirements are less intensive in lightweight nodes than in full nodes because it only requieres to download the headers of every block in the blockchain not the body of them.

"The relationship between full nodes and lightweight nodes exists because if it did not, then lightweight nodes would not be able to connect to the cryptocurrency network, which might lead them to use a centralized service instead. It is also important to note

that SPV nodes are effectively placing their trust in full nodes in ensuring that blocks and transactions are being correctly validated against consensus rules" [33].

Specialized hardware is used by mining nodes which can be either full nodes or SPV nodes. Mining nodes use the previous mentioned hardware to participate in the consensus mechanism and solve the proof-of-work algorithm which will be later explained.

User wallets can also be part of full nodes, they are represented in figure 2.15 as a circle named "wallet".

Different servers and nodes running other protocols can be found in bitcoin peer-to-peer network, this protocols are specialized in other functions like mining pools and light-weight client access [31].

In figure 2.17 it is seen every node type on bitcoin extended network.

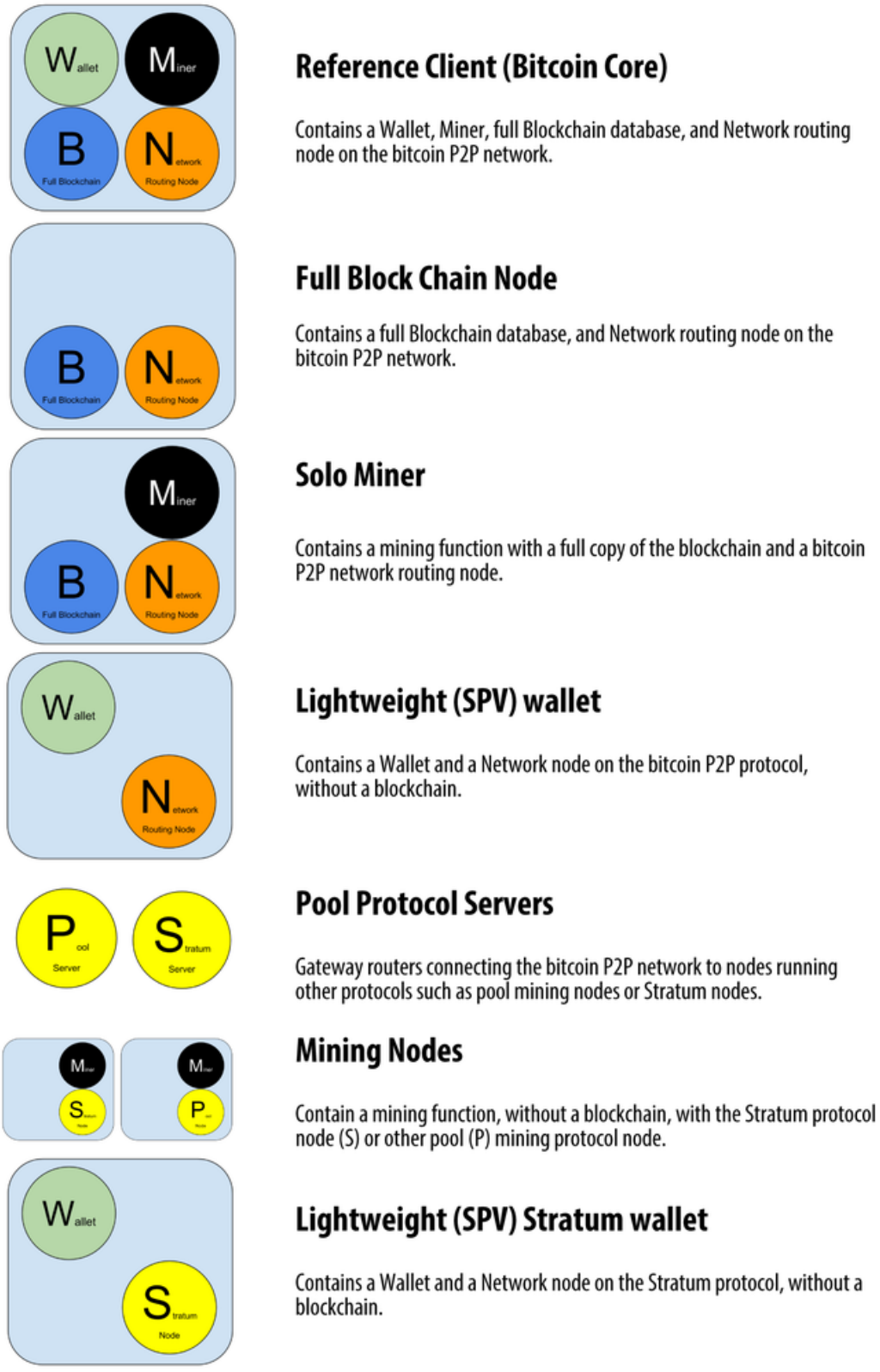


Fig. 2.17. NODES TYPES ON BITCOIN EXTENDED NETWORK [31]

Bitcoin blockchain is a global peer-to-peer network that operates globally as well, therefore nodes are spread all over the world as seen in figure 2.18 and 2.19 [34]. The fact that the nodes are spread all over the world is beneficial, this supports the concept of

decentralization since, as it will be seen in later chapters of the thesis, it is necessary to control 51% of the nodes in the network to be able to perform malicious behavior, thus, if the nodes are spread all over the world, it is more difficult to control 51% of them under the management of a central authority inside a specific country or its government.

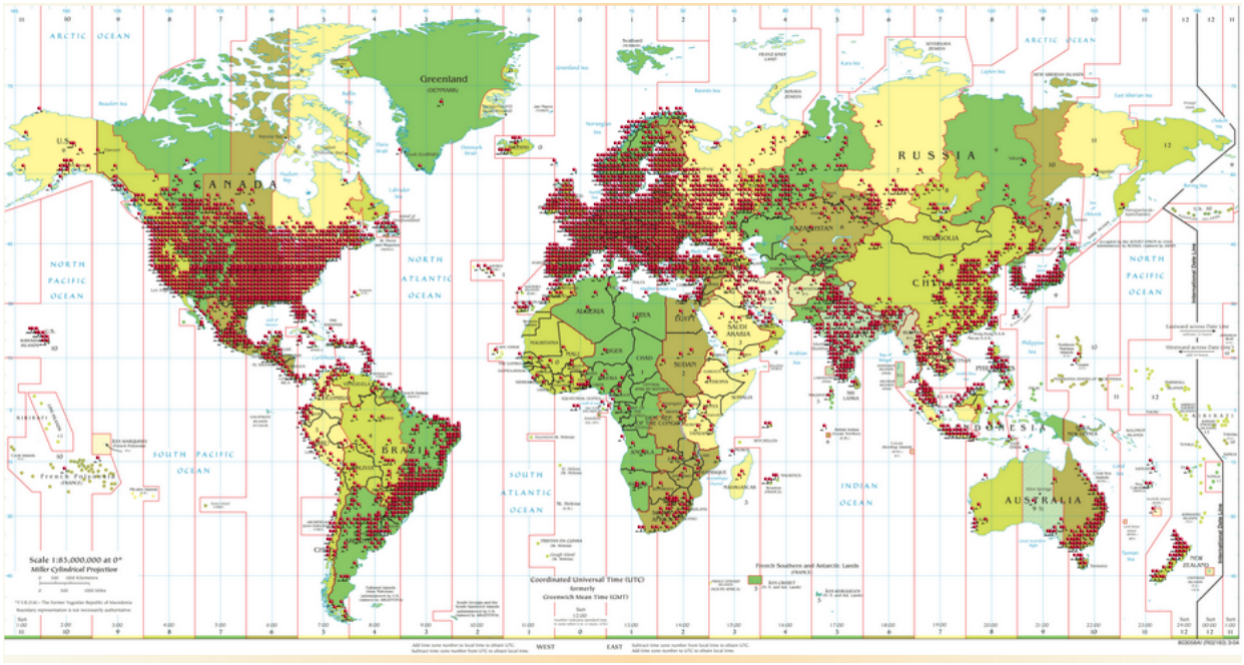


Fig. 2.18. BITCOIN NODES WORLD LOCATION [31]

Country	# of Bitcoin nodes (37 days)	# of Bitcoin nodes (1st day)	# of Internet users[15]	Bitcoin node rate (per 100.000)
United States	145.495	24.621	254.295.536	9,68
China	172.662	16.700	568.192.066	2,94
Germany	80.067	7.695	68.296.919	11,27
United Kingdom	43.369	6.849	54.861.245	12,48
Russian Federation	66.705	6.848	75.926.004	9,02
Canada	23.308	4.664	29.760.764	15,67
Netherlands	16.490	4.070	15.559.488	26,16
France	17.249	2.752	54.473.474	5,05
Australia	15.239	2.364	18.129.727	13,04
Poland	19.242	2.265	24.969.935	9,07
Spain	14.303	1.726	33.870.948	5,10
Ukraine	13.606	1.688	15.115.820	11,17
Italy	17.098	1.572	35.531.527	4,42
Brazil	16.452	1.476	99.357.737	1,49
Czech Republic	6.019	1.403	76.32.975	18,38
Taiwan	16.335	1.375	17.656.414	7,79
Sweden	7.958	1.366	8.557.561	15,96
Norway	4.036	1.016	4.471.907	22,72
Switzerland	5.463	933	6.752.540	13,82
Finland	4.692	901	4.789.266	18,81
Japan	6.631	843	100.684.474	0,84
Austria	7.012	828	6.657.992	12,44
Belgium	5.810	726	8.559.449	8,48
Argentina	5.863	663	23.543.412	2,82
Hong Kong	4.917	648	5.207.762	12,44
...	...	...	...	...
Anguilla	1	0	9.133	0,00
Burundi	1	0	128.799	0,00
Cape Verde	1	0	181.905	0,00
Dominica	1	0	40.349	0,00
Equatorial Guinea	1	0	162.202	0,00
Samoa	1	0	25.111	0,00
Sao Tome & Principe	1	0	39.515	0,00
Timor-Leste	1	0	10.461	0,00
Others (180 countries)	136619	15483	-	-
Total	872648	111475	-	-

Fig. 2.19. BITCOIN NODES LOCATION BY COUNTRY [31]

2.4.2.3 Extended network

The extended in the Bitcoin blockchain contains the bitcoin peer-to-peer protocol network and the nodes which are running some specialized protocols. These protocols are mostly mining nodes and lighweight wallet clients. A copy of the entire blockchain is not held by these nodes.

The rewards distributed to nodes participating in bitcoin blockchain minnin pools by applying the aforementioned protocols are a non-linear function of the pool's processing capacity. As a result, the teams are inherently unstable, and any reward allocation mechanism in a pool is likely to result in some miners preferring to transfer pools. [35].

Figure 2.20 is a representation of the extended bitcoin network itself [31].

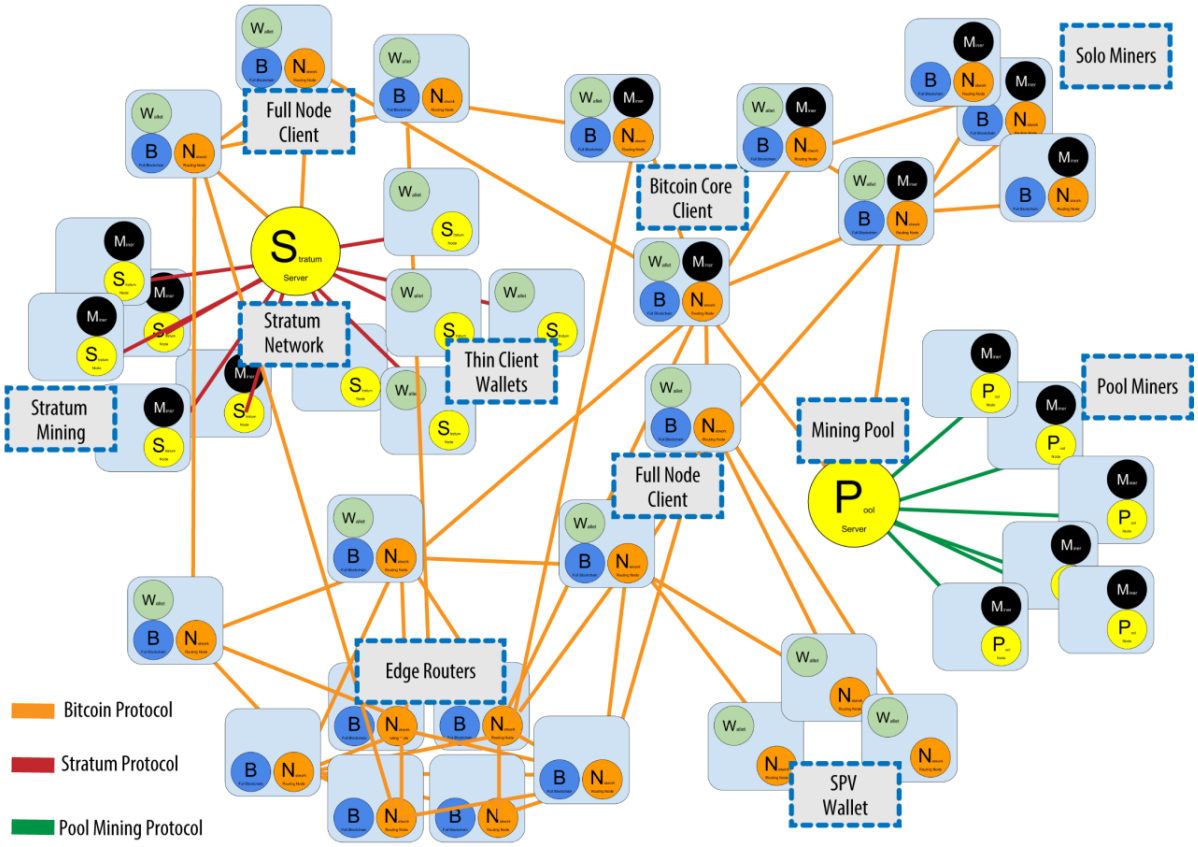


Fig. 2.20. EXTENDED BITCOIN NETWORK [31]

2.4.2.4 Network discovery

As in every distributed system, when a new node wants to join the bitcoin blockchain network, it has to follow a series of steps to be able to join the network. For a new node to become part of the blockchain first need to discover and connect to an existing node that is already part of the blockchain. These existing nodes are selected randomly by the one trying to join the blockchain because the geographicall location of the nodes is irrelevant.

When a new node has already discovered a node belonging to the blockchain it establish a TCP connection to the port 8333 or an alternative one. When the TCP connection is already established, a *"handshake"* happends by interchanging previously defined messages between the two nodes [31], [24].

- **PROTOCOL_VERSION:** defines the bitcoin peer-to-peer protocol version.
- **nLocalServices:** list of local services supported by the node.
- **nTime:** current time.
- **addrYou:** IP address of the remote node seen by another node.
- **addrMe:** IP address of the local node.

- **subver:** software run on the node.
- **BestHeight:** height of the node.

In figure 2.21 is it shown thhe actual Bitcoin code for "Version" message defined in net.cpp [24].

```
PushMessage( "version", PROTOCOL_VERSION, nLocalServices,
nTime, addrYou, addrMe,
                nLocalHostNonce, FormatSubVersion(CLIENT_NAME,
                CLIENT_VERSION,
                std::vector<string>()), nBestHeight, true );
```

Fig. 2.21. "VERSION" MESSAGE IN THE ACTUAL BITCOIN CODE [31]

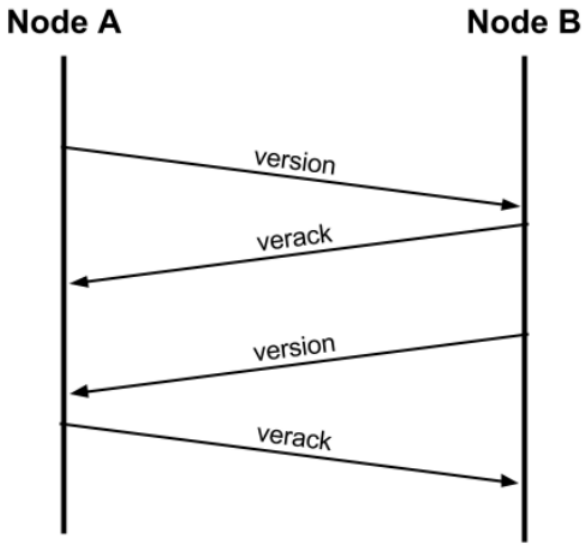


Fig. 2.22. INITIAL HANDSHAKE TO STABLISHED CONNECTION BETWEEN PEERS DIAGRAM [31]

In bitcoin blockchain there are a type of nodes listed on the client as *seed nodes* that are long running stable nodes. When a new node want to discover and connect to a belonging to the blockchain node it can connect to these seed nodes in order to discover new nodes fater, altough this is not a must. To decide if it is desired for the new node to connect to a seed node the -dnseed option of the Bitcoin core is used, which will be set to 1 if it is desired for a new node to connect to a seed one. In case it is not desired to connect to a seed node, it is required intrude an IP address of at least one node that belongs to the blockchain to establish connection [36].

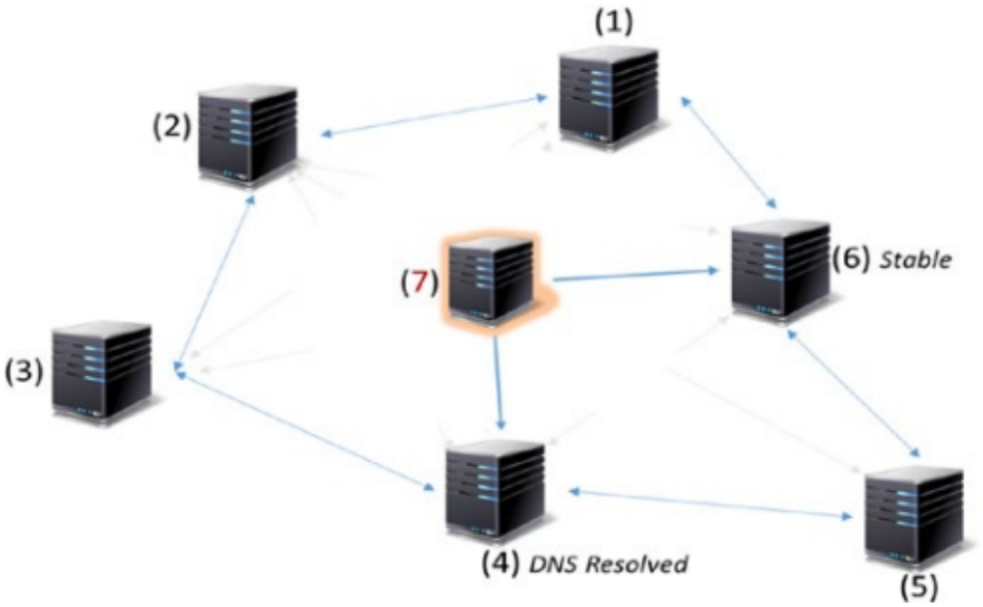


Fig. 2.23. NEW BITCOIN NODE CONTACT SOME PEERS [24]

When the connection is already stablished, the new node sends an addr message to the already contacted node, it can be just one or more than one as seen in figure 2.23, and when the addr has arrived, it will be forwarded to neighboring nodes so that a more robust connection is established and more nodes are aware of the existence of the new participant to the network.
The new node can also send a getaddr message to its neighbors and they will send back a message containing their IP addresses or IP port as seen in figures 2.24 and 2.25 [31].

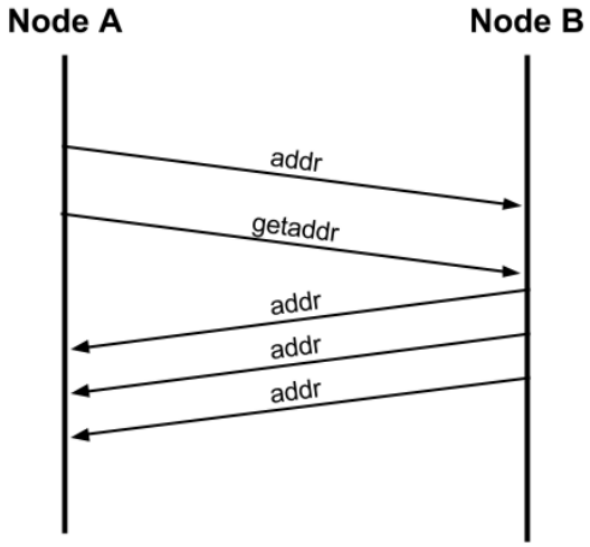


Fig. 2.24. ADRESS PROPAGATION AND DISCOVERY DIAGRAM [31]

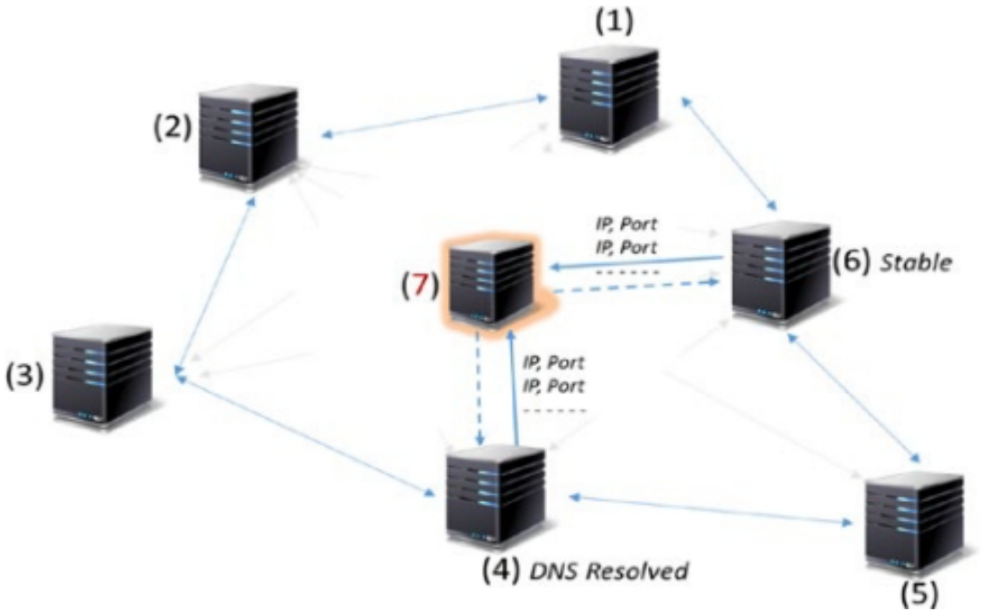


Fig. 2.25. BITCOIN NODES RESPOND NEW NODE REQUEST [24]

Since nodes in bitcoin blockchain are not permanent, a large number of nodes disconnect and new nodes connect on a daily basis, the established paths are not reliable because they can break. Thus, it is necessary for a node to established new connections with new nodes at the same time they are loosing old connections as well as assist other nodes when they bootstrap.

"After bootstrapping, a node will remember its most recent successful peer con- nections, so that if it is rebooted it can quickly reestablish connections with its former peer network. If none of the former peers respond to its connection request, the node can use the seed nodes to bootstrap again" [31].

The connection between nodes has to be continuously refreshed, if there is no traffic exchange between nodes, they will send each other messages to maintain their connection. A node is assumed to be disconnected if it has not established a connection after 90 minutes.

2.4.3. Blocks

2.4.3.1 Introduction

Blockchain networks and their data structure are organized as an ordered back-linked list of blocks of trasactions. The term *"height"* is used to refer the length, or number of blocks, of a blockchain networks. This term is used because blockchain networks are commonly visualized as a vertical stack.

Each block in a blockchain network is linked to its previous block in the chain. Blocks in a blockahin are identified by a *hash*. This hash is an unique hash generated by the SHA256 cryptographic hash algorithm. Each block store its hash identifier on the header of the block. The connection between blocks is made through a hash that points to the previous block, known as the parent block, this identifier, "previous hash pointer", is also stored in the header of each block. The only block in the chain that does not contains a hash pointing to its previous one is, obviously, the first block of the blockchain, the one known as genesis block. A graphical representation of the bitcoin blockchain data structure can be seen in figure 2.26.

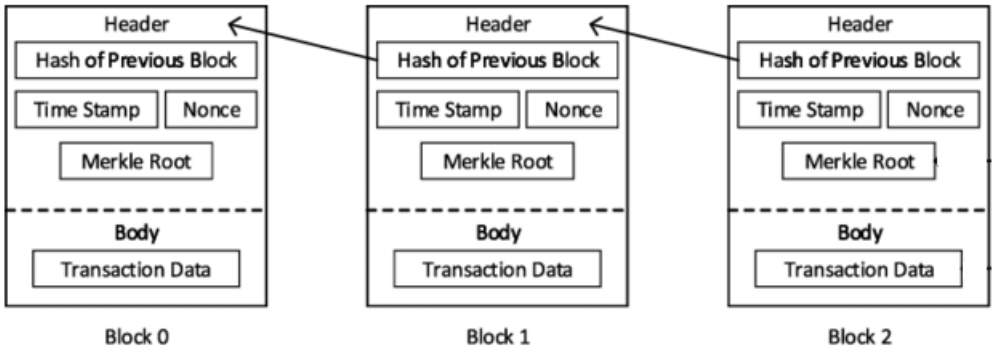


Fig. 2.26. BITCOIN BLOCKCHAIN'S BLOCKS GRAPHICAL REPRESENTATION

2.4.3.2 Forks

A curious phenomenon can happen in the blockchain, a *"fork"*. As it has been declared before, each block can only have a parent block, or in other words, the hash of each block can only points to a single previous block, the immediately previous one. However a block can have several *children blocks*, this happends when several children blocks's hases point to the same parent block. The situation that takes place when various blocks are discovered or generated by different nodes at the same time, or almost at the same time, is known as a *"fork"* [37]. At a specific point in the course of blockchain development, this *"fork"* situation will be resolved and one of the temporary subchain blocks created by the appearance of several child nodes will become part of the main blockchain and will be added as another node of the chain [38]. The mentioned type of fork is known as *"temporary fork*.

The term fork also refers to software updates that a blockchain network runs by decision of the programmers who are taking part of it. There two main types of blockchain forks.

- **Hard fork:** Consensus rules are normally changed when a hard fork takes place. Hard fork are incompatible with older version of the software [38]. A hard fork makes the previously added block to the blockchain become invalid, therefore every

node is required to upgrade to the latest version to remain part of the blockchain. In other words, a hard fork splits the path of the asset's underlying blockchain, where separate, updated blocks begin to follow new sets of rules [39].

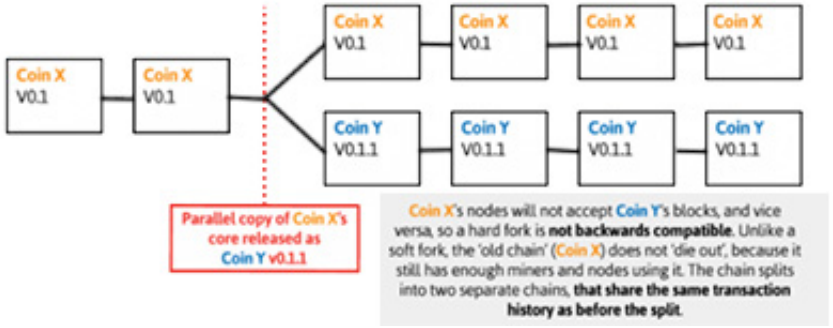


Fig. 2.27. HARD FORK [40]

- **Soft fork:** Soft forks are compatible with older versions of the software. Soft forks are common events on blockchain networks as it takes places when the core software part determine that the block specifications are updated [40].

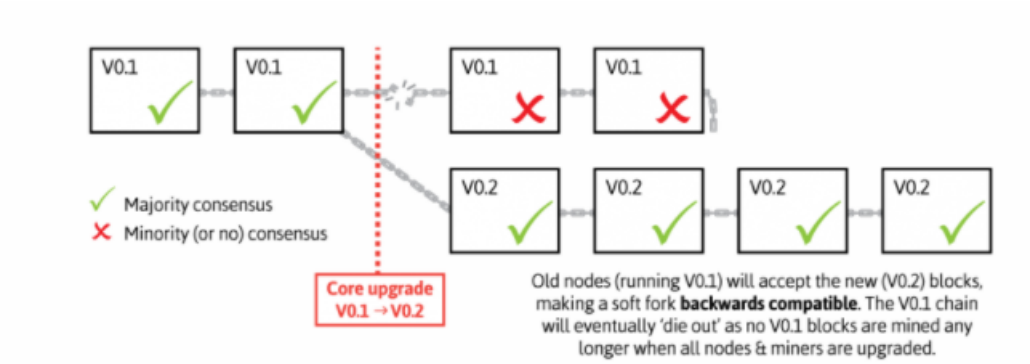


Fig. 2.28. SOFT FORK [40]

2.4.3.3 Block structure

Blocks in a blockahin are simply a data structured design to contain the aggregates transaction of inclusion in the public ledger. However, blocks also include some other information. The strucute of a block on the Bitcoin blockchain is equal to every block in the network. The size of the block header is 80 bytes, transactions are normally 250 bytes size and there are 500 transactions registered in each block [31]. In table 2.3 it can be seen the different fields contained in a block [24], [41].

	Backwards compatible	Rules	Blockchain splitted
Hard fork	No	Usually loosens the rules	Will split the blockchain into two separate chains with a common history (up to the fork)
Soft fork	Yes	Usually tightens rules	Will normally not split the blockchain

Table 2.2. COMPARISON BETWEEN HARD FORKS AND SOFT FORKS

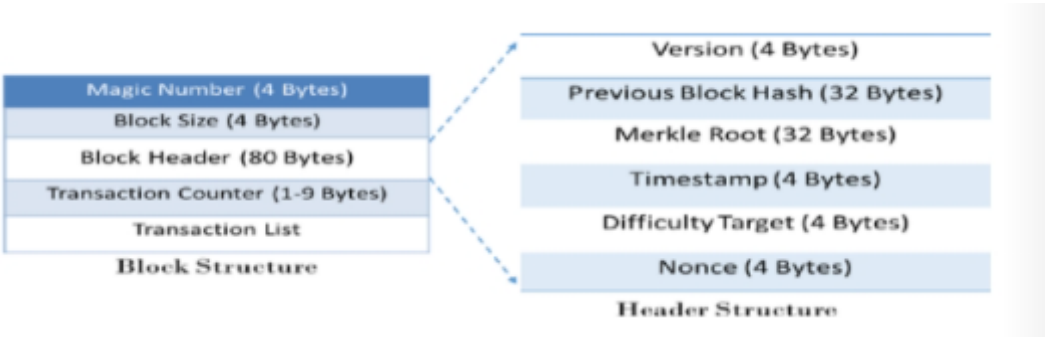


Fig. 2.29. STRUCTURE OF A BLOCK [24]

Field	Size	Description
Magic Number	80 bytes	Starting point of the block and type
Block Size	4 bytes	Number of bytes a block dimension is
Block Header	80 bytes	Several fields form header of the block
Transaction Counter	1-9 bytes Type: VarInt	How many transactions are contained in a block
Transaction List	Could vary	Transactions that are stored in a block

Table 2.3. STRUCTURE OF A BLOCK

Related to the block header, this is made up of 3 data sets. The first one, as mentioned earlier in this thesis, contains a hash pointer which is pointing to the previous block header, the parent block, in the blockchain which is hashed using the SHA256 algorithm producing a 256-bit result or 32 bytes [24]. The second element of information

provides the difficulty of the mining competition, the timestamp and the nonce, while the third set contains the root of the Merkle tree, which is used to summarize all transactions in the block.

Field	Size	Description
Version	4 bytes	Bitcoin protocol version number that is running must be the same in every node
Previous Block Hash	32 bytes	Hash of the block header of the previous block in the chain
Merkle Root	32 bytes	Root hash of this Merkle tree. The transactions are hashed and ordered in a Merkle Tree
Timestamp	4 bytes	Time when the block was created using Unix time format
Difficulty Target	4 bytes	Difficulty level of the consensus mechanism (PoW)
Nonce	4 bytes	Random number which is seek by mining node in the network

Table 2.4. BLOCK HEADER

2.4.4. Transactions and cryptography

2.4.4.1 Introduction

Transactions and their security is one of the most important parts of the bitcoin blockchain. The bitcoin network is designed in all its facets to ensure the creation of secure transactions. Transactions in the Bitcoin blockchain are simply data structure that store the information of value exchanged between different parties [31].

2.4.4.2 P2PKH

Most of the transactions that are processed on the Bitcoin blockchain are P2PKH, Pay-to-Public-Key-Hash, transactions. When processing a transaction as a P2PKH transaction, the output, which corresponds to the bitcoin address, is encrypted witha public key hash [31].

Processing P2PKH transactions always follow the same steps [42]:

- Previous transaction identificaction which contains UTOXs that you have control over.
- Inputs outpoints creation of the new transaction to identify the previous transaction's UTXO to spend.
- Output of the new transaction creation so the locking script will contain the conditions in which the newly created UTXOs can be spent/unlocked by the next transaction.
- Unlocking script creation which meets the conditions set by the previous transactions output's locking script.

2.4.4.3 SHA256

SHA256 is the cryptographic hash algorithm used in Bitcoin blockchain to reach integrity and security in the information stored in each block.

Every hash algorithm allows the creation of a hash using a specific document or data as input but not the other way around, therefore hash algorithms are one way only. Hashes created using SHA256 lengths are always the same,the lenght of the output hash is independent from the lenght of the input content that is desired to be hashed [43].

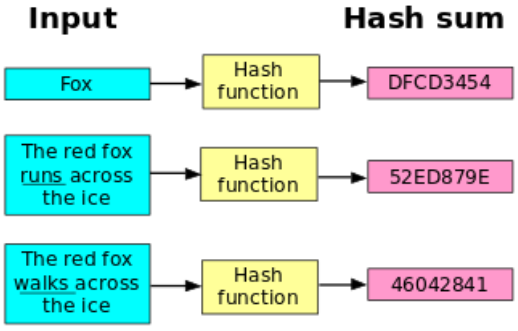


Fig. 2.30. 256SHA HASH FUNCTION EXAMPLE [44]

Every node on the Bitcoin blockchain would have a copy of the 64-character hash that reflects the data that, for example, comprises an entire block. Any manipulation of this

information attempting to change any character of the validated hash would be quickly identified and deleted after it has been validated by the network. [45].

"This property is useful in the generation and verification of digital signatures and message authentication codes, as well as the generation of random numbers or bits. message authentication codes, as well as the generation of random numbers or bits" [46].

2.4.5. Consensus mechanism and mining

2.4.5.1 Introduction

Since blockchain networks are made up of a large number of nodes operating in a decentralized manner, common rules are needed to be taken by every node in the network to perform transaction authentication tasks on the network and reach an unified agreement on the state of the network by a decentralized mechanism.

This mechanism is known as *consensus mechanism*.

"Consensus is basically a distributed computing concept that has been used in blockchain in order to provide a means of agreeing to a single version of truth by all peers on the blockchain network" [7].

Each blockchain network defines the type of consensus mechanism they are using to unifique their decisions. Bitcoin blockchain's consensus mechanism is Proof-of-work (PoW) mechanism which define the rules for adding new blocks to the blockchain which is commonly known as *"mining"* or *"block mining"* maintaining decentralization.

Everything in the Bitcoin blockchain is represented as transactions. Those transactions to be made need to consume one or more previous transactions in the blockchain as input. When a miner proposes a block as the result of the consensus mechanism puzzle it does not mean that it would contain a fraudulent transaction but in that case the rest of the nodes in the Bitcoin blockchain will reject it. If all transactions included in the block are legitimate, it will be quickly validated by the other nodes.

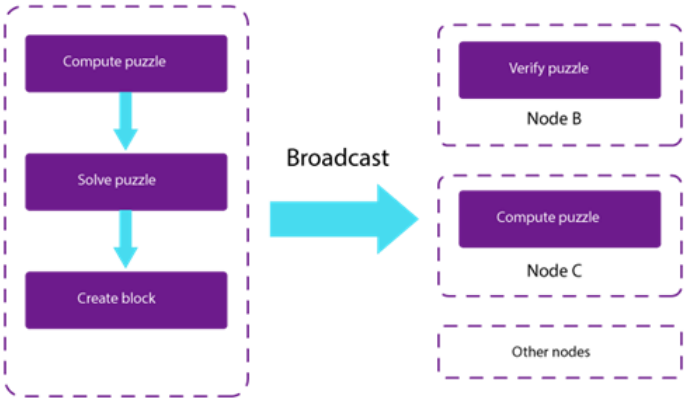


Fig. 2.31. Consensus mechanism diagram

2.4.5.2 Proof-of-work

Bitcoin consensus mechanism, proof-of-work, involves the nodes of the blockchain having to solve a mathematical problem, when the problem is solved, the new block is added to the blockchain and verified by the rest of the nodes. The difficulty of this mathematical puzzle is not fixed. This difficulty varies depending on the number of "miners" or computational power in the blockchain at that moment, if they increase, the difficulty of the puzzle to be solved increases with them so that the distribution of rewards to the miners for solving the mathematical puzzle can be maintained. This mathematical puzzle is usually solved and a block is added to the blockchain in an average of 10 minutes per block [41], [47]. The fact that the mathematical problem to be solved is difficult and resource-consuming, such as the cost of the mining equipment and above all the computational power and electricity costs, makes the blockchain more secure because a hacker or malicious agent who wanted to introduce a fraudulent transaction would have to invest a lot of resources to have at least a chance to do so.

"The economic incentive behind proof of work is a perfect example of how to keep the consensus in a completely decentralized network. To be a node/miner in a proof of work blockchain requires hardware resources, electricity and time, but if nodes play by the rules, i.e. according to the consensus mechanism, they receive a reward. If nodes try to attack the blockchain, tamper with data or perform double-spending, they will not get the reward, and they will be just wasting resources" [41].

Proof of work adventages:

- Network becomes highly scalable.
- Increase of security network as the number of nodes increase.
- Miners receive an incentive for their contribution.

- Very secure consensus mechanism.

Proof of work disadvantages:

- If the network is not very big (a few number of nodes are members of the network) it is vulnerable of receiving 51% attacks.
- The difficulty of mining typically increases, and mining hardware becomes obsolete soon.
- Specialized and expensive mining hardware is required to be profitable.
- Huge amoumt of electricity is required.
- Usually have low throughput.
- The time needed to a transaction to be valid could be be significantly large.

2.4.5.3 Comparison between PoW and PoS

As it has been mentioned in previous chapters of this thesis, each blockchain network is beign design with its own consensus mechanism. The most famous and common used consensus mechanisms are proof-of-work and proof-of-stake which present very marked differences.

	Proof-of-work	**Proof-of-stake**
Mining a block	Determined by the amount of computing work	Determined by the amount of stake
Distribution of reward	Received by the miners that took part in resolving the mathematical puzzle	Validator does not receive a block reward as they are paid a network fee
Competition	Determined by the amount of computing work	Determined by the amount of stake
Adding a malicious block	It is needed 51% of the network computing power	It is needed to hold 51% of all cryptocurrencies on the network
Efficiency and reliability	Less energy-efficient and less expensive but more reliable	Far more cost and energy-efficient but less reliable
Security	The greater the hash, the more secure the network is	In return for a payment, staking helps lock crypto assets to safeguard the network.
Forking	Naturally prevent constant forking	Forking is not automatically discouraged

Table 2.5. COMPARISON BETWEEN POW AND POS

2.5. Ethereum Blockchain

2.5.1. Introduction

Once Bitcoin power was already recognized by people, developers started to transitionate into the development of cryptocurrency applications. At that point the dilemma was very clear: start building these applications on top of the Bitcoin blockchain or either create a whole new Blockchain. Building on the top of Bitcoin had obvious limitations such as the transaction types, the data types and the size of the storaged data. Developers might have to deal with workaround solutions as the creation of off-chain layers but that was a direct negation of the advantages of using a public blockchain. The creation of a new Blockchain seemed to be best approach.

In October of 2013, Vitalik Buterin [48] proposed a model of flexible and scriptable contracts implementation to the Mastercoin [49] team to substitute the specialized contract language used in Mastercoin at that time but the proposed changes was too radical for being fitted into Mastercoin roadmap.

In December of 2013, Vitalik Buterin shared a whitepaper that presented the idea behind Ethereum [50]: a Turing-complete, general-purpose blockchain. As in most technology project, the Ethereum Blockchain project and whitepaper are constantly changing and readapting to the different situations or problems encountered [51].

One of the first persons to come out to Vitalik and offer to help with his C++ programming abilities was Dr. Gavin Wood, who eventually became the creator of Polkadot. Gavin finally became the cofounder, codesigner, and CTO of Ethereum.

Ethereum was tought as a blockchain withour a specific purpose, but a blockchain where a variety of applications could be developed on top of it.

"The idea was that by using a general-purpose blockchain like Ethereum, a developer could pro- gram their particular application without having to implement the underlying mech- anisms of peer-to-peer networks, blockchains, consensus algorithms, etc. The Ethereum platform was designed to abstract these details and provide a deterministic and secure programming environment for decentralized blockchain applications" [52].

In next sections of the thesis some technical aspects and main features as well as applications or protocols developed over the Ethereum Blockchain will be presented and analyzed so that the reader can gain a notion of knowledge necessary to understand the service prototype developed in the thesis.

2.5.2. Innovations

Ethereum Blockchain design was not for a money transaction only use, the whole protocol aims to act as a decentralized computer.

The Ethereum network is a decentralized open-source and distributed blockchain network whose native cryptocurrency is called Ether *(ETH)*. The execution of transaction and interaction with the decentralized applications built on top of the Ethereum network is done by using Ether.

The development, deployment and execution of decentralized applications, known as DApps is possible by the implementation of Smart Contracts which are been analyzed in future chapters of the thesis.

"Ethereum's co-founders include Buterin, Gavin Wood, Jeffrey Wilcke, Charles Hoskinson, Mihai Alisie, Anthony Di Iorio and Amir Chetrit. The co-founders also set up the Ethereum Foundation in Switzerland, a non-profit organization dedicated to supporting the Ethereum network.

In July 2015, the Ethereum network was launched as one of the most ambitious projects in the crypto space with the goal of decentralizing everything on the internet. Similar to Bitcoin, Ethereum is a decentralized platform without a governing central authority that uses PoW to ensure malicious actors aren't able to tamper with the blockchain data" [53].

Solidity is the programming language used to write the Smart Contracts, programmable scripts, in the Ethereum Blockchain which is compatible with the Ethereum Virtual Machine (EVM) which has been also adapted and therefore is compatible with other newer Blockchains. The implementation of the concept of Smart Contract lead to an incredible wide-range of potential applications developed on the Ethereum Blockchain. innovation in the network and its applications is emerging with the birth of decentralized financial applications *(DeFi)* [54], non-fungible tokens *(NFT)* [55], decentralized gaming applications *(GameFi) [56]* with the implementation of *Play-to-earn* protocols [57], decentralized autonomous organizations *(DAOs)* [58] and those who are going to be created in the upcoming near future [59].

The following table shows the most remarkable differences between the Bitcoin and the Ethereum blockchain [53], [60].

2.5.3. Smart Contracts

Smart Contracts are inmmutable and deterministically pieces of code or entire programs that are executed in the context of an Ethereum Virtual Machine as part of the Ethereum network protocol [52], [61].

Characteristics of Smart Contracts:

- Ether is used for the execution of Smart contracts.
- Imutability is one of the most remarkable characteristics of smart contracts. After the development of a Smart Contract and the deployment of it, the contract cannot

	Bitcoin	**Ethereum**
Creators	Satoshi Nakamoto	Vitalik Buterin, Charles Hoskinson, Gavin Wood, Jeffrey Wilcke, Mihai Alisie, Anthony Di Iorio and Amir Chetrit
Launch date	January 2009	July 2015
Currency vs platform	Alternative to traditional fiat currencies	Platform to run programmatic contract and applications using Ether
Consensus algorithm	Proof-of-Work	Proof-of-Work *(transicionating to Proof-of-Stake in Casper update with the launch of Ethereum 2.0*)
Block time	10 minutes average	15 seconds average
Transaction speed	7 transactions/second	30 transactions/seconds
Supply	21 million BTC	Infinite

Table 2.6. COMPARISON BETWEEN BITCOIN AND ETHEREUM BLOCKCHAINS

be change under any condition. In traditional software programs can be updated and re-executed but in terms of Blockchain Smart Contracts, it is imposible to introduce changes once deployment has taken place.

- The output generated after the execution of a smart contract does not vary depending on which person execute the contract, the outcome is always the same.

  contracts operate in a relatively constrained execution context. They have access to their own state, the transaction's context.

- Every node in the Ethereum network runs a local instace of the Ethereum Virtual Machine (EVM) operating as a whole as it was a "world computer" [52].

The four major phases of the life cycle of smart contracts are shown in the following figure [62].

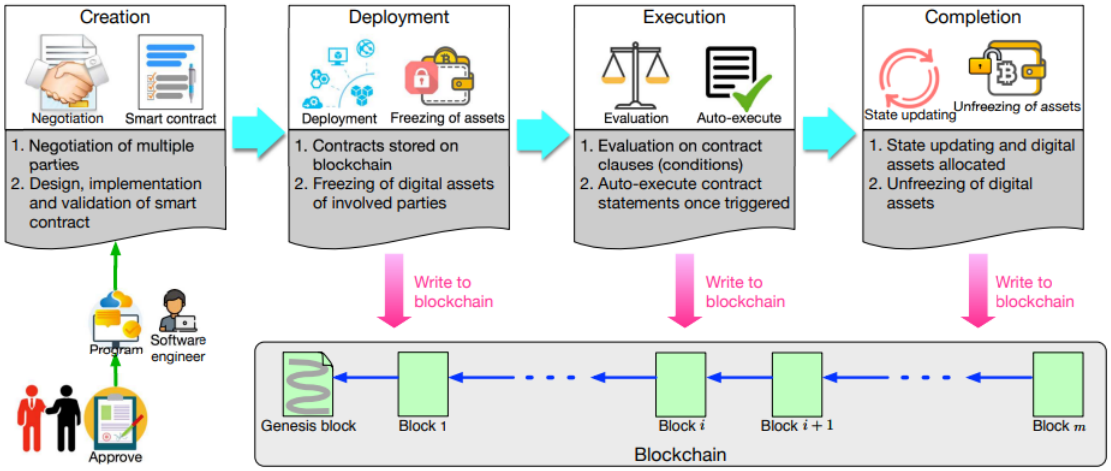


Fig. 2.32. SMART CONTRACTS LIFE CYCLE [63]

2.5.4. Decentralized Applications (DApps)

A combination of smart contracts and a user interface executed over a decentralized network lead to the concept of decentralized applications, also known as DApps.

A decentralized application is formed by a backend code running on a decentralized peer-to-peer network which can also implement a frontend code with an user interface written in any language as a normal application which is programmabled to communicate with the backend code running on the blockchain, normally as a smart contract. Frontend part of decentralized applications can be hosted on decentralized storage platforms such as IPFS [64].

Decentralized application advantages [65].

- **No inactivity time:** The network as a whole will always be able to service clients seeking to engage with the smart contract after it has been deployed in the main app and on the blockchain. As a result, hostile actors are unable to conduct denial-of-service attacks against specific dapps.
- **Privacy:** There is not need of providing your personal identity to implement a dapp or interact with it.
- **No censorship:** Users cannot be prevented from submitting transactions, establishing dapps, or accessing data from the blockchain by any entity in the network.
- **Integrity of data:** Due to cryptographic primitives, data saved on the blockchain is immutable and irrefutable. Malicious actors are unable to fabricate transactions or other publicly available data.
- **No need of trust software:** Without the need for a centralized partie, smart contracts can be examined and guaranteed to execute predictably. This does not apply

in traditional models; as users use online banking systems, they must trust that banks would not misuse their financial information, distort their records, or attack them.

Decentralized application implications [65].

- **Maintenance:** Because the code and data published on the blockchain are more difficult to alter, decentralized apps can be more complex to maintain. Even if defects or security problems are discovered in an older version, it is difficult for developers to make changes to their decentralized apps (or the underlying data contained in a dapp) once they have been released.
- **Overload:** There is a lot of operational overhead, and scaling is difficult. Every node processes and stores every transaction in order to achieve the degree of security, integrity, transparency, and reliability that Ethereum aims to. Furthermore, Proof of Work necessitates time. According to a fast estimate, the overhead is around 1 000 000 times, which is the current computing norm.
- **Congestion of the network:** If a decentralized program uses certain computing resources, the entire network is backed up, at least in the present form. The network can only process roughly 10 transactions per second at the moment; if transactions are submitted at a higher pace, the pool of unconfirmed transactions would quickly grow.
- **Experience of user:** Designing user-friendly experiences can be more complex; the ordinary end user may find it difficult to establish a set of tools required to safely connect with the blockchain.
- **Centralization:** The foundation of Ethereum is user and developer-friendly solutions, which may resemble centralized services. For example, such services may store keys or other sensitive server-side information, service a front-end using a centralized server, or otherwise run important business logic on a centralized server before writing to the blockchain. Many (if not all) of the benefits of the blockchain over the old paradigm are now gone.

2.5.5. Ethereum Virtual Machine

The Ethereum protocol was created to be operating in a continous, uninterrupted and immutable mode. The Ethereum Virtual Machine (EVM) is the enviroment where all the smart contracts and Ethereum accounts are hosted. Ethereum has an unique "canonical" state at every given block on the blockchain, and it is the EVM that establishes the rules for computing a new valid state from block to block. [66].

Ethereum is a distributed state machine rather than a distributed ledger. The state of Ethereum is a large data structure that not only stores all accounts and balances, but also

serves as a home for the state machine. This can alter from block to block according to a set of rules, and it can also run arbitrary machine code. The EVM defines the particular rules for updating the state from block to block.

The next figure is a representation of the Ethereum virtual machine [66].

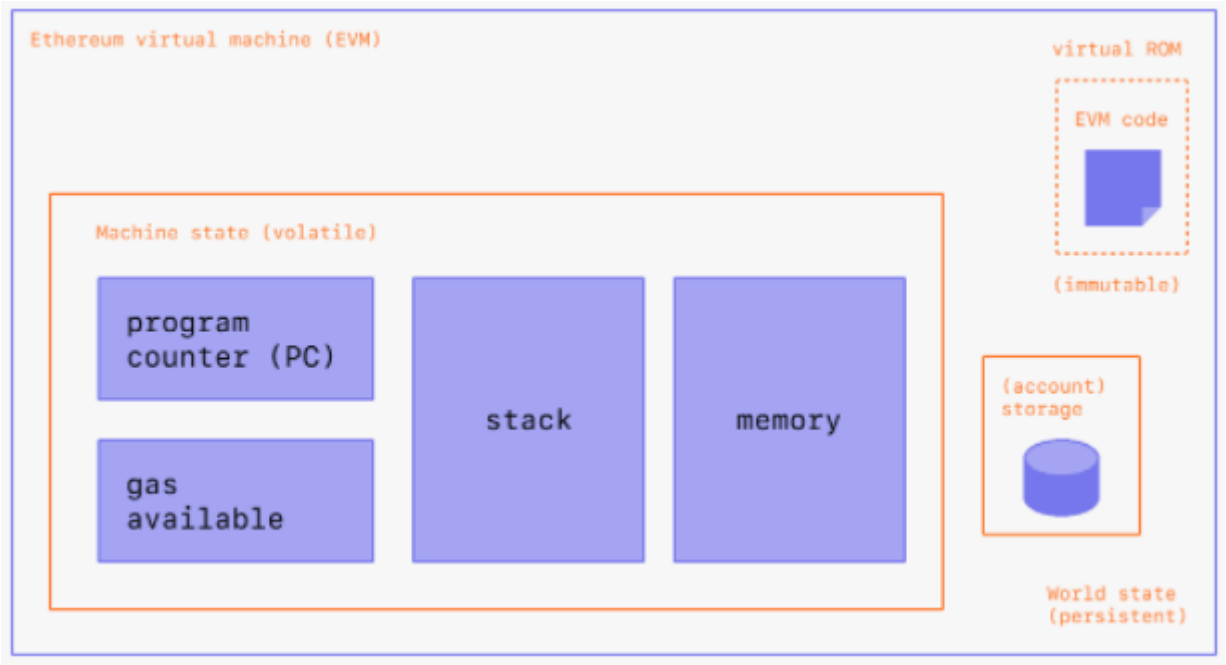


Fig. 2.33. ETHEREUM VIRTUAL MACHINE [66]

2.5.6. Tokens

As it has been explained in previous sections of this thesis, Ethereum is not a single chain of blocks that records transactions, commonly known as *"public ledger"*, Ethereum is a decentralized platform that allows the creation of decentralized applications which execute smart contracts deployed on the Ethereum virtual machine by the use of Ether. Ether is the gas currency on the Ethereum blockchain but Ethereum protocol provides the possibility of creation of tokens over the Ethereum Blockchain. This creation of new tokens lead to an incredible wide, almost infinit, range of opportunities for the development of new decentralized application and business models.

There are some different types of tokens, each one with some specific characteristics. In this thesis it will be presented and analyzed the standar Ethereum token ERC-20, and the ERC-721 standar token which result very important for the service provide on the thesis.

2.5.6.1 ERC-20 standar

ERC-20 present a standard suitable functional tokens. Every ERC-20 token is exactly the same as another ERC-20 token, they are interchangeable, because of one of their properties (in type and value). For example, an ERC-20 token acts the same as ETH, 1 token is and always will be the same as all other tokens.

Fabian Vogelsteller proposed in November 2015 the ERC-20 [67], which is a token standard that makes possible the implementation of an API for tokens within smart contracts. The ERC-20 standard includes features such as moving tokens between accounts, getting information about an account's current token balance, and seeing the entire token supply available on the network. The ERC-20 standard also allows for the spending of a token from one account by a third party account [68].

The main utility of these tokens is to standardize the interface for the creation and issuance of new tokens on the network. This is achieved by enforcing certain rules and parameters for their acceptance.

"The goal and need for ERC-20 tokens, is to design a standard, to create interoperability and compatibility between tokens and foster improvements in the Ethereum ecosystem. This thanks to the fact that ERC-20 tokens greatly facilitate the work of creating new tokens. Since the infrastructure was designed for it. It was also accompanied by tools for that purpose such as the Solidity programming language, or the EVM virtual machine" [69].

In the next figures the methods and events provided by the ERC-20 standar are shown [68].

```
1    function name() public view returns (string)
2    function symbol() public view returns (string)
3    function decimals() public view returns (uint8)
4    function totalSupply() public view returns (uint256)
5    function balanceOf(address _owner) public view returns (uint256
     balance)
6    function transfer(address _to, uint256 _value) public returns (bool
     success)
7    function transferFrom(address _from, address _to, uint256 _value)
     public returns (bool success)
8    function approve(address _spender, uint256 _value) public returns (bool
     success)
9    function allowance(address _owner, address _spender) public view
     returns (uint256 remaining)
```

Fig. 2.34. METHODS IN ERC-20 STANDAR [68]

```
event Transfer(address indexed _from, address indexed _to, uint256 _value)
event Approval(address indexed _owner, address indexed _spender, uint256 _value)
```

Fig. 2.35. EVENTS IN ERC-20 STANDAR [68]

2.5.6.2 ERC-721 standar

A non-fungible token (NFT) is token who has the characteristic of been unique. Non-fungible tokens usage fit perfectly on platforms that offer collectible items, acess to keys, lottery tickets and some other unique items. Dieter Shirley in late 2017 was the developer that makes the proposal for creating this standard, the EIP-721 [70].

The ERC-721 standard, in particular, was created with the goal of producing interchangeable tokens that were both unique and non-fungible.

With the launch of ERC-721 tokens on the Ethereum blockchain, a whole new token ecosystem arose, fueled by the notion of digital scarcity and uniqueness, in which the value of things is preserved and grown owing to the uniqueness of their qualities.

In the next figures the methods and events provided by the ERC-721 standar are shown [71].

```
function balanceOf(address _owner) external view returns (uint256);
function ownerOf(uint256 _tokenId) external view returns (address);
function safeTransferFrom(address _from, address _to, uint256 _tokenId, bytes data) external payable;
function safeTransferFrom(address _from, address _to, uint256 _tokenId) external payable;
function transferFrom(address _from, address _to, uint256 _tokenId) external payable;
function approve(address _approved, uint256 _tokenId) external payable;
function setApprovalForAll(address _operator, bool _approved) external;
function getApproved(uint256 _tokenId) external view returns (address);
function isApprovedForAll(address _owner, address _operator) external view returns (bool);
```

Fig. 2.36. METHODS IN ERC-721 STANDAR [71]

```
1       event Transfer(address indexed _from, address indexed _to, uint256
    indexed _tokenId);
2       event Approval(address indexed _owner, address indexed _approved,
    uint256 indexed _tokenId);
3       event ApprovalForAll(address indexed _owner, address indexed
    _operator, bool _approved);
```

Fig. 2.37. EVENTS IN ERC-721 STANDAR [71]

ERC-721 tokens features: [72].

- Each ERC-721 token get a name. The name field is used to indicate to contracts and external applications the name of the token.
- A symbol is defined that allows DApps to access an abbreviated name for these tokens.
- The total supply of the token is defined.
- The balance of tokens in an address is defined in a specific field.
- An owner function is defined for each ERC-721.
- ERC-721 tokens have a specific field called Owner, which ensures the token's non-fungibility and cryptographically identifies it.
- It features a property called Approval which is used for allowing an entity to transfer the token on the owner's behalf.
- Existance of field Takeover, use to allow users to own certain number of tokens and also allow users to withdraw them from another user's balance.
- Existance of Transfer field which allows tokens to be transmitted to another user in the same manner as bitcoin is sent, and provides information on which account sent the token and which account got it, as well as the token's ID.
- The Owner Token by Index feature was introduced due to the token's uniqueness and the fact that a person can hold many ERC-721 tokens. This feature allows tokens to be tracked using a unique ID.
- Existance of Token Metadata field which allows ERC-721 tokens to be non-fungible, and it contains all of the features that differenciate one token from another.
- Structure of the ERC-721 can not be modified.

3. OPENTIMESTAMPS PROTOCOL RESEARCH

3.1. Introduction

One of the main goal of the thesis, providing a research about a protocol that allows timestamping using decentralised trust systems backed by cryptographic techniques of hashing and verifying evidence as a proof of integrity, presents OpenTimestamps (OTS) protocol as the perfect candidate that fits the already mentioned requirements. OpenTimestamps is an open protocol that enables creating timestamps of data and validate existing timestamps for which proofs have been received using Bitcoin blockchain to provide security, trust and decentralization.

It's timestamping infrastructure with three main features described by its designer, Peter Todd:

Trust —*Trust feature is provided by the use of the Bitcoin blockchain as the network where the OTS protocol is developed*

Cost — *Scalability in the OTS protocol is infinite due to the possibility of combination of different number of timestamps in just one transaction in the Bitcoin network*

Convenience — *A thirhd party verifiable timestamp is provided so that the user can check inmediately the tiemstamps with no need of waiting for a Bitcoin transaction confirmation* [73]

It is important to analyze the OTS protocol in detail in order to be capable of providing objective and substantiated data on its reliability and accuracy. The aim of this chapter is to present and analyze in depth the objectives and technical performance of the OTS protocol and its implementation of timestamps on the Bitcoin blockchain in a context given by the previous chapters of this thesis on the fundamental aspects of this distributed network.

3.2. OTS protocol analysis

In an general overview the OTS protocol makes it possible to prove that a message existed prior to some point in time. The mechanism used to prove that data existed in the past by using the OTS protocol is the creation of timestamps stored on the Bitcoin blockchain, which by design is append-only and immutable, so that these timestamps can be independently verified at any point in time.

A list of commitment operations applied in sequence to the document which end with one or more time attestations is what a timestamp proof made by OpenTimestamps is made of. This list is distributed and organized in a merkle tree containing the document

as root of the tree, the commitment operations as the edges and time attestations as the leaves [74]. Verification of the proof can easily be done by replaying the operations made with the hashes representing each document of the list and checking that the final result or the final hash matches with the hash of the root of the merkle tree. Since commitment operations grant that different inputs will always result in different outputs and these inputs are outputs are hash functions computed by cryptographic algorithms that make them almost unique and very difficult to repeat, making the root hash unique thus making it almost impossible to change the original document without invalidating the proof.

3.2.1. Time attestations

For blocks on the Bitcoin blockchain to be accepted, the *nTime* field of their block header must be set to the time the block was approximately created. It is not necessary that the *nTime* field is set with absolute precision, for the purpose of the OTS protocol and its correct operation, it is sufficient that it coincides with a time interval of about 2 or 3 hours [73].

Due to the Bitcoin blockchain features of inmutability and the fact that it is designed to be an "append-only" data structure, Bitcoin can be considered a *notary* and the Bitcoin block headers can be used as *time attestations*: proof that certain data existed at some point in the past, confirmed by a notary we trust.

An example of a Bitcoin block header hash is provided in figure 3.1 in order to understand the next OTS protocol concepts.

```
02000000b96394585a281b7e5f438fd1c9ed492645a1fd61cb380204000000000000000007ee445
d23ad061af4a36b809501fab1ac4f2d7e7a739817dd0cbb7ec661b8a1e376755f58616186272def6
```

Fig. 3.1. BLOCK HEADER HASH [73]

3.2.2. Merkle trees

A Merkle hash tree, also known as a merkle tree or hash tree, is a tree-like data structure in which each non-leaf node is labeled with the hash of its child nodes' labels or values as it can be seen in figure 3.2 [31].

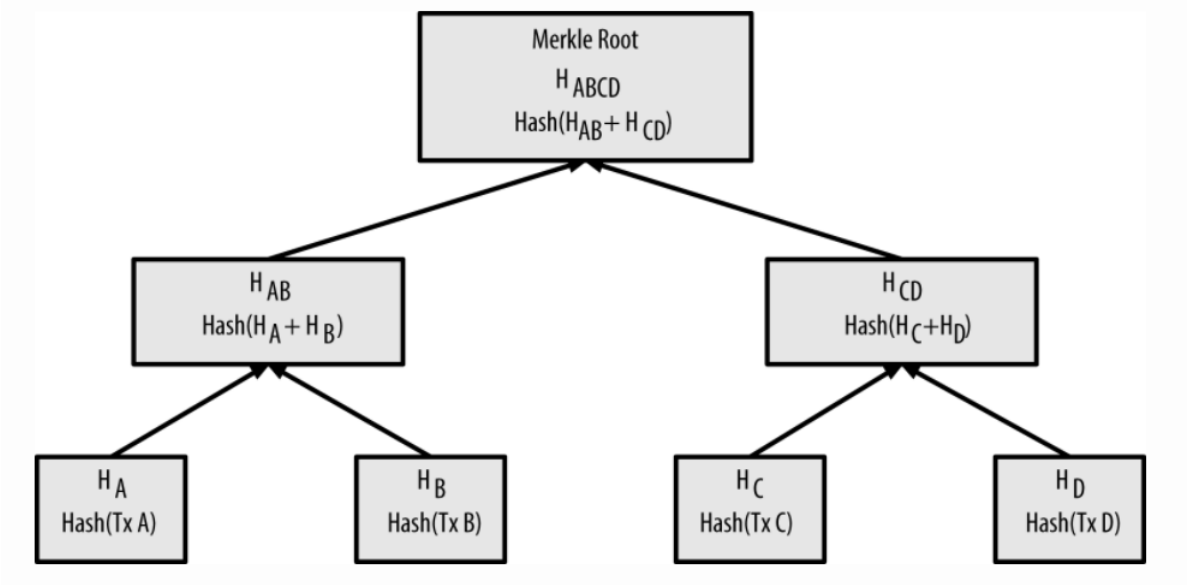


Fig. 3.2. MERKLE TREE [73]

The merkle root hash, which can be observed in figure 3.2, is made up as a concatenation of its children hashes thus the merkle root is used to link the transactions in a Bitcoin blockchain block with the hash header of the block.

The main reasons of using Merkle trees formed by hash functions is that cryptographic hash functions are able to match a variable-lenght message as input with a fixed-length output. The mechanism used by hash functions to generate those fixed-lenght messages involve such a enormous numbers that finding collisions is very difficult, it is not believed that finding collisions can be physically possible. Hash functions are generated using SHA256 algorithm.

"These numbers have nothing to do with the technology of the devices; they are the maximums that thermodynamics will allow. And they strongly imply that brute-force attacks against 256-bit keys will be infeasible until computers are built from something other than matter and occupy something other than space" [75].

Any attempt to modify "messages" that have already been timestamped on the Bitcoin blockchain will generate a different hash for themselves, since the message has been altered, and therefore one of the hash functions of the elements of the merkle tree is now different and therefore all the other hashes of the merkle tree formed as concatenations of its children nodes will also change, until reaching the merkle root, which will therefore be different.

3.2.3. Commitment operations

In line with what has been commented in the final part of the previous point, cryptographic hash functions can be described as commitment operation due to the fact of changing the imput will affect in changing the result.

The proofs provided by the OpenTimestamps protocol are collection of commitment operations which are applied to the message in sequence. Since every commitment operation will generate a different hash output for a different input the proof generated by the OTS protocol can be easily verify by knowing which is the list of opperations applied and reapplying the steps checking that the final result matches.

"The big advantage of representing a timestamp this way is it doesn't matter how you create the timestamp: so long as you can extract a valid list of commitment operations that the OpenTimestamps protocol supports, you can create a timestamp that any OpenTimestamps-compatible verifier can verify" [73].

Since commitment operations are ordered as a merkle tree, that means that timestamps are not a linear list of operations but a tree of operation, it allows a single message to be timestamped with multiple notaries.

3.3. Scalability

In most existing Bitcoin timestamping solutions it is needed to perform one transaction for each document you want to timestamp on the Bitcon blockchain, this means that if it is wanted to register 100,000 documents it would be needed have to perform 100,000 transactions which is quite inefficient and very expensive. However, the OpenTimestamps protocol offers a solution through which we can register those 100,000 documents with a single transaction. Since OTS organizes the desired documents to be timestamps into a merkle tree it is only needed to perform a single transaction with the merkle root of the tree to obtain a proof for each document that is contained in the merkle tree.

"Each per-file timestamp is simply the list of commitment operations that comprise the path up the first merkle tree, then up the Bitcoin block's merkle tree, to finally get to the block header. The verifier doesn't care: to it it's just a series of operations like any other timestamp" [73].

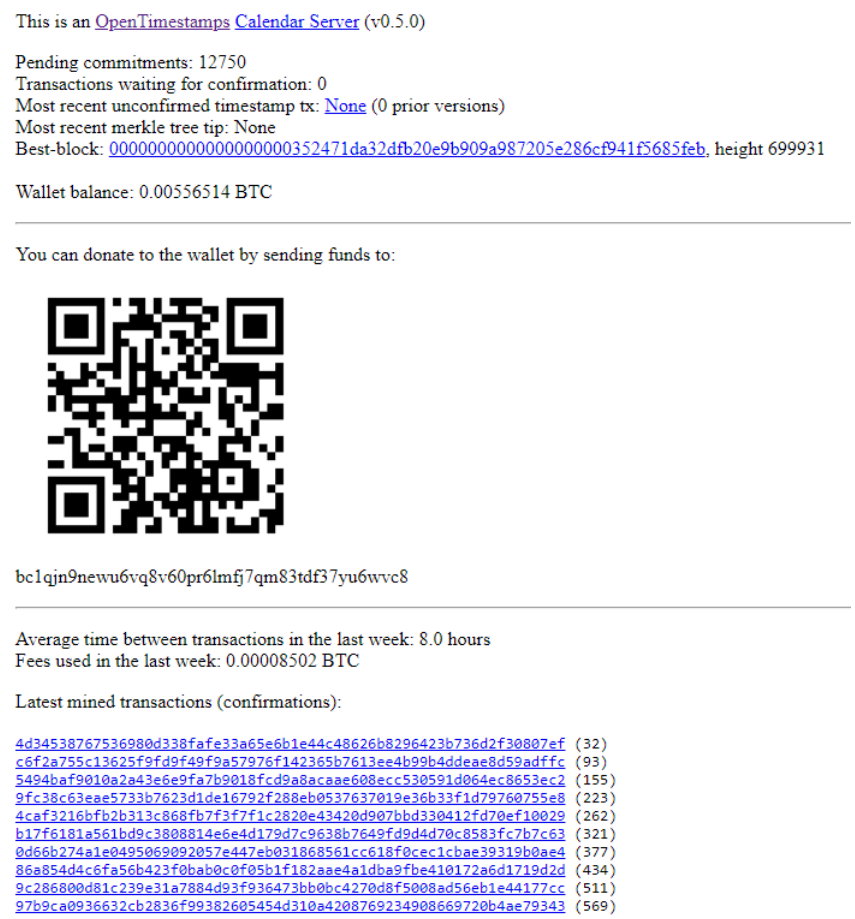
Although the design of the OTS protocol operation already makes it very efficient due to the distribution of documents in merkle trees, the OTS protocol offers some advantages that make the system better.

3.3.1. Aggregation servers

Aggregation servers is one of the improvements proposed by the OTS protocol. Aggregation servers are just public servers where users can submit a digest to be timestamped. The method of performance of this servers is that when a user submit one digest to be timestamped it is added to a list of pending digests. The submitted documents on the list are continously being distributed into different merkle trees with digests submitted by others users and then the merkle root of the created merkle tree is timestamped with Bitcoin. Individual timestamp proofs for each submitted digest are returned by the servers.

Aggregation server allows an unlimited number of digests to be timestamped every second without making any changes and using the same method for timestamp one single document or an unlimited number of them.

The aggregation server **a.pool.opentimestamps.org"** interface can be visualized in figure 3.3.


This is an OpenTimestamps Calendar Server (v0.5.0)

Pending commitments: 12750
Transactions waiting for confirmation: 0
Most recent unconfirmed timestamp tx: None (0 prior versions)
Most recent merkle tree tip: None
Best-block: 00000000000000000000352471da32dfb20e9b909a987205e286cf941f5685feb, height 699931

Wallet balance: 0.00556514 BTC

You can donate to the wallet by sending funds to:

bc1qjn9newu6vq8v60pr6lmfj7qm83tdf37yu6wvc8

Average time between transactions in the last week: 8.0 hours
Fees used in the last week: 0.00008502 BTC

Latest mined transactions (confirmations):

4d34538767536980d338fafe33a65e6b1e44c48626b8296423b736d2f30807ef (32)
c6f2a755c13625f9fd9f49f9a57976f142365b7613ee4b99b4ddeae8d59adffc (93)
5494baf9010a2a43e6e9fa7b9018fcd9a8acaae608ecc530591d064ec8653ec2 (155)
9fc38c63eae5733b7623d1de16792f288eb0537637019e36b33f1d79760755e8 (223)
4caf3216bfb2b313c868fb7f3f7f1c2820e43420d907bbd330412fd70ef10029 (262)
b17f6181a561bd9c3808814e6e4d179d7c9638b7649fd9d4d70c8583fc7b7c63 (321)
0d66b274a1e049506909205 7e447eb031868561cc618f0cec1cbae39319b0ae4 (377)
86a854d4c6fa56b423f0bab0c0f05b1f182aae4a1dba9fbe410172a6d1719d2d (434)
9c286800d81c239e31a7884d93f936473bb0bc4270d8f5008ad56eb1e44177cc (511)
97b9ca0936632cb2836f99382605454d310a4208769234908669720b4ae79343 (569)

Fig. 3.3. OTS AGGREGATION SERVER [73]

3.3.2. Public calendars

Using aggregation servers implies wait for the underlying Bitcoin transaction to be confirmed. Bitcoin transactions takes 10 minutes on average to be confirmed, sometimes it can be larger, therefore aggregation servers are efficient but not convenient.

In some cases, such as when sending a timestamped PGP-signed email or signing a timestamped Git commit, instantaneous timestamps are required. The OTS protocol incorporated public calendar servers to address the aforementioned issue. A calendar is nothing more than a collection of timestamps, and a calendar server allows you to view it from anywhere.

Calendar servers assure that the timestamping of each commitment added to the cal-

endar will be processed in a reasonable amount of time and that when the commitments get completed they will be made available to the public and kept indefinitely.

Therefore using OTS aggregation servers and public calendars huge a amount of timestamps can be processed quickly and in a conveniently way in about one second. Those timestamps can be later verify by anyone using the decentralized, trustless, Bitcoin blockchain.

4. TIMESTAMPING SERVICE DESIGN

4.1. Introduction

This chapter of the thesis will be used to propose and present a service design that will contribute to improve the usability that the final user can experience when deploying the opentimestamp protocol which will serve as a service to prove the existence of *'objects'* (documents, tweets, web elements) at a specific moment in time and in addition the content of these objects. In parallel, a functionality will be implemented to register the ethereum addresses that execute the timestamping of the objects and generate a unique token in the ERC-721 standard as irrefutable proof of possession of the timestamp of the object.

The chapter will be divided into a presentation and explanation of what the complete functionality of the proposed service is based on including a functional diagram and a more detailed analysis of the specific parts that make up the complete designed service.

4.2. Functional description

As described in the previous chapter of the thesis, on the analysis of the opentimestamp protocol technology, it can be deduced that it is a perfect, cheap and infinitely scalable protocol for the creation of decentralized and reliable timestamps. However, the timestamps created by the opentimestamp protocol can serve to prove and verify that the content data related through the timestamp hash, has existed at a specific time, but this relationship can only be established if we know the content of the data to which the timestamp hash refers. For example, the content of a tweet timestamped using the OTS protocol, or a webpage's content timestamped.

In order to provide a practical and enhanced timestamping service for the end user, it is proposed a model design in which when the user introduces an *'object'*, referring to object as any of the proposed information type, as input of the program this object is parsed info a *'node-type object'* containg some of its most important metada. Inmediately after the node is generated a hash string linked to the node is computed. By the computation of the node linked hash, the node's metadata become unalterable because any attemp of modifying the information would result in a different hash string.

The previous mentioned node-type objects and their metadata are being stored in a centralized data base. As centralized databases are susceptible to manipulation and the use of them involves risks and trust placed in third parties, which is wanted to be avoid, the database is managed to, in practical terms, behave as a decentralized one. When the user have decided to timestamp the whole database, contaning an infinity number of nodes,

a Merkle Tree Hash is computed using the previously node link hashes and the Merkle Tree Hash Root is resulted. Once the hash of the Merkle Root is computed, a *'.txt'* file is generated containing the hash of the Merkle Root this whole process in performed in local by the user and the resulted '.txt' file is inmediately timestamped by using the OpenTimestamp protocol in the Bitcoin blockchain. By the performance of the previously described process it is possible to timestamp a database containing an infinite scalable number of nodes because of the timestamp of just a file contaning a single string, the hash of the merkle root of the database nodes instead of the whole database containing an incredible high number of nodes and metadata. This process also has solved the problem of having a centralized database because once the hash of the merkle root is computed any modification of a single bit in any of the metadata of the nodes inside the database will result in a different merkle root hash. In practical terms this is as having a decentralized database because its impossible to alter the content. The already described process is a solution for achieven the objectives of 'object timestamping' and 'timestamp verficacion' proposed as goals of the thesis.

In parallel to the timestamping process described above, which is completely anonymous, a new process has been implemented thanks to which the user can register in a decentralized database, the Ethereum blockchain, that himself has been the user who has saved the object with information in the database, a record of his Ethereum address will remain right next to the link of the object entered. This way if necessary the user could compare the link from his address to the object with the object in the other 'centralized' database and its hash in order to prove that it has not been altered and that the owner of the Ethereum adress was the one who executed the object timestamp.

A process has also been implemented through which the user can decide if he wants to automatically generate a token in the ERC-721 standard, also known as an NFT, with the link to his timestamped object and the timestamp of the moment of the program execution. This process is linked with the aboved mentioned one and it is also developed over the Ethereum blockchain. The ERC-721 tokens besides demonstrating possession of an irrefutable and unalterable proof or evidence by their intrinsic properties can reach very high values depending on their rarity or importance, so the implementation of this service could be of interest to the user if he would like to capitalize on timestamps of valuable information as irrefutable evidences of them. This parallel process that has been already described allows to achieve the objectives in relation to ownership of timestamps and in real time generation of the ERC-721 token that proves this ownership proposed as objectives of the thesis.

The presentation of the proposed service will be divided into two parts. The first part will explain how the inclusion of data timestamps will be performed using the OTS protocol over the Bitcoin blockchain and the parallel process of the Ethereum database storage and the ERC-721 token generation and the second part will explain how it will be possible to verify the timestamps of tweets, web pages, documents and other source formats by the user, introducing object as input, not the hash of the merkle tree root

created with the concatenation of all the documents that belong to such a tree.

4.2.1. Object contents timestamping

On the design of the proposed service it is illustrated how it is possible to achieve a timestamping of information, including such information using the OTS protocol by using a database with hashed nodes and the computation of a merkle tree resulting in a timestamping of only the hash of the root of the tree.

The user Ethereum address and the link to the stored informacion process and NFT creation is also illustrated.

In figure 4.1 it is presented a graphic view of the design of the proposed service prototype, which will be discussed in more detail in the following sections of the thesis.

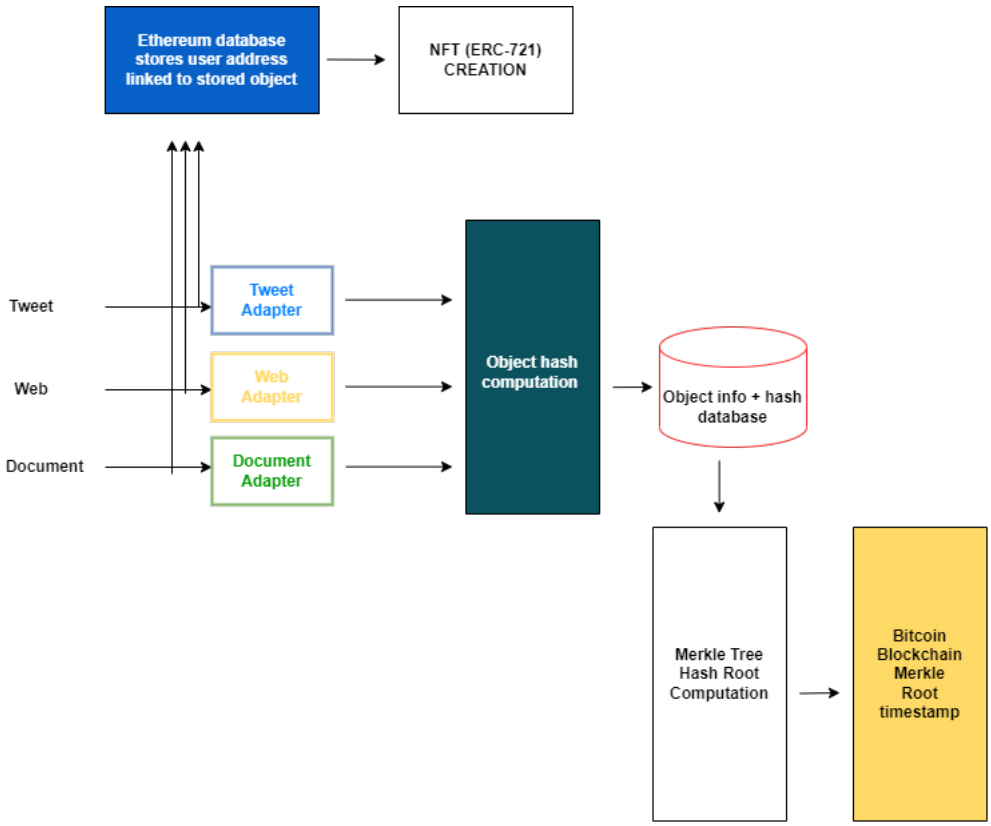


Fig. 4.1. SERVICE DESIGN DIAGRAM

4.2.1.1 Sources and adapters

The proposed service design allows the user to timestamp the content of the desired documents regardless of their format, from three proposed sources. Although only three sources of different formats, tweets, web pages and documents (PDFs), have been proposed for the service design, its is scalable to any source desired by the user.

This previously mentioned infinite scalability in terms of sources providing the desired contents to be timestamped by the user is feasible due to the *adapters* included in the diagram. The adapters are fundamental pieces for the performance of the service since they are in charge of parsing the information from the sources to objects that contain such relevant information in order to be included in a merkle tree structure.

4.2.1.2 Object hash computation and database storaged

After the file contents from the different sources have passed through their respective adapters and therefore objects containing the relevant information from the originating sources have been generated, these objects will be the nodes that will be stored on the database.

Before saving the database elements, the hash of each of these elements and their metadata is calculated. In this way the database is incorruptible because any modification of the metadata of the objects would also result in a modification of the hash of the node stored in the database and subsequently in a modification of the hash of the root of the merkle tree created with all the nodes stored in the database.

The node hash of, a tweet service implementation, is calculated using a string combining the timestamp of the text creation, the status id provided by the API from which the object metadata is obtained and the text contained in the text as shown in the next coding lines.

```
#Concatenation of tweet date + user_id + text to get an unique string to hash
string_status = str(status.created_at) + str(status.id) + str(status.text)
encoded = string_status.encode()
tweet_hash = hashlib.sha256(encoded)
```

4.2.1.3 Merkle Tree Hash Root computation and timestamping

Once the database has been populated with all nodes with their corresponding metadata and hashes, a Merkle Tree Hash is computed and the result is the root hash of the tree. After that, a *'.txt'* is generated and timestamped using the OpenTimestamp (OTS) protocol over the Bitcoin blockchain.

4.2.1.4 Ethereum database and ERC-721 generation

As described in the general function description chapter, the user have the chance of use a parallel process by which it is possible to store in an Ethereum database a link between its own Ethereum adress and the object introduced as input. This is possible by the use of a smart contract which once it is deployed it will never be erased nor modificated so the

user will always be able to demostrate that he was the author of the timestamping of the information or execution of the code.

The user will also have the chance of automatically generating an ERC-721 token that proves his ownency of the timestamp. The ERC-721 tokens besides demonstrating possession of an irrefutable and unalterable proof or evidence by their intrinsic properties can reach very high values depending on their rarity or importance, so the implementation of this service could be of interest to the user if he would like to capitalize on timestamps of valuable information as irrefutable evidences of them.

4.2.2. Timestamped object contents verification

Due to the inherent design of the proposed service and especially because of the external database where the merkle tree with the nodes that have the content of the files previously timestamped by the user is stored it is relatively easy for the user to verify the timestamp of a certain file by introducing it hash as input and obtaining the complete file, with its contents, as well as a verification evidence provided by the ots protocol itself.

In figure 4.2 it can be seen a diagram for the proposed service design when an user makes a file verification request.

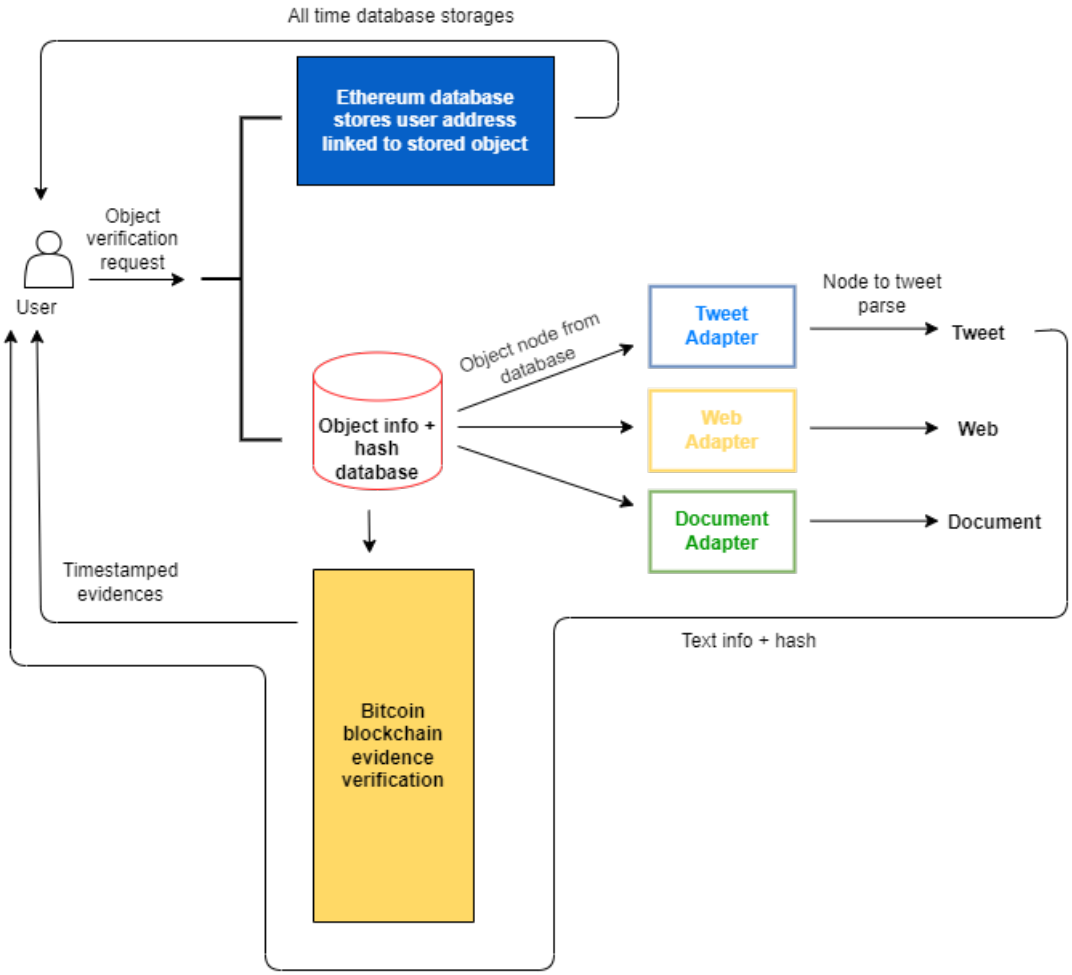


Fig. 4.2. SERVICE DESIGN DIAGRAM

The way the file verification mechanism would work would be, when a verification request has been made by the user, it will be proceeded to search for a node in the database within the auxiliary database that matches the user's input hash. If a node with these characteristics exists, a request will be made to the OTS protocol to get an evidence of its timestamp and at the meantime the node obtained from the merklee tree will be threaded through an adapter and the node will be parsed to a .json object containing the metadata of the original file, tweet, web page, PDF, etc. Subsequently the original file obtained and parsed along with the evidence of its timestamp in the Bitcoin blockchain will be returned to the user.

In order to verify that the user is the owner of the timestamp it is only needed to execute the Ethereum database reading function to show the all time storage inputs or just simply show the generated ERC-721 token.

5. SERVICE PROTOTYPE IMPLEMENTATION AND TESTING

This section which consist on two parts which the first one of them will be used to explained which modules of the proposed service are being developed and implemented and how these modules are enough for check the performance of the whole design and how the proposed goals of the thesis are achieved.

It will also be described the tests to be performed, their description, evaluation criteria and results after the implementation of the proposed service prototype described in last chapter of the thesis.

The entire project code can be found on Github. It is recomendable for the user to check the Github page and follow the instructions in order to execute the code.

Note that for implementation and testing an Ethereum address configured in Rinkeby testnet with some funds (eth) is required.

Github link: https://github.com/JMariadlcs/Indeliable-Evidences

5.1. Prototype implementation

In order to verify the performance of the proposed design, a propotype has been developed and implemented.

For the implementation of the propotype, all the modules described on the designed are being developed but just for the case of input data being in format of tweet URLs. This developed case of input data, tweet, is enough to prove the performance of the whole service design. In case that it was desired to develop and implement the proposed design in its totality, it was only be needed to develop the appropriate adapters corresponding to the different types of data that it was desired to be accepted as inputs. The rest of the prototype would remain the same.

In order to develop this module the only hardware needed was a laptot, a MacBook Pro was used. OpenTimestamps was the main Open Source software used and Python as programming language. Tweepy and Twitter for developers were used for the development of the tweets adapter.

As every described module is implemented, the final developed service is enough to observe that every objetive proposed for the thesis is fulfilled.

The locally executed part which consist on the adapters, the object hash computation, the generation of the database and the Merkle Tree Hash Root computation plus the execution of the OpenTimestamp commands over the Bitcoin blockchain allows the object timestamping and timestamping verification targets to be reached.

The parallel processes developed over the Ethereum blockchain, the decentralized database which is used for linking the user address with the timestamps already performed over the Bitcoin blockchain and the in real time ERC-721 token generation, which are managed by Smart Contracts, allos the prove of timestamping ownership target proposed as objetive of the thesis to be reached.

In order to develop the above mentioned module JavaScript to manage the user interface and Solidity for the Smart Contracts were used as programming languages. Node.js, hardhat, ethers.js, nomic labs, alchemy, open zepelin were the Open Source softwares used.

5.2. Tests

It will be performed different tests to check if every proposed goal of the thesis is achieved and every module is performing correctly.

The tests will be divided into two groups, it will first be tested the Timestamping over Bitcoin blockchain process and then the Ethereum database implementation and ERC-721 token generation process.

5.2.1. Timestamping over Bitcoin blockchain

The main goal of this section of the thesis is to test and show the results from the execution of the timestamping feature from the proposed service over thee Bitcoin blockchain by using the OpenTimestamp protocol.

5.2.1.1 Object storing in database + hashing

In this test, the functionallity of storing an input object, in the implemented case its a tweet url, in the generated .csv database and also the object hashing.

Evaluation criteria

The following criteria is the one followed for the testing results:

- The object is correctly managed by introducing an URL by input.
- The object is correctly parsed from .json format to .csv database.
- Hashing of the object is correctly performed.

Results

The obtained results after execution are shown in the following figures. In figure 5.1 the execution of the command used for adding the tweet as input to the database is shown.

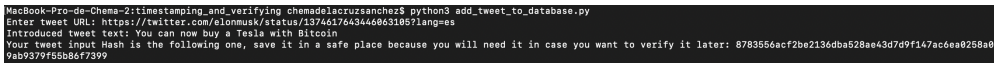


Fig. 5.1. ADDING TWEET TO DATABASE COMMAND EXECUTION

In figure 5.2 the creation of the database with all the metadata pased from the json object from the input tweet plus a hash creation and asignation to the hashed tweet is shown.

Fig. 5.2. DATABASE GENERATION RESULT

As it can be observed, a .csv database is generated containing the hash of the input tweet and all of the its most important metadata so expectations of the results have been successfully met.

5.2.1.2 Database timestamping

In this section of the thesis the database timestamping feature is going to be tested.

Evaluation criteria

The following criteria is the one followed for the testing results:

- Computation of the hash of the database merkle tree root.
- Generation of .txt file containing a string with the root hash.
- OpenTimestamp command execution for the .txt file generated.

Results

The obtained results after execution are shown in the following figures. In figure 5.3 the generation of a .txt file containing the hash of the merkle root and the execution of the ots command for the mentioned file is shown.

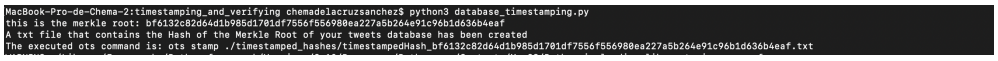


Fig. 5.3. DATABASE TIMESTAMPINNG COMMAND

In figure 5.4 the mentioned .txt file containing the merkle root hash is shown.

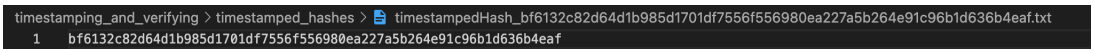


Fig. 5.4. .TXT FILE WITH DATABASE HASH GENERATION

As it can be observed the expected .txt file is correctly generated containing the merkle root hash of the database that become timestamped by the execution of the OpenTimestamp (OTS) command.

5.2.1.3 Timestamp verification

In this section of the thesis the verification of the previously timestamped database is going to be tested.

Evaluation criteria

The following criteria is the one followed for the testing results:

- The OpenTimestamp verification command is correctly executed.
- A Json object is generated containing all the metadata and hash of the request tweet verification.

Results

The obtained results after execution are shown in the following figures.

In figure 5.5 the introducion of the tweet hash as input and the OpenTimestamp verification command execution is shown.

Fig. 5.5. VERIFY TWEET EXECUTION

In figure 5.6 the generation of a .json object containing the hash and metadata of the input tweet is shown.

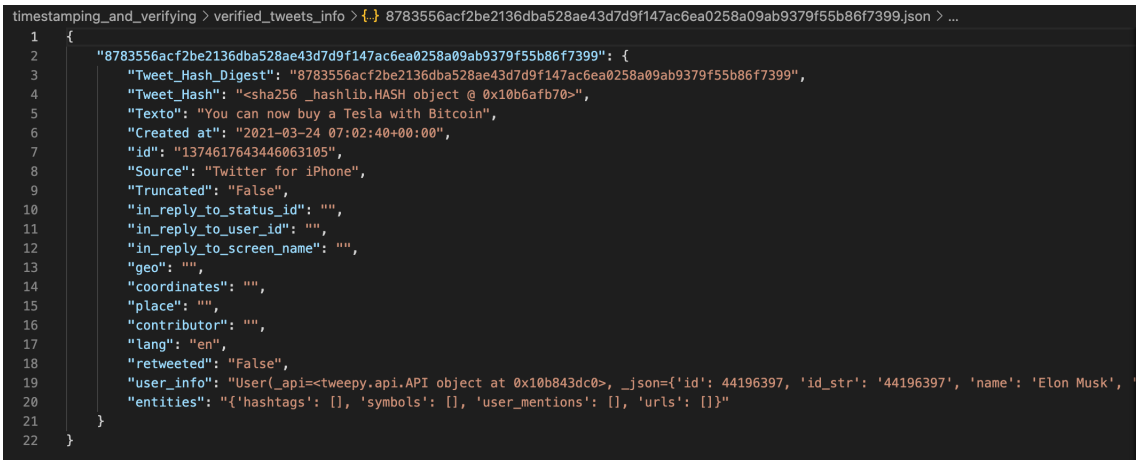


Fig. 5.6. TWEET METADATA AS .JSON OBJECT GENERATION

As it can be seen, the verification OpenTimestamp command is correctly executed by introducing the Hash of the desired verfication tweet and the .json object containing the metadata and hash of the tweet is also shown.

5.2.2. Ethereum database feature and ERC-721 token generation

This section of the thesis is going to be used to show the testing results from the Ethereum database process and the ERC-721 token generation.

5.2.2.1 Ethereum database storage

In this section the Ethereum database parallel process where the user address is being linked with the tweet url is tested.

Evaluation criteria

The following criteria is the one followed for the testing results:

- The input tweet is processed correctly.
- The smart contract function is correctly executed.
- The block in the Ethereum blockchain is correctly mined.

Results

The obtained results after execution are shown in the following figures.

In figure 5.7 the input tweet can be seen as correctly managed and the storing of Ethereum database + ERC-721 (mint EvidenceNFT) execution is performed.

```
MacBook-Pro-de-Chema-2:src chemadelacruzsanchez$ node --experimental-json-modules App.mjs
(node:24245) ExperimentalWarning: Importing JSON modules is an experimental feature. This feature could change at any time
(Use `node --trace-warnings ...` to show where the warning was created)
Do you want to:
 -Store an evidence in database (type: '1')
 -Store an evidence in database + mint EvidenceNFT (type: '2')
 -Show all stored evidences (type: '3').
2
Introduce evidence URL you want to save in database + EvidenceNFT mint (make sure url does exist at this moment, otherwise wrong url will be
saved but wont serve as a proof for you in case you need it in the future
https://twitter.com/elonmusk/status/1374617643446063105?lang=es*/
Evidence URL introduced by user: https://twitter.com/elonmusk/status/1374617643446063105?lang=es*/
```

Fig. 5.7. ETHEREUM DATABASE AND ERC-721 EXECUTION

In figures 5.8 and 5.9 the metadata from block mining is correctly processed and shown.

```
Mining block...
{
  to: '0x1aAA08e2a582D47B34963FF29F8A74b2027669D4',
  from: '0x0aEbeBee37D530961e05FF525409801Ab97341dE',
  contractAddress: null,
  transactionIndex: 13,
  gasUsed: BigNumber { _hex: '0x02be85', _isBigNumber: true },
  logsBloom: '0x00000000000200000000100000000000000000000000000000020000000000000000000000000000000000000000000000200000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000004000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000004000000000000000000000000000000000000000000000000002000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000001000000008000000',
  blockHash: '0x301664e3ace5888699007475004747a725095798d44bb805b1593fe77f1169ed',
  transactionHash: '0x9f9da8be0b2b3b848eb67612cf225dd036c89cfbfa01c9f86a8910988dc851b2',
  logs: [
    {
      transactionIndex: 13,
      blockNumber: 9945223,
      transactionHash: '0x9f9da8be0b2b3b848eb67612cf225dd036c89cfbfa01c9f86a8910988dc851b2',
      address: '0x1aAA08e2a582D47B34963FF29F8A74b2027669D4',
      topics: [Array],
      data: '0x0000000000000000000000000000000000000000000000000000000061d72a160000000000000000000000000000000000000000000000000000000000000
04000000000000000000000000000000000000000000000000000000000000000416874747470733a2f2f747769747465722e636f6d2f656c6f6e6d75736b2f7374617475732f3
13337343631373634333434363036333130353f6c616e673d65732a2f00000000000000000000000000000000000000000000000000000000000000',
      logIndex: 13,
      blockHash: '0x301664e3ace5888699007475004747a725095798d44bb805b1593fe77f1169ed'
    }
  ],
  blockNumber: 9945223,
  confirmations: 1,
  cumulativeGasUsed: BigNumber { _hex: '0x0fb68e', _isBigNumber: true },
  effectiveGasPrice: BigNumber { _hex: '0x950807c7', _isBigNumber: true },
  status: 1,
  type: 2,
  byzantium: true,
  events: [
    {
      transactionIndex: 13,
      blockNumber: 9945223,
      transactionHash: '0x9f9da8be0b2b3b848eb67612cf225dd036c89cfbfa01c9f86a8910988dc851b2',
      address: '0x1aAA08e2a582D47B34963FF29F8A74b2027669D4',
      topics: [Array],
      data: '0x0000000000000000000000000000000000000000000000000000000061d72a160000000000000000000000000000000000000000000000000000000000000
04000000000000000000000000000000000000000000000000000000000000000416874747470733a2f2f747769747465722e636f6d2f656c6f6e6d75736b2f7374617475732f3
13337343631373634333434363036333130353f6c616e673d65732a2f00000000000000000000000000000000000000000000000000000000000000',
      logIndex: 13,
      blockHash: '0x301664e3ace5888699007475004747a725095798d44bb805b1593fe77f1169ed',
      args: [Array],
      decode: [Function (anonymous)],
      event: 'NewTweetStored',
      eventSignature: 'NewTweetStored(address,uint256,string)',
```

Fig. 5.8. BLOCK MINED METADATA PART 1

```
      logIndex: 13,
      blockHash: '0x301664e3ace5888699007475004747a725095798d44bb805b1593fe77f1169ed',
      args: [Array],
      decode: [Function (anonymous)],
      event: 'NewTweetStored',
      eventSignature: 'NewTweetStored(address,uint256,string)',
      removeListener: [Function (anonymous)],
      getBlock: [Function (anonymous)],
      getTransaction: [Function (anonymous)],
      getTransactionReceipt: [Function (anonymous)]
    }
  ]
}
💥Your Evidence is already stored on the database.
```

Fig. 5.9. BLOCK MINED METADATA PART 2

As it has been shown in last figures, the Ethereum blocks are correctly mined and their metadata is shown, therefore the function from the smart contract is correctly executed.

The smart contract execution for the Ethereum database storage function can be seen using Etherscan.io in Rinkeby mode: https://rinkeby.etherscan.io/tx/0x6a3e72436eaf1870b588d84b2169049ba90b27a8c4314f59414875ab53c8d39d

5.2.2.2 Ethereum database verification

In this section a verification of all time stored tweets on the Ethereum database is going to be tested.

Evaluation criteria

The following criteria is the one followed for the testing results:

- The all time stored tweets on the Ethereum database are shown.

Results

The obtained results after execution are shown in the following figures.

In figure 5.10 is shown all the tweets stored on the Ethereum database since the smart contract was deployed.

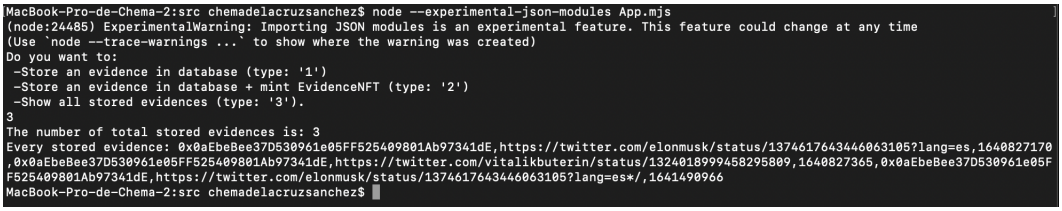


Fig. 5.10. ETHEREUM DATABASE VERIFICATION

As it can be seen, all the tweets stored on the Ethereum database are correctly shown. In figure 5.10 could be observe 3 tweets, this are tweets used for testing since anything can be deleted or change from a smart contract once it has been deployed or any of its functions executed.

5.2.2.3 ERC-721 token generation

In this section of the thesis the generation of the ERC-721 token once the user has introduce the tweet URL is correctly done is going to be tested.

Evaluation criteria

The following criteria is the one followed for the testing results:

- The Ethereum block is correctly mined.
- The ERC-721 token is correctly generated.

Results

The obtained results after execution are shown in the following figures.

In figures 5.11 and 5.12 the block mining and its metadata are shown.

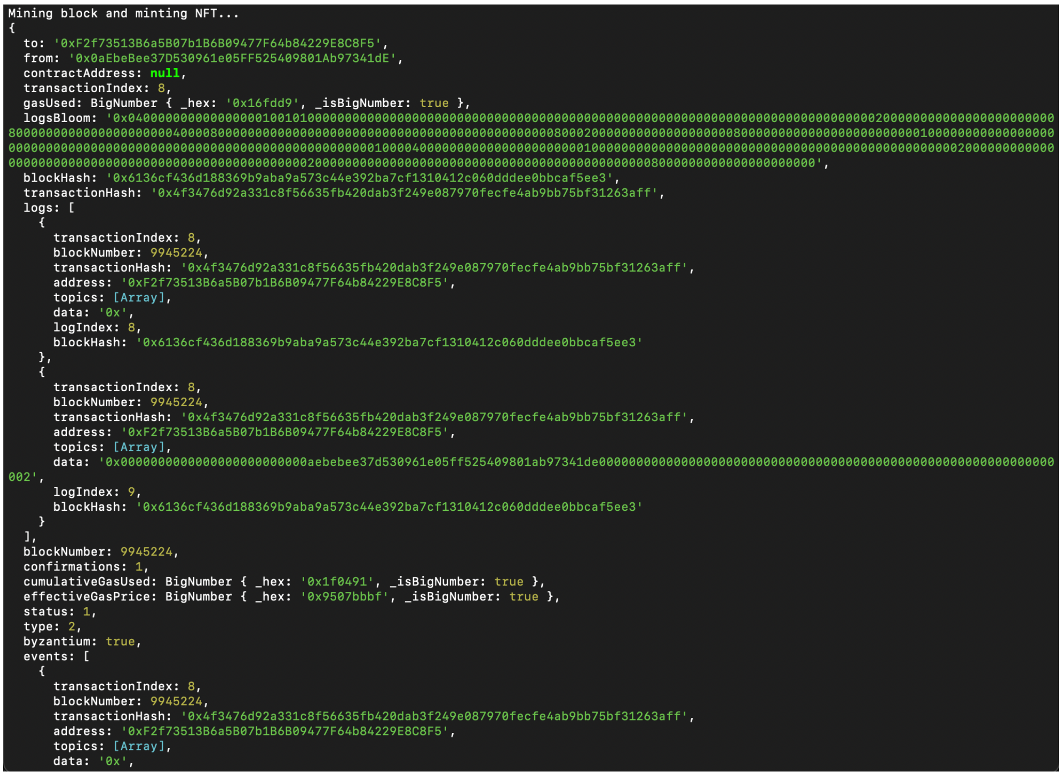


Fig. 5.11. ERC-721 BLOCK METADATA PART 1

Fig. 5.12. ERC-721 BLOCK METADATA PART 2

As it can be shown in previously figures the Ethereum block is correctly mined and its metadata is shown. The ERC-721 token is also correctly generated.

The smart contract execution for the ERC-721 token can be seen using Etherscan.io in Rinkeby mode: https://rinkeby.etherscan.io/tx/0x735654596712f5f675f77a48174462f492efc73ac3c729837ca3a4ae1178b9fb

To display the ERC-721 token, Opensea can be used, note that our contract and tokens are deployed and generated on Rinkeby network, a testnet on the Ethereum blockchain. Therefore the Rinkeby version from Rarible must be used to correctly display the generated ERC-721 token.

As it can be seen in last figure the ERC-721 token image is an on real-time and on-chain computation of the url input and timestamp. The description of the token also contains this data with an unique ERC-721 identifier.

In figure 5.13 the ERC-721 token generated during the test example is provided by Opensea.

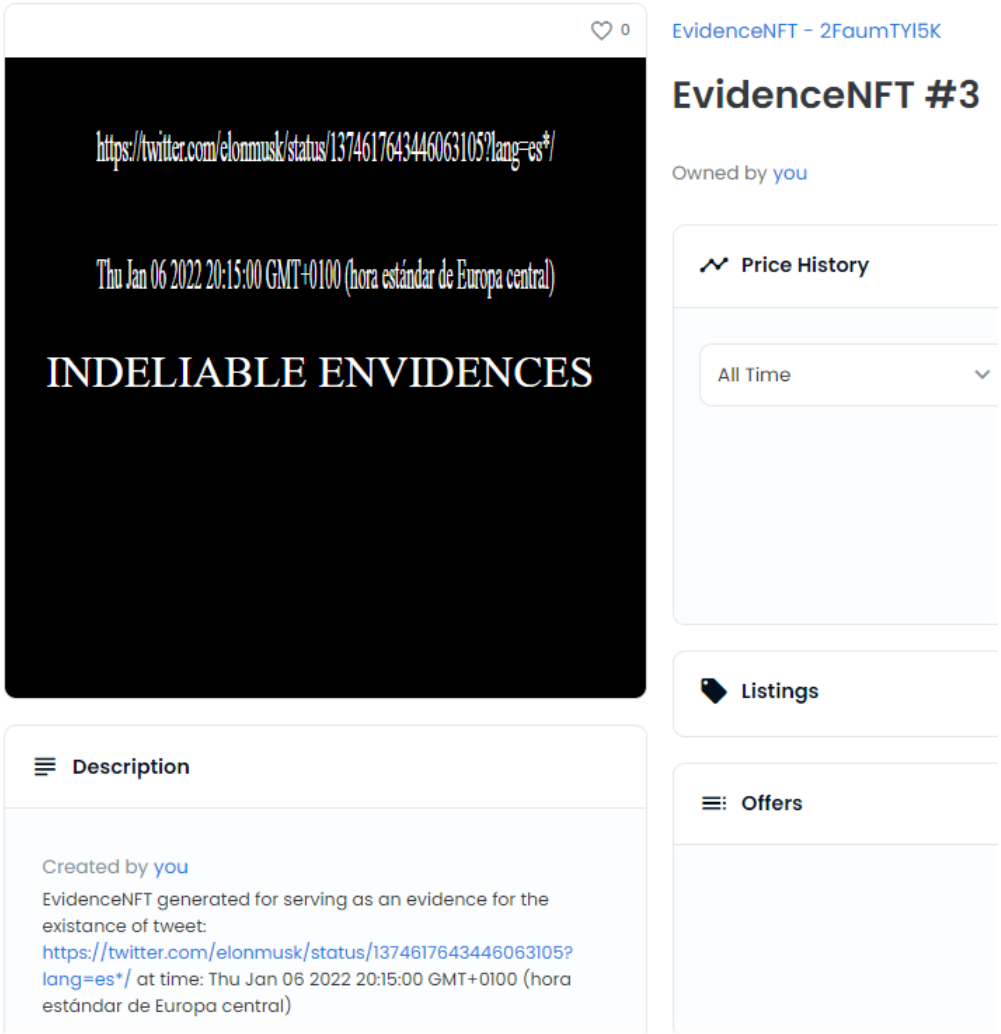


Fig. 5.13. ERC-721 TOKEN SHOWN IN OPENSEA

Also a link to the ERC-721 token display in Opensea on Rinkeby testnet is provided: https://testnets.opensea.io/assets/

0xf2f73513b6a5b07b1b6b09477f64b84229e8c8f5/3

In figure 5.14 the ERC-721 token generated during the test example is provided by Rarible.

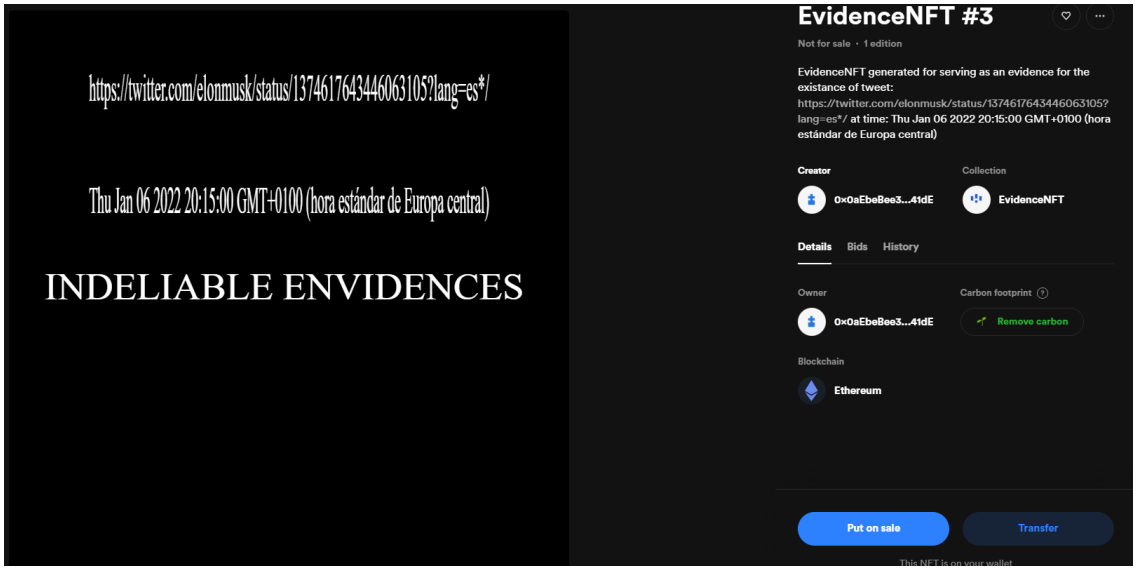


Fig. 5.14. ERC-721 TOKEN SHOWN IN RARIBLE

Also a link to the ERC-721 token display in Rarible on Rinkeby testnet is provided: https://rinkeby.rarible.com/token/0xf2f73513b6a5b07b1b6b09477f64b84229e8c8f5:3?tab=details

6. CONCLUSIONS AND FUTURE WORK

As mentioned in Chaper 1, the main goal of this thesis was to analyze the OpenTimestamp protocol and propose a decentralized service that improves the user's possibilities when it comes to the use of the OpenTimestamp protocol with the addition of a parallel process which add some useful features.

In order to reach the conclusion of the proposed final service prototype design it has been necessary not only to understand and analyze in depth, in a technological way, the timestamp protocol but also how the two blockchain networks work on which we have directed our processes that work in parallel and make up the final service prototype, the Bitcoin blockchain and the Ethereum blockchain networks.

6.1. Conclusions

The main goal of the whole prototype was to provide a fully decentralized behaviour service for object timestamping. After running the code and execute all the processes, the result of the final prototype performance is the one expected.

Reviewing the specific objectives proposed in the first section of the thesis whose fulfillment is necessary to achieve the main objective it can be seen that after the implementation of the prototype modules and the questions asked with satisfactory results have been met.

By the implemenation of the whole first module and the performance of the tests object timestamps, tweets in the implemented code, are perfectly executed using the OpenTimestamps protocol and verification of the timestamps also works perfectly. Verification of previously timestamped objects has been successfully achieved as a result of the database implementation which due to the hashing of the stored nodes in the database and the merkle tree hash computation lead to an incorruptible database in which any change on it could be easy notice because of the totally difference merkle root hash resultant.

By the implementation of the second module with the development of a parallel process over the Ethereum blockchain, the objetives of creation of an ownership protocol that links the timestamps generated by a user to his own Ethereum address to provide evidence that he was the originator of the timestamp and an in real time generation of ERC-721 token for the timestamp evidence generated by the user are succesfully achieved.

Overall, the study of the technology on which our prototype has been built, the blockchain networks, specifically Bitcoin and Ethereum and the study and use of the OpenTimestamp protocol and the implementation of our service prototype on the proposed design and the tests performed to prove its correct performance results in all the objectives proposed for the thesis being satisfactorily achieved.

6.2. Project vision

OpenTimestamps protocol implementation allows the user to make object timestamps in an anonimous way, which may prove to be the preferred option in some of the cases. The implementation of a parallel process as the one in our prototype by the use of smart contracts in the Ethereum blockchain results in a fully decentralized database where the object stored are directly linked to the Ethereum address of the user that executes the program. In certain situations, the user may want to be linked as the author of a timestamp execution of a specific object, in these situations the implemented prototype fits perfectly. If the user want to be linked as the author of the timestamp he must use the ots protocol and store the object in the database with the generated hash and also execute the parallel process with the link, in case of tweet timestamping, of the timestamped object, the link of the tweet stored in the Ethereum database will match with the one in the hashed database. The Ethereum database is not possible to become altered because of the smart contract implementation and the hash of the other process database will change in case of any alteration, that is how the user will be linked to the timestamp.

As blockchain technology, cryptocurrencies, decentralized applications and in general web3, decentralized web, are becoming increasingly relevant and popular, there are an infinite number of new opportunities and new business models are emerging in which there exist a multitude of ways to capitalize among them, NFTs are becoming very popular and some of them also very valuable. This is an added reason for the implementation of the token autogeneration in the ERC-721 standard of our service prototype, which in addition to providing ownership and uniqueness can result in a tool for future implementations of business ideas, such as the sale of the most viral timestamped tweets that have been tweeted by mistake and subsequently deleted and have passed into history, or maybe just the timestamping and NFT generation of the most generational important tweets as for example the tweet of Vitalik Buterin presenting the Ethereum whitepaper.

6.3. Future work

As it can be seen in the service design section, the proposed prototype includes tweet, web and document possibilities of timestamping with their corresponding adapters but it has been only implemented the one related with tweets timestamping to prove the performance of the prototype. For the future work it could be possible the implementation of the web and document adapters to make possible the timestamping of information from these sources also. The implementation of a frontend code with a good user interface through which the code can be executed would also be a good idea as a future implementation. The best way to carry out this implementation would be through a logging with a decentralized ethereum wallet such as Metamask, for our project to be fully focused on web3, it could be done with Javascript using React and Node.js. It could be used a server with the OpenTimestamp protocol installed so that the user can make use of our prototype at

any time.

The implemented service could also be used as a backend tool for the implementation of different business paths ort projects. For example, it could be the implementation of different frontends depending on each idea that can take advantage of the proposed backend service. An example could be the creation of a type of service that consists of making a copy of the most viral tweets, most important or any other proposed type or idea where users would perform by hand the timestamp of these tweets which would receive native tokens (ERC-20) generated as a reward, in this way it could be created a kind of decentralized social network with tweets chosen depending on some imposed conditions. It could be something similar to the IPFS protocol that a decentralized database but only of tweets. This is just one example among many others that we could think of or that anyone could think of using our service as a backend tool.

7. SOCIOECONOMIC AND PROJECT MANAGEMENT

7.1. Socioeconomic impact

The last decade has been socially developed by the hand of emerging and overdevelopment of traditional technology companies. The TIC and software sector has been one of the main ones in terms of technological development in this decade and has directly influenced the social development.

During the course of the last decade technological companies based on the web2.0 have emerged and although the origin of cryptocurrencies corresponds to the last decade with the whitepaper of Satoshi Nakamoto, the explosion of blockchain technology is arriving in the last years of the decade and the first years of the new one we are going through. With this peak of blockchain technology a whole new web is beginning to flourish, the web3.0, where the business opportunities that are emerging and the infinite technological ideas still unimaginable thanks to the intrinsic characteristics of this new web3.0 such as decentralization, security and ownership of data by the user, is more than remarkable.

As seen in next figure [76], the total cryptocurrencies market capitalizacion reached a maximum of 1.99 billions of dollars on the 31 March, and with them the development of blockchain technology which seems to continue growing in the next years.

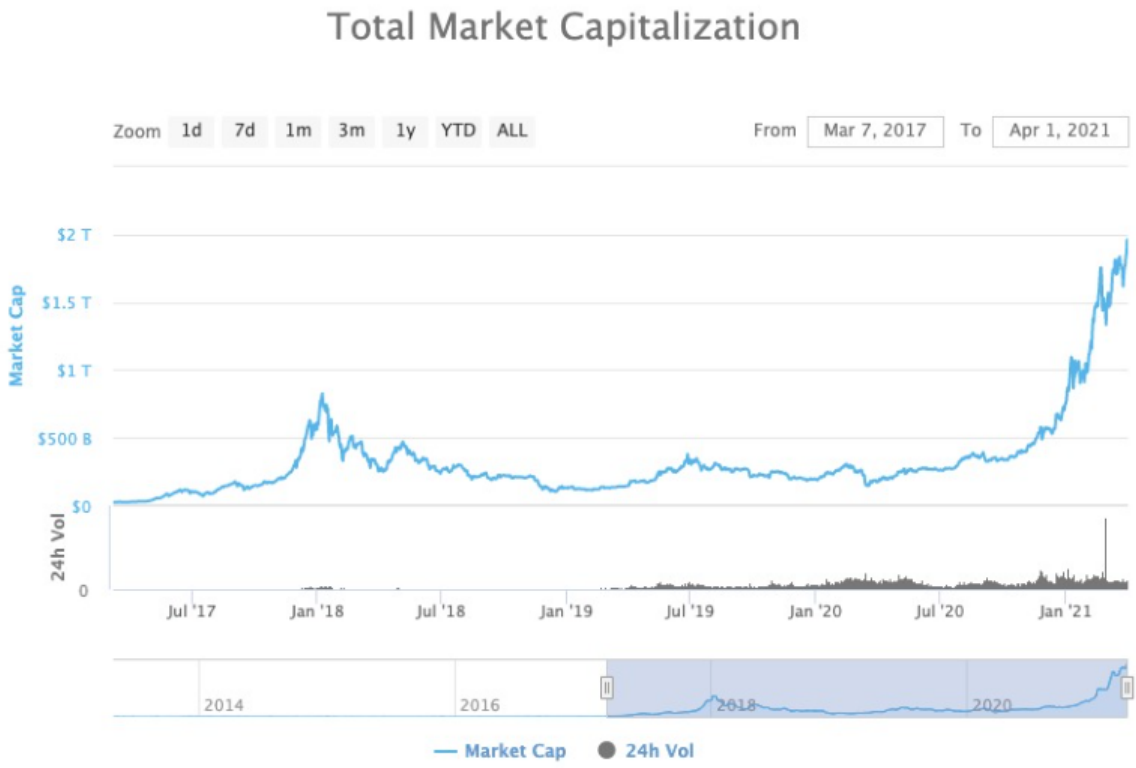


Fig. 7.1. CRYPTOCURRENCIES MARKET CAPITALIZATION [76]

7.2. Project management and budget

This section of the thesis is developed following the guidelines from the UNE-ISO 21500:2013 [77] and the Project Management Body of Knowledge [78], which result very effective tools for project management.

7.2.1. Management

In the table 6.1 it is shown the number of hours used for the development of the thesis.

ID	Task	Hours
A	Subject of the thesis and tutor searching	10
B	Search and study the state of the art in the filed of blockchain technology: Bitcoin blockchain, Ethereum blockchain and OpenTimestamp protocol	40
C	Installation, execution and working understanding of OpenTimestamp protocol	15
D	Development of object timestamping by user input program over Bitcoin blockchain	20
E	Development of Ethereum database and ERC-721 token generation	30
F	Thesis report writting by following the guidelines provided by University Carlos III of Madrid	100
G	Presentation creation for thesis defending in front of the evaluation court	30
	Total	245

Table 7.1. PERFORMED TASKS AND DURATION

In the following figure a Gantt diagram is provided with the overview of the activities performed.

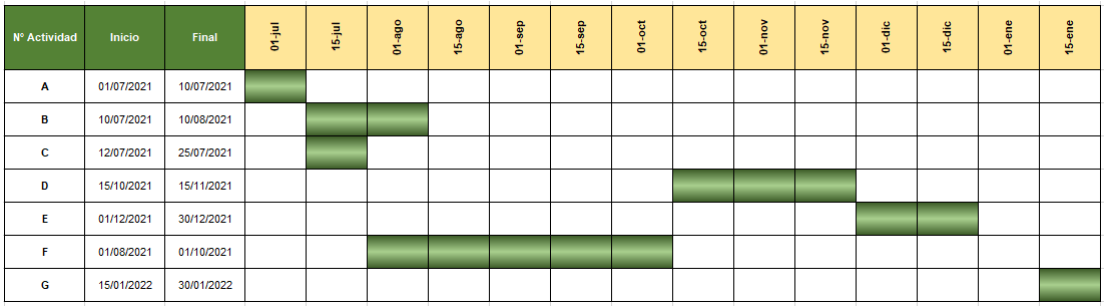


Fig. 7.2. GRANT DIAGRAM

7.2.2. Budget

For the budget calculation of the project the materials costs, human resources and indirect below detailed costs are taking into account.

Material costs include all the recources that has been used to the development of the proposed service in the thesis.

- In terms of sotfware, Python, JavaScript and Solidity has been used which have an open-source license. For the writting of the thesis Overleaf has been used which also has an open-source licence. OpenTimestamp protocol has been used,framewors like Visual Studio Code and workin shells like Hardhat, Node.js which count with open-source licences.
- In terms of hardware, a MacBook Pro 13" was used for the development of the project. These costs have been detailed in table 6.2.
- In terms of human resources, costs are being calculated taking into account the working hours by the people involved in the project and thesis development: Daniel Díaz Sánchez, thesis supervisor, and the author, as engineer. These costs are specified in table 6.3

Product	**Cost**	**Time of use (months)**	**Total cost**
MacBook Pro 13"	1500€	7	214.29€
		Total	214.29€

Table 7.2. MATERIAL COSTS

Name	**Charge**	**Salary (€/hour)**	**Hours**	**Total cost**
Proffesor Daniel Díaz Sánchez	Senior Engineer	45	10	450€
Jose María de la Cruz Sánchez	Junior Engineer	25	245	6,125€
			Total	6,575€

Table 7.3. HUMAN RESOURCE COSTS

A computation of the total mentioned above cost is shown in table 6.4

Cost name	**Cost**
Cost of Material	214.29€
Cost of HR	6,575€
Total	6,789.29€

Table 7.4. TOTAL COSTS

BIBLIOGRAPHY

[1] T. C. Earle, "Trust, confidence, and the 2008 global financial crisis," *Risk Analysis: An International Journal*, vol. 29, no. 6, pp. 785–792, 2009.

[2] G. Greenwald, "Xkeyscore: Nsa tool collects' nearly everything a user does on the internet'," *The Guardian*, vol. 31, p. 2013, 2013.

[3] M. Marquis-Boire, G. Greenwald, and M. Lee, "Xkeyscore," *The Intercept*, vol. 1, 2015.

[4] K. O'Flaherty, "This is why people no longer trust google and facebook with their data," *Retrieved from Forbes. com: https://www.forbes.com/sites/kateoflahertyuk/2018/10/10/this-is-why-people-no-longer-trust-google-and-facebook-with-their-data*, 2018.

[5] L. Lamport, R. Shostak, and M. Pease, "The byzantine generals problem," in *Concurrency: the Works of Leslie Lamport*, 2019, pp. 203–226.

[6] S. Nakamoto, "Bitcoin: A peer-to-peer electronic cash system," *Decentralized Business Review*, p. 21 260, 2008.

[7] I. Bashir, *Mastering blockchain*. Packt Publishing Ltd, 2017.

[8] M. Van Steen and A. S. Tanenbaum, *Distributed systems*. Maarten van Steen Leiden, The Netherlands, 2017.

[9] S. Tarkoma, *Overlay Networks: Toward Information Networking*. CRC Press, 2010.

[10] S. R. Mantena, "Transparency in distributed systems," *CSE*, vol. 6306, p. 13,

[11] W. Diffie and M. Hellman, "New directions in cryptography," *IEEE transactions on Information Theory*, vol. 22, no. 6, pp. 644–654, 1976.

[12] *Historia de la tecnología blockchain: Guía definitiva*, Dec. 2018. [Online]. Available: https://101blockchains.com/es/historia-de-la-blockchain/.

[13] S. Haber and W. S. Stornetta, "How to time-stamp a digital document," in *Conference on the Theory and Application of Cryptography*, Springer, 1990, pp. 437–455.

[14] *Adam back*, Jun. 2021. [Online]. Available: https://en.wikipedia.org/wiki/Adam_Back.

[15] A. Preukschat and A. P. por la continua transformación social propiciada por la tecnología y de la nueva economía P2P. Desde 2012 es asesor de desarrollo estratégico, *Hashcash*, Jan. 2019. [Online]. Available: https://libroblockchain.com/hashcash/.

[16] W. Dai, *bmoney*, 1998. [Online]. Available: http://www.weidai.com/bmoney.txt.

[17] ——, *Wei Dai Homepage*, 1998. [Online]. Available: http://www.weidai.com/.

[18] N. Reiff, *B-money*, May 2021. [Online]. Available: https://www.investopedia.com/terms/b/bmoney.asp.

[19] J. Frankenfield, *What is a wei?* Jul. 2021. [Online]. Available: https://www.investopedia.com/terms/w/wei.asp.

[20] N. Szabo, "Bit gold proposal," *Decentralized Business Review*, p. 21 449, 2008.

[21] K. Raj, *Foundations of blockchain: the pathway to cryptocurrencies and decentralized blockchain applications*. Packt Publishing Ltd, 2019.

[22] M. Kolvart, M. Poola, and A. Rull, "Smart contracts," in *The Future of Law and etechnologies*, Springer, 2016, pp. 133–147.

[23] *Introduction to smart contracts*. [Online]. Available: https://ethereum.org/en/developers/docs/smart-contracts/.

[24] B. Singhal, G. Dhameja, and P. S. Panda, *Beginning Blockchain: A Beginner's guide to building Blockchain solutions*. Springer, 2018.

[25] P. Hooda, *Comparison - centralized, decentralized and distributed systems*, Apr. 2019. [Online]. Available: https://www.geeksforgeeks.org/comparison-centralized-decentralized-and-distributed-systems/.

[26] E. Rutland, "Blockchain byte," *FINRA. R3 Research*, p. 2, 2017.

[27] D. Xiao, *The four layers of the blockchain*, Jul. 2017. [Online]. Available: https://medium.com/@coriacetic/the-four-layers-of-the-blockchain-dc1376efa10f.

[28] J. Eberhardt and J. Heiss, "Off-chaining models and approaches to off-chain computations," in *Proceedings of the 2nd Workshop on Scalable and Resilient Infrastructures for Distributed Ledgers*, 2018, pp. 7–12.

[29] J. Frankenfield, *Off-chain transactions (cryptocurrency) definition*, Aug. 2021. [Online]. Available: https://www.investopedia.com/terms/o/offchain-transactions-cryptocurrency.asp.

[30] S. Gupta, S. Sinha, and B. Bhushan, "Emergence of blockchain technology: Fundamentals, working and its various implementations," in *Proceedings of the International Conference on Innovative Computing & Communications (ICICC)*, 2020.

[31] A. M. Antonopoulos, "Mastering bitcoin," 2019.

[32] Sheinix, *The bitcoin network*, Oct. 2020. [Online]. Available: https://medium.com/coinmonks/the-bitcoin-network-6713cb8713d.

[33] Mark, By, -, and Mark, *Full node and lightweight node*, Nov. 2018. [Online]. Available: https://www.mycryptopedia.com/full-node-lightweight-node/.

[34] J. A. D. Donet, C. Pérez-Sola, and J. Herrera-Joancomartı, "The bitcoin p2p network," in *International conference on financial cryptography and data security*, Springer, 2014, pp. 87–102.

[35] Y. Lewenberg, Y. Bachrach, Y. Sompolinsky, A. Zohar, and J. S. Rosenschein, "Bitcoin mining pools: A cooperative game theoretic analysis," in *Proceedings of the 2015 International Conference on Autonomous Agents and Multiagent Systems*, 2015, pp. 919–927.

[36] *P2p network*. [Online]. Available: https://developer.bitcoin.org/devguide/p2p_network.html.

[37] T. Neudecker and H. Hartenstein, "Short paper: An empirical analysis of blockchain forks in bitcoin," in *International Conference on Financial Cryptography and Data Security*, Springer, 2019, pp. 84–92.

[38] N. Maddrey, *Blockchain forks explained*, Sep. 2018. [Online]. Available: https://medium.com/digitalassetresearch/blockchain-forks-explained-8ccf304b97c8.

[39] F. Schär, "Blockchain forks: A formal classification framework and persistency analysis," *The Singapore Economic Review*, pp. 1–11, 2020.

[40] K. Leussink, *Hard and soft forks*, Feb. 2018. [Online]. Available: https://cryptographics.info/cryptographics/blockchain/hard-soft-forks/.

[41] H. Centieiro, *A complete decoding of the bitcoin block*, Jun. 2021. [Online]. Available: https://levelup.gitconnected.com/a-complete-decoding-of-the-bitcoin-block-578904267142.

[42] R. Rybarczyk, *Bitcoin p2pkh transaction breakdown*, Aug. 2020. [Online]. Available: https://medium.com/coinmonks/bitcoin-p2pkh-transaction-breakdown-bb663034d6df.

[43] *qué es sha-256?* Mar. 2021. [Online]. Available: https://academy.bit2me.com/sha256-algoritmo-bitcoin/.

[44] J. M. Lowery, "MD5 vs SHA-1 vs SHA-2 - Which is the Most Secure Encryption Hash and How to Check Them," *FreeCodeCamp*, Mar. 2020. [Online]. Available: https://www.freecodecamp.org/news/md5-vs-sha-1-vs-sha-2-which-is-the-most-secure-encryption-hash-and-how-to-check-them.

[45] B. i. Action, *What is the algorithm sha256?* Apr. 2021. [Online]. Available: https://medium.com/coinmonks/what-is-the-sha256-bitcoin-8b6e5b147629.

[46] J. Domínguez Gómez, "Criptografía: Función sha-256," 2018.

[47] J. Frankenfield, *Proof of work (pow)*, Sep. 2021. [Online]. Available: https://www.investopedia.com/terms/p/proof-work.asp.

[48] *Vitàlik buterin*, Dec. 2021. [Online]. Available: https://es.wikipedia.org/wiki/Vit%C3%A1lik_Buterin.

[49] . [Online]. Available: https://cryptorating.eu/whitepapers/Omni/MasterCoin%20Specification%201.1.pdf.

[50] V. Buterin *et al.*, "Ethereum white paper," *GitHub repository*, vol. 1, pp. 22–23, 2013. [Online]. Available: http://kryptosvet.eu/wp-content/uploads/2021/05/ethereum-whitepaper-kryptosvet.eu_.pdf.

[51] *Ethereum whitepaper*. [Online]. Available: https://ethereum.org/en/whitepaper/.

[52] A. M. Antonopoulos and G. Wood, *Mastering ethereum: building smart contracts and dapps*. O'reilly Media, 2018.

[53] Cointelegraph, *Bitcoin vs. ethereum: Key differences between btc and eth*, Dec. 2021. [Online]. Available: https://cointelegraph.com/ethereum-for-beginners/bitcoin-vs-ethereum-key-differences-between-btc-and-eth.

[54] Y. Chen and C. Bellavitis, "Decentralized finance: Blockchain technology and the quest for an open financial system," *Stevens Institute of Technology School of Business Research Paper*, 2019.

[55] U. W. Chohan, "Non-fungible tokens: Blockchains, scarcity, and value," *Critical Blockchain Research Initiative (CBRI) Working Papers*, 2021.

[56] B. Learn, *What is gamefi: Gamifying your crypto earning experience*, Oct. 2021. [Online]. Available: https://learn.bybit.com/crypto/what-is-gamefi/.

[57] *Explained: Play-to-earn games in metaverse and how they work*, Dec. 2021. [Online]. Available: https://www.cnbctv18.com/technology/explained-play-to-earn-games-in-metaverse-and-how-they-work-11903402.htm.

[58] *What is a dao and how do they work?* Oct. 2021. [Online]. Available: https://consensys.net/blog/blockchain-explained/what-is-a-dao-and-how-do-they-work/.

[59] L. V. Kiong, *DeFi, NFT and GameFi Made Easy: A Beginner's Guide to Understanding and Investing in DeFi, NFT and GameFi Projects*. Liew Voon Kiong, 2021.

[60] L. Ross, *Bitcoin vs ethereum*, Dec. 2021. [Online]. Available: https://www.benzinga.com/money/bitcoin-vs-ethereum/.

[61] *Introducción a los contratos inteligentes*. [Online]. Available: https://ethereum.org/es/developers/docs/smart-contracts/.

[62] Z. Zheng *et al.*, "An overview on smart contracts: Challenges, advances and platforms," *Future Generation Computer Systems*, vol. 105, pp. 475–491, 2020.

[63] *Figure 2: Smart Contracts LifeCycle 1) Creation of smart contracts:...* [Online; accessed 26. Feb. 2022], Feb. 2022. [Online]. Available: https://www.researchgate.net/figure/Smart-Contracts-LifeCycle-1-Creation-of-smart-contracts-Several-involved-parties-first_fig1_340376424.

[64] J. Benet, "Ipfs-content addressed, versioned, p2p file system (draft 3)," *arXiv preprint arXiv:1407.3561*, 2014.

[65] *Introducción a las dapps*. [Online]. Available: https://ethereum.org/es/developers/docs/dapps/.

[66] *Máquina virtual de ethereum (evm)*. [Online]. Available: https://ethereum.org/es/developers/docs/evm/.

[67] F. Vogelsteller and V. Buterin, *Eip 20: Erc-20 token standard, 2015*. [Online]. Available: https://eips.ethereum.org/EIPS/eip-20.

[68] *Estándar de token erc-20*. [Online]. Available: https://ethereum.org/es/developers/docs/standards/tokens/erc-20/#top.

[69] Jrgebk, *qué es un token erc-20?* Nov. 2021. [Online]. Available: https://academy.bit2me.com/que-es-erc-20-token/.

[70] W. Entriken, D. Shirley, J. Evans, and N. Sachs, "Eip 721: Erc-721 non-fungible token standard," *Ethereum Improvement Proposals*, 2018. [Online]. Available: https://eips.ethereum.org/EIPS/eip-721.

[71] *Estándar de token no fungible erc-721*. [Online]. Available: https://ethereum.org/es/developers/docs/standards/tokens/erc-721/#top.

[72] S. Bernal, *qué es un token erc 721? la era del coleccionismo digital*, Apr. 2021. [Online]. Available: https://academy.bit2me.com/que-es-token-erc-721/.

[73] P. Todd, Sep. 2016. [Online]. Available: https://petertodd.org/2016/opentimestamps-announcement#what-can-and-cant-timestamps-prove.

[74] A. BRANDOLI, "Blockchain notarization: Extensions to the opentimestamps protocol," 2019.

[75] B. Schneier, "Applied cryptography protocols," *Algorithms and Source Code in C*, 1995.

[76] Rssfeed, *La capitalización total del mercado de criptomonedas alcanza un nuevo máximo histórico por encima de los usd 1,900 millones*, Apr. 2021. [Online]. Available: https://www.criptostar.com/la-capitalizacion-total-del-mercado-de-criptomonedas-alcanza-un-nuevo-maximo-historico-por-encima-de-los-usd-1900-millones/.

[77] *Une-iso 21500:2013*. [Online]. Available: https://www.une.org/encuentra-tu-norma/busca-tu-norma/norma/?c=N0050883.

[78] A. Guide, "Project management body of knowledge (pmbok® guide)," in *Project Management Institute*, 2001.