uc3m | Universidad Carlos III de Madrid

University Degree in Computer Science and Engineering

2018-2019

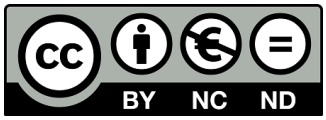
Bachelor Thesis

"Characterization of the topology of the Bitcoin network"

Guillermo Escobero Hernández

Marcelo Bagnulo Braun

Leganés, 2019



SUMMARY

Cryptocurrencies have shown great potential over the last years. They introduced a revolutionary concept: the no need for banks or third-party institutions to validate the payments. Transactions are public and known by all the users connected to a distributed network, so anyone can check the historical data. Having knowledge about different network parameters and nodes interaction can help to improve the protocol and to find vulnerabilities. Bitcoin was the pioneer and is nowadays the most used digital coin. This project proposes a method to explore the reachable nodes of the main Bitcoin network. The script connects to a set of initial nodes and requests their known peers, in order to connect to them and request their list of known peers also. Over 198,000 unique IP addresses were observed, although only 7500 nodes could be reached. The analysis of the data shows statistics of parameters of the nodes, like version used, geographic location or services implemented, and try to use them to give a general overview of the interaction between nodes.

Keywords: Peer-to-peer computing, Monitoring, Online Banking, Network Topology

DEDICATION

To my parents, Fabiola and Juan Carlos, who always supported me in every decision I made.

To LaVendicionDevs, my friends from university.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

1. INTRODUCTION

Cryptocurrencies are considered one of the most revolutionary after the Internet invention. In a globalized world, most of the everyday life aspects have suffered a digital transformation, and payments are not an exception.

Bitcoin was the first and is the most famous cryptocurrency nowadays. Its popularity has grown enormously in the last four years, showing great potential as a new digital currency and attracting the interest of people and governments (Fig.1.1).

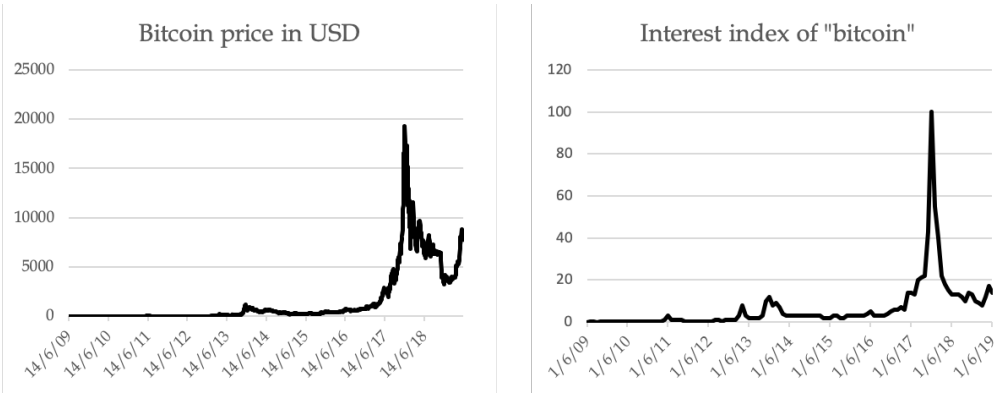


Fig. 1.1. Comparison of Bitcoin interest [1] with market price in USD [2]

1.1. Motivation

Bitcoin operates through a decentralized network to make all the transactions between users. All the data in the network is public and known by every node.

This project gives a top view of the network and extracts conclusions from the data obtained from all the reachable network participants. To obtain this data, an application is implemented to connect to the Bitcoin network and explore it.

On the one hand, monitoring a wide part of the network can give the contributors and maintainers the ability to detect anomalies and possible vulnerabilities of the network, helping to improve the protocol.

On the other hand, attackers can find weak points of the network, like highly connected nodes, that could attack to take control of them and perform a denial-of-service attack or

51% attack. This attack is based on controlling more than half of the nodes of the network in order to infect the network with tampered transactions.

1.2. Document structure

This document is divided in seven chapters:

- *Introduction*: this chapter. Contains the motivation for developing this project and the summary of the project documentation.
- *State of the art*: in this chapter background about electronic cash, cryptocurrencies and Bitcoin protocol is given. This will help the reader to understand the different related works discussed at the end of this chapter.
- *Description of the solution*: this chapter explains the methodology used and the design of the tool implemented.
- *Data obtained and results*: this chapter contains the experiment details and discusses all the obtained data.
- *Regulatory framework*: the current Bitcoin regulations applicable to this project are commented. This chapter also contains all the necessary software licenses and copyright notices.
- *Socio-economic environment*: first, the general planning and budgeting of the project are explained here. Then, the possible socio-economic impact of the project is discussed.
- *Conclusions and future lines of work*: this chapter contains the main conclusions of the project and some possible future lines of work.

2. STATE OF THE ART

2.1. Electronic cash and transactions

Since decades ago, electronic cash protocols have been widely used [3]. These protocols use digital currency signed by a trusted bank, and users can send this money directly to another user without exchanging card or account numbers. Security is ensured by public key digital signature schemes, but payee needed to verify the bank database of spent money to avoid double-spent coins.

Double spending is one of the biggest problems in an electronic cash system. When using conventional cash, this problem does not exist, as it is physical. But when using digital transactions, a user could easily copy a single transaction several times, paying with money already spent. Double spending is traditionally avoided making all the users trust in the validation of a third-party institution, like a bank.

However, this adds a point of failure to the system. This institution has to keep a record of all the transactions and sensible information of all users. An attack or even a human error could have important consequences. Also, if the servers or the connection fail, the service will be unavailable.

2.2. Bitcoin

Bitcoin is born as a peer-to-peer version of electronic cash. Created by Satoshi Nakamoto (pseudonym) in 2008, its first appearance was on a cryptographic mailing list. This message contained a white paper [4] describing a new concept: a purely peer-to-peer network to make payments that solved the problem of the previous proposed electronic cash models, the need of a trusted third-party institution to validate the transaction.

Nowadays, the Bitcoin project is called "Bitcoin Core" and is distributed under the MIT license. It is maintained by the Bitcoin Core Team, but anyone can contribute to its official repository in GitHub[1].

[1] https://github.com/bitcoin/bitcoin

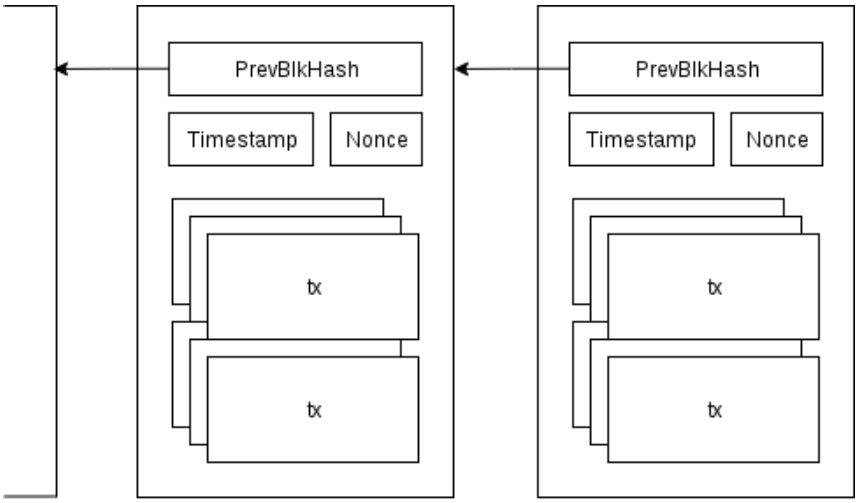


Fig. 2.1. Blockchain diagram

2.2.1. Blockchain

Bitcoin solves this problem without the need of a third-party institution. The solution is called “Blockchain”, a decentralized record of transactions based on cryptography proof instead of trust. The Blockchain is basically a chain of blocks of signed transactions. All transactions are public. This way, the payee can check all the blockchain back to the first transaction to check the validity of the transaction received.

To keep track of the order of transactions, Bitcoin uses timestamps. Data contained in each block of transactions is hashed with the current timestamp. This proves that the data in the block existed at that time. This hash also includes the previous block hash (chaining the blocks) (Figure 2.1).

The current size of the Bitcoin blockchain is about 223 GB [2].

2.2.2. Proof-of-Work and Consensus system

But, as the Bitcoin network has to be peer-to-peer and does not have a centralized server, it is needed a proof-of-work system. This works as a system of consensus between the nodes to agree on the order of transactions. Each block includes a “nonce”, a number that will be incremented until the hash of it with the rest of the data of the block begins with specific a number of zeros. This number of leading zeros is known as “difficulty of

[2]https://www.blockchain.com/es/charts/blocks-size

the block". It is variable as the protocol is designed to maintain a nearly constant release of new blocks per hour. This mathematical problem requires a lot of CPU effort, and the first node that founds a valid nonce gets a reward in bitcoins.

This difficult task will also avoid dishonest modifications of the blocks. If a malicious user tries to modify a block, he will need to redo the proof-of-work of that block and the following ones. The attacker could finish the hashing of all the blocks, but the network will always choose the largest chain. So, if more of the half of the total nodes are honest, the honest chain will grow faster, and the attacker's chain will be rejected. Then, he could be successful only if he controls a 51% or more of the total CPU power.

Providing that he can make more profit validating blocks honestly, this works as an incentive to not to try to attack the network, but to put an effort in validating and maintaining the network up. Making profit of these validations is commonly known as "mining".

2.2.3. Anonymity

Another of the advantages of not relying on a third-party institution is privacy. Users connected to the Bitcoin network are anonymous. Only the public key of the payee is needed to make payments, and real information of the user is not linked to transactions or stored in the network.

2.2.4. Bitcoin network

Bitcoin network has the structure of a peer-to-peer (decentralized) network. Makes use of the TCP/IP protocol over the port 8333 by default.

Protocol messages

- Version messages (*VERSION*): message containing the basic information of a node. When a node initiates a connection to a new peer, they exchange version messages (Fig.2.3).
- Address messages (*ADDR*): message containing a list of peers known by the node, with their IP addresses, ports, running services, and a timestamp.

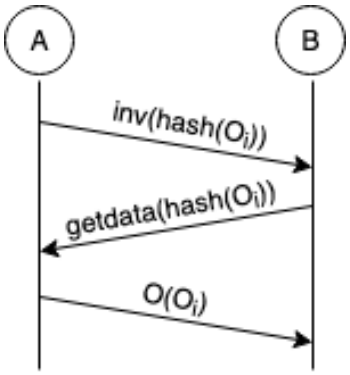


Fig. 2.2. Protocol used to broadcast new objects in the Bitcoin network (Standard relaying protocol)

- Get address messages (*GETADDR*): message used request the list of active known peers of a node. In response, the node will send a *ADDR* message.
- Inventory messages (*INV*): when a node creates or receives a new object (i.e. block or transaction), it broadcasts an inventory message to its neighbors that contains the hash of the new object(s). It can contain a maximum of 50,000 entries.
- Get data messages (*GETDATA*): nodes use this message to request a specific object, usually in response to an *INV* message. When a node receives an *INV*, it checks what objects are new, and request them using the hashes received and sending them to the peer embedded in a *GETDATA* message (Figure 2.2).
- Transaction messages (*TX*): it contains all the information about a certain bitcoin transaction. Nodes send this as a reply to a *GETDATA* message.
- Block messages (*BLOCK*): it contains all the information about a certain bitcoin block. Nodes send this as a reply to a *GETDATA* message.

Nodes classification

Bitcoin network is very width and not all the users connected to it implement the same services. Of course, all the users need to connect to the peer-to-peer network, but in order to make the clients lighter and faster, the developers limit them just to use the needed operations. This permits to use bitcoins to the less powerful and battery-limited devices like smartphones. For example, only a few of the total nodes download and

maintain the full blockchain, but Bitcoin protocol implements solutions to validate blocks and transactions without the need of keeping the whole blockchain in memory.

However, other nodes can also verify transactions without a full copy of the blockchain. They are called SPV nodes (simplified payment verification).

The bitcoin community has created different types of nodes, based on the needs and interests of each user, for example creating nodes with special network protocol for creating mining pools or connecting to a special subnet.

The main four components of a node are *wallet*, *miner function*, *blockchain* and *network* [5]:

- Wallet: a wallet functionality is needed if the user wants to store and make payments with her Bitcoins.
- Miner: this is only needed if the node is intended to make validations and block mining. Also, it consumes a high amount of resources.
- Blockchain: nodes can store the entire blockchain, the last n blocks mined, or even no blockchain.
- Network: the node can be configured to connect to the main Bitcoin network or can implement custom network protocols, for example, to support a mining pool.

2.2.5. Peer discovery

When a new node is connected to the network, the first operation is to find peers to connect.

If the node already knows at least one peer, it will connect to this peer establishing a TCP connection. After this, nodes will exchange version messages containing basic information to make sure they are compatible (Figure 2.3).

If the node does not know any peer yet, it will query DNS using "DNS seeds", which are usually hardcoded in the bitcoin clients. These servers provide a list of IP addresses of bitcoin nodes.

Now that our node is connected to one or more peers (and the "version handshake" is

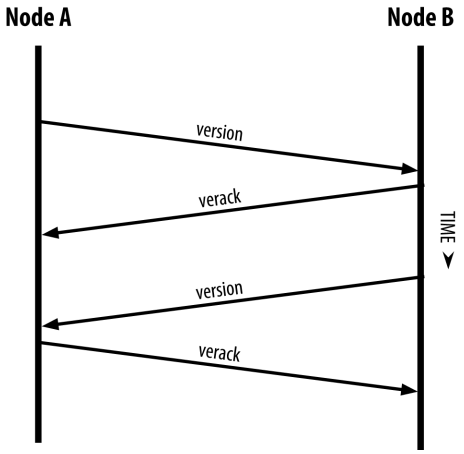


Fig. 2.3. The initial handshake between peers [5]

done), it will start to propagate its address and to discover new peers. It will send its IP address to its new peers, and these will forward the message to their neighbors. Another option is to ask its neighbors for a list of their peers and send the address message directly to them.

After all these connections, our node has different paths in the network. However, this peer discovering process must not be stopped, as nodes come and go, it will lose old peers and make new connections. Nodes send messages to check if the connection stills alive. If the peer does not reply for more than 90 minutes, it is assumed to be disconnected.

This "peer discovery" process also makes the network grow and shrink without any central control and ignore idle peers and network problems.

2.2.6. Transactions propagation

When a client wants to create a new transaction, these are the steps done:

- The payer creates a new transaction and signs it with its private key of the wallet currently used. This wallet contains a pair of keys (public key and private key) and a set of unspent transactions (bitcoins received but not spent yet).
- The payer broadcasts the transaction to its peers.
- The nodes that receive the transaction, will check if its correct (the coins are not double-spent, the transaction has no errors and the signature is correct). If it is correct, then they will also broadcast it. If not, they ignore the new transaction.

- This unconfirmed transaction will be kept in memory (mempool) and will be there until included in a block and mined. When the block containing the transaction is mined, is said that the transaction has one confirmation. Each block added to the chain after it will give an additional confirmation to the transaction.

- This process is expected to last about ten minutes. The protocol keeps a historical record of the average mining time of the blocks and increases or decreases the difficulty of the hash problem to keep the mining time around ten minutes.

- Once a miner mines the block, it broadcasts the block to its peers. Peers only accept the node if all transactions included are valid and not already spent. If it is correct, they will start working in the next block with other not confirmed transactions.

2.2.7. Block propagation

According to the Bitcoin documentation [6], when a node mines a new block it can propagate it in three different ways:

Unsolicited Block Push. The miner directly sends a *BLOCK* message to its peers.

Standard Block Relay. This is the default method. The node follows the standard relaying protocol already discussed (Figure 2.2).

Direct Headers Announcement. The sender can send immediately a *HEADERS* message containing the header of the new block without any request of its peers. The node will use this method with the peers that during the initial version handshake asked it with a *SENDHEADERS* message.

2.2.8. Synchronizing blockchains

Every node blockchain starts with the same block, the genesis block. This block is embedded in the source code of every client. Once a node connects to the bitcoin network, for the first time or after going offline a few minutes, it will compare its local blockchain with the copies of its peers.

The synchronization starts with the version message previously explained. This message contains a field called “BestHeight” that indicates the current height of the local

blockchain of the node. This refers to the number of blocks (or length) of the local chain. This way, when the outdated node notices that its peer has a different number of blocks, they will exchange the hash of their local top blocks.

Then, the node with the larger blockchain will notice that the received hash corresponds to a block in its blockchain, but not to the top block. This node will send to its peer a list with the hashes of the missing blocks.

After that, the outdated node starts asking for the content of the missing blocks from all its connected peers. This is done in order to not to overwhelm a single peer with a lot of requests.

2.3. Related work

There are several works focused on the structure and behavior of the Bitcoin network. The literature about this topic is wide and, due to the rapid development of Bitcoin, some methods are useless in the latest versions of the protocol.

Some of the main techniques developed to infer data about the nodes and the possible edges between them are AddressProbe[7], TxProbe[8] or time-based analysis[9].

2.3.1. AddressProbe and Coinscope

In 2015, a group of researchers of the University of Maryland developed a technique to discover links in the Bitcoin network [7]. The technique is called AddressProbe and is based on the way the Bitcoin clients keep track of the addresses of known peers.

Each node has a local database (*addrMan*) containing the addresses and its timestamps of known peers, currently connected or announced by other peers by *ADDR* messages. Bitcoin protocol updates these timestamps depending on the connection nature: for outgoing connections, the timestamp associated with that peer is updated for every message received. For incoming connections, the timestamp is fixed to the time when the connection was established. The rest of the peers that are known by *ADDR* messages received, the node keeps the new addresses and waits two hours before adding them to the *addrMan* database. However, if the address received is in the database, the timestamp will be updated with the new one if it is more recent.

A's ts of B	B's ts of A		
	ts≥2hr	Unique & ts<2hr	Not unique & ts<2hr
ts≥2hr	$\nexists$ edge	$\exists$ edge $B \rightarrow A$	Unclear $A \nrightarrow B$
Unique and ts<2hr	$\exists$ edge $A \rightarrow B$	$\exists$ edge $A \leftrightarrow B$	$\exists$ edge $A \rightarrow B$
Not unique and ts<2hr	Unclear $B \nrightarrow A$	$\exists$ edge $B \rightarrow A$	Unclear

Table 2.1. CONNECTION INFERENCE RULES FOR ADDRESSPROBE

GETADDR messages can be sent to the maximum possible number of nodes in the network and the *ADDR* messages received can be analyzed, comparing the timestamps to infer the possible edges of each node (Table 2.1).

The authors implemented this method and develop a platform called Coinscope, that isolates the functionalities of the Bitcoin protocol that AddressProbe needs. These are focused on creating and maintaining long-lived connections, sending *GETADDR* messages in an efficient way and listening only to relevant messages. Authors declare that using Coinscope, they were able to scan the whole network in a matter of minutes.

After the experiment, the authors concluded:

- Most of the nodes have degree between 8 and 12. However, some persistent nodes reach 10000 connections. The majority of them are mining pools and measurement and research nodes.
- Bitcoin network does not behave like a traditional random graph.
- The broadcast topology conceals influential nodes that represent disproportionate amounts of mining power (2% of the nodes account for 75% of the total mining power).

After the release of this work, the contributors of BitcoinCore changed the way the Bitcoin clients updates the timestamps (version 0.10.1), making AddressProbe method useless [10].

2.3.2. TxProbe

TxProbe [8] is a technique to infer the topology of the public Bitcoin network. Its basic inferring technique is based on "orphan" transactions.

An orphan transaction is a transaction that arrives before its parents. That means that it spends coins that are not yet included in the blockchain. When a node receives a new transaction that cannot be validated, it is marked as an orphan transaction and stored in a list called *MapOrphanTransaction*. This way, the node waits until the ancestors arrive to validate this orphan transaction.

The basic technique used by TxProbe takes advantage of these orphan transactions. When a node receives an orphan transaction, it will keep it in *MapOrphanTransaction* structure. After that, if it receives a *INV* message containing the hash of the orphan transaction, it will not request the transaction to the peer (avoiding the broadcast of a non-validated transaction). This allows knowing if the node has received the transaction or not.

To check if an edge between two nodes exists, the testing node creates two transactions that spend from the same coin, and send one to the first node and the other to the second node. This way, each node will accept the received transaction, and if one node sends its new transaction to the other peer, this will refuse it, as will detect it as a double-spending transaction.

After that the testing node creates a third new transaction that spends coins from the first transaction, and sends it only to the first node. If the edge between the nodes exists, the second node will receive the new transaction and will mark it as an orphan, storing it in *MapOrphanTransaction* as it does not know about its parent.

Finally, the testing node will send an *INV* message with the third transaction hash. If the edge exists, the node will have the transaction stored, so it will not reply asking for it.

One problem of this technique is that is an invasive method. *MapOrphanTransaction* has a limited size of 100 orphan transactions. TxProbe cleans this structure of the targeted nodes evicting previous existing transactions to get the maximum possible space to store its custom orphan transactions.

TxProbe scans the Bitcoin main network in about 8.25 hours.

After the experiment, the authors concluded (the following data was obtained from the Bitcoin test network (2018), not the main network):

- Average degree is 16.6. Most of the nodes have 7 to 14 peers. The maximum degree observed was 59.
- 733 nodes with 6090 edges.
- Non-random structure.
- This conclusions cannot be applied to the main network. TxProbe would work on the main network, but the authors did not perform the experiment on the Bitcoin main network as they could not predict the effect that would occur to real transactions and network congestion.

2.3.3. Timing analysis of the Bitcoin network

In 2016, researchers from the Karlsruhe Institute of Technology developed a technique to infer the structure between the nodes of the reachable Bitcoin network [9]. They created a model of the Bitcoin network and then compared it with the real main network. The method used was based on connecting to different nodes and calculating the receiving times of each transaction or block from each node. This data permitted the authors to design a probabilistic model that calculates if two nodes are connected. The precision obtained was ~40%.

Bitcoin clients implement *trickling* techniques that add random delay to messages to avoid time analysis. The authors solved this comparing the time that the message took to be received with the approximation of node location, network delay (using *ping* system messages) and the application latency (using Bitcoin protocol *PING* messages) the *trickling* latency can be approximated.

They also concluded that the *trickling* techniques can help an attacker to analyze the traffic if it is not configured properly.

2.3.4. Bitnodes.earn.com crawler

Bitnodes [11] is a project oriented to estimate the size of the Bitcoin network. It is also deployed in a website[3] where the user can have access to historical data of different snapshots of the network, as well as search nodes by IP or check current statistics of the network.

The data obtained from the methodology implemented in this project was compared with data from Bitnodes website.

[3] https://bitnodes.earn.com/

3. DESCRIPTION OF THE SOLUTION

3.1. Goal

To retrieve all the possible data about each reachable node of the Bitcoin network.

3.2. Main methodology

The solution proposed is based on the list of known active peers that every node keeps in memory. The system connects to new peers concurrently, asking about their peers to connect to them and repeat the process until all the nodes are discovered.

1. First, the application makes an initial DNS discovery of peers. It connects to a set of known DNS nodes (marked as "SeedNodes" in fig.3.1) that keep an updated list of active nodes.

2. Once connected, the application will ask each one for their list of known nodes (IP addresses and ports) sending them a *GETADDR* message.

3. Then, the application will wait for the *ADDR* message. When received, the list of nodes will be added to the list of pending nodes, and the connected node will be marked as "explored" to avoid connecting to the same node twice in the same session.

4. The application will start trying to connect to each node in the "pending" list and sending its *VERSION* message. If the node replies, it will be saved as an "explored" node and a *GETADDR* message will be sent to ask for its list of known peers.

5. If the node replies, the list of announced peers will be saved in the "pending" list. Then the application will start to try connecting to them.

To avoid overloading the network, it does not make any request of blocks or transactions, and it only requests the list of peers once to each node. The connection is closed after receiving the information to minimize the number of simultaneous connections.

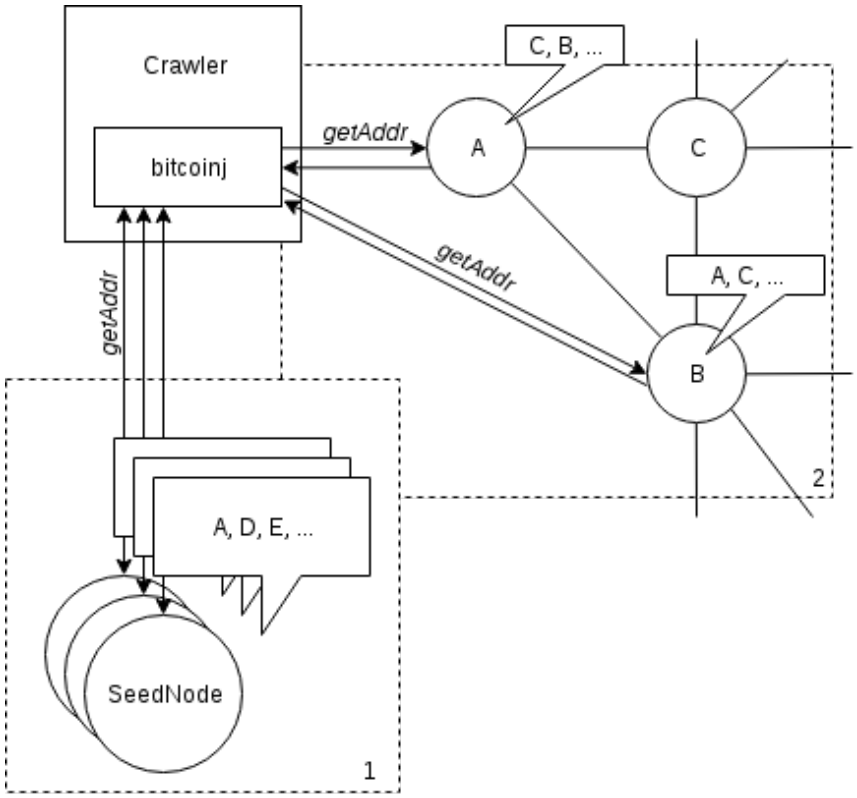


Fig. 3.1. Detail of the technique

3.3. Detailed design

The proposed solution is a "*crawler*" application developed in Java programming language [12], and uses Bitcoinj [13] library to connect and manage the connections to the different peers. The source code can be found in GitHub[4]. After obtaining the experiment data, a Python [14] script is used to analyze it.

After connecting to each node, the parameters are obtained from two types of messages: *VERSION* and *ADDR*. From the *VERSION* message, the relevant data for this case are *version*, *services*, *addr_from* and *start_height* fields (Table 3.1). This indicates that the node is running the Bitcoin protocol and online. This data is saved as a tuple *(IPv4/IPv6 address, port, height, version, sub-version, local services)* that represents each explored node.

ADDR messages help to know all the nodes that the active nodes have seen connected to the Bitcoin network recently. This is the data that the crawler uses to explore new nodes.

[4]https://github.com/GuillermoEscobero/bitcrawler/blob/master/Crawler.java

Description	Data type	Comments
version	int32_t	Version of the Bitcoin protocol that the node is running
user_agent	int32_t	Version of the Bitcoin client application that the node is running. Also called "subVersion"
services	uint64_t	Number that indicates the services that the node implements
timestamp	int64_t	Current system timestamp of the node
addr_recv	net_addr	net_addr structure containing data about the IP address and port of the receiver of this version message
addr_from	net_addr	net_addr structure containing data about the IP address and port of the sender of this version message
start_height	int32_t	Length of the largest local blockchain of the node. Also called "best height"

Table 3.1. VERSION MESSAGE

Description	Data type	Comments
count	var_int	Number of entries that the message contains
addr_list	(uint32_t + net_addr)[]	List of the peer addresses (Table 3.3)

Table 3.2. ADDR MESSAGE

Each peer address sent by a node is defined by a tuple: (IP of the sender node, port of the sender node, IP of the known peer, port of the known peer).

Description	Data type	Comments
time	uint32	Timestamp associated to the peer
services	uint64_t	Services supported
IPv6/4	char[16]	IPv6/4 address
port	uint16_t	Port number

Table 3.3. NET_ADDR STRUCTURE

4. DATA OBTAINED AND RESULTS

Bitcoin network was monitored for 3 days, making an experiment every two hours. Over three millions of entries were obtained in every snapshot of the network (~150 MB of data). The following discussed data is obtained from a random network snapshot, as the experiment period was not long enough to extract conclusions of the differences between snapshots. This data was obtained on July 1st, 2019 from Madrid, Spain.

The crawler application was capable of analyzing the main Bitcoin network in 44 minutes on average. The end of the analysis was considered if the application did not connect to any new peer in the last 15 minutes. This is because the rate of explored nodes drops gradually in time as it is harder to find new nodes as the number of explored nodes increases.

After analyzing the data, the number of unique active nodes discovered was 7530. Only a 84,25% replied to our *GETADDR* request. These nodes sent over 198 thousands of unique IP addresses.

It is important to note that those addresses belong to nodes of the Bitcoin nodes, but they do not have to be full nodes. Bitcoin network is formed by several types of nodes as discussed before (subsection 2.2.4). Some of them are programmed to not to reply to *GETADDR* messages or any other messages, in order to light the computation. Also, the list of known peers that each node keeps do not have to be a list of connected peers. Even the node maybe never connected to them, so edges between nodes cannot be inferred from these data. These addresses can be learned from *ADDR* messages received from other peers that can contain addresses not currently online. These two reasons explain the big difference between explored nodes (~7500) and discovered addresses (~198,000).

Figure 4.1 shows how most of the nodes are divided into two big groups. The first group sent 1,000 peer addresses and the second group sent only one address. The reasons for these numbers are two. 1,000 is the limit of addresses per each *ADDR* message. Also, when a node connects to others, it is normal to send an *ADDR* message containing its own address. That explains the big number of nodes that only sent one entry.

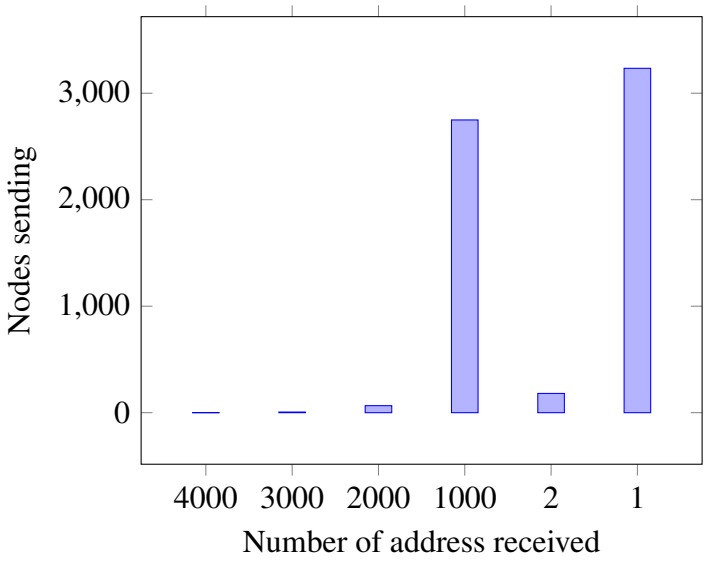


Fig. 4.1. Length of the ADDR messages received

Protocol Version

Table 4.1 contains the protocol versions seen in the experiment. The most used in the Bitcoin network is 70015 with 95.87% of the explored nodes using it. This matches with the version protocol used by Bitcoin Core[5].

Client version

Table 4.2 shows the most used Bitcoin clients used. There are hundreds of different clients and implementations. Also, the text identifying the client running (*user_agent*) can be changed and some people use it to promote their website or "post" a message.

The reader can easily notice that most of the client versions start by "Satoshi". All these versions are releases of Bitcoin Core that are named after the creator of Bitcoin. The last version is 0.18.0, released on May, 2nd 2019. It can be observed that most of the nodes still running the previous version 0.17.1. Famous bitcoin libraries as btcd-wire [15], bcoin [16] or bitcore [17] also appear on the data obtained.

[5]https://github.com/bitcoin/bitcoin/blob/master/src/version.h

Protocol version	Nodes observed
70015	7219
70012	159
70014	61
70013	32
70002	32
80002	21
70016	3
50400	2
80003	1

Table 4.1. PROTOCOL VERSION

Services

When connecting to a new peer, it will send some flags indicating the services that it supports (Table 4.3). This is interesting as powerful nodes support services that can help other nodes with limited resources to check if the transactions are valid or not, making the deployment of Bitcoin applications easier and less resource-consuming.

The obtained data shows a 97.38% of total nodes declare themselves as full-blockchain nodes (*NODE_NETWORK* flag). The rest of them are *pruned* nodes, that means they do not have the full blockchain downloaded.

78.87% of explored nodes implement *NODE_NETWORK*, *NODE_BLOOM*, *NODE_WITNESS* and *NODE_NETWORK_LIMITED*, that are all the services currently supported by Bitcoin Core.

Bitcoin Client and version	Nodes observed
Satoshi:0.17.1	2103
Satoshi:0.18.0	1800
Satoshi:0.16.3	631
Satoshi:0.17.0	533
Satoshi:0.13.2	411
Satoshi:0.17.0.1	346
Satoshi:0.15.1	295
Satoshi:0.16.0	260
Satoshi:0.12.1(bitcore)	100
Satoshi:0.16.2	98
Satoshi:0.18.99	97
Satoshi:0.14.2	74
Satoshi:0.14.99	64
Satoshi:0.16.1	59
Satoshi:0.15.0.1	58
Satoshi:0.17.99	53
Satoshi:0.16.99	44
Satoshi:0.12.1	42
Satoshi:0.13.1	32
btcwire:0.5.0btcd:0.12.0/	28

Table 4.2. USER AGENTS

Local services supported	Description
NODE_NETWORK	This node is capable of sending all historical full blocks, not just headers.
NODE_GETUTXO	This node gives support to check unspent transactions. These nodes help light-weight nodes to validate transactions. Bitcoin Core does not support it yet.
NODE_BLOOM	This node supports bloom filters. Light nodes can query transactions in the blockchain fast without downloading all of it.
NODE_WITNESS	This node supports a different serialization format for transaction messages.
NODE_NETWORK_LIMITED	The node is capable of sending at least the last 288 blocks. Nodes with pruned blockchain use this flag.

Table 4.3. SUPPORTED SERVICES

Analysis of IP addresses

Figure 4.2 shows the occurrences of each IP address observed in one network snapshot, where all the IP addresses are ordered from most to less frequent. The most frequent IP appears 87 times. This means that 87 different peers knew that peer at the moment of the experiment. It can be seen that the number of occurrences drops, with most of the IP addresses being known by 10 or less nodes.

This difference can be due to different factors. One of them is up-time. The nodes with the largest up-time are the most known, as they maintain the connections for more time and have more probability of being broadcast. Another possible reason is that there are nodes with a high number of connections, with 2% of the nodes controlling the 75% of the total computation power [7].

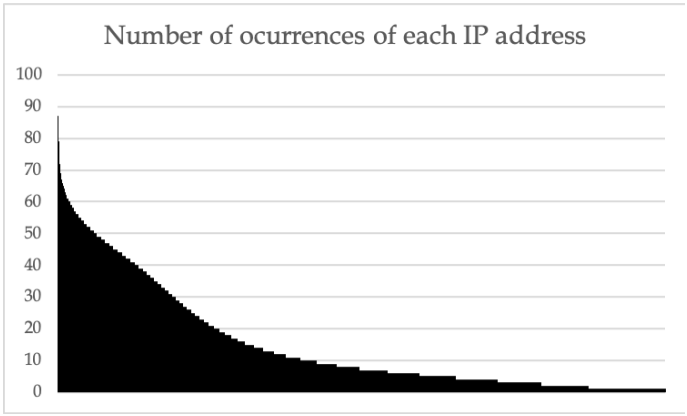


Fig. 4.2. Number of occurrences of each peer address

Ports used

Bitcoin network uses port 8333 by default. However, nodes can use custom ports, as they inform its port number to its peers in the initial handshake. 94.08% of the total observed IP addresses uses the default port. Table 4.4 shows the five more used ports of the explored nodes.

Port used	Nodes observed
8333	6791
8433	70
9595	49
8885	34
6333	12

Table 4.4. PORTS USED

Block height

While the tool is exploring the network, new blocks are mined and broadcast in the network, extending the blockchains of the nodes. In average, every 10 minutes a new block is mined, so during each experiment, the block height will increase around 4 blocks. Because of this, updated nodes can report different block heights depending on the time they are explored.

This was taken into account in the data analysis. 95.29% of the total nodes reported

Block height observed	Percentage of the total nodes
733864	0.096980
580976	0.027709
580975	3.809920
580974	11.637573
580973	79.855916
580972	0.055417
580971	0.027709

Table 4.5. BLOCK HEIGHT EXAMPLE DATA

a block height of the last block. This was decided based on the most appeared block height and the following three consecutive blocks. Table **??** shows an example. It was assumed that *580973* was the last mined block when the experiment started, but the Bitcoin blockchain reached *580976* blocks while the tool was executing.

The rest of the nodes showed anomalous values. There were some groups of nodes that announced similar heights between them, but with a big difference (ahead and behind) with the accepted blockchain. This may be due to the existence of small groups of nodes connected using a custom blockchain with research or testing purposes. Other nodes were very outdated, reporting a block height of a few months, or even 10 years ago [6].

Geographic distribution

Bitcoin network has participants around the world. IP addresses of the explored active nodes were located and are represented in figure 4.4. The United States is the country with the highest number of running nodes, followed by Germany and France. In countries where Bitcoin use is forbidden, like China or Brazil [18], users decide to connect although they are breaking the law.

It is known that China is the country with most mining pools, controlling about 75% of the total computation mining power [19], mainly due to the low power supply price. The top three mining pools (BTC.com [20], AntPool [21] and F2Pool [22]) are located there and mine 42% of the daily blocks in average [19]. However, these powerful nodes

[6] https://www.blockchain.com/es/btc/block-height/99409

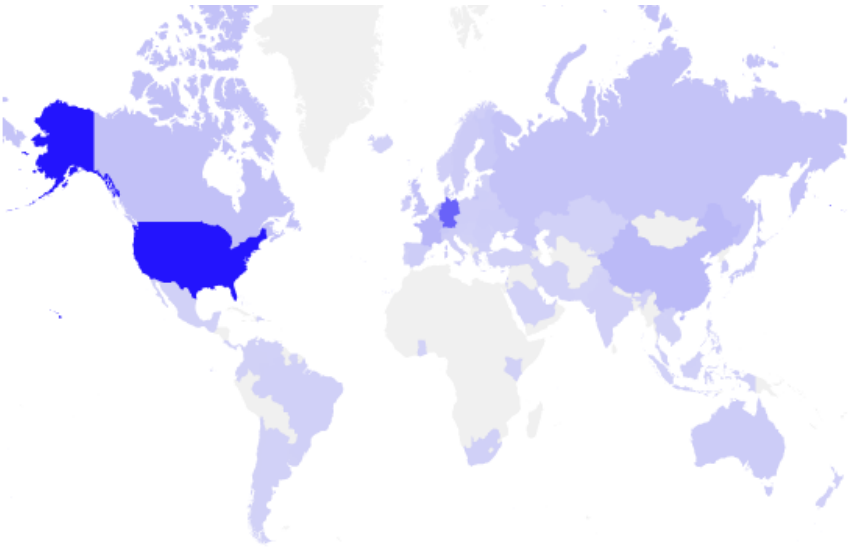


Fig. 4.3. Heat map of the Bitcoin network

are not represented in the map, as all the nodes of the mining pools are not reachable from the public network. Mining pools also make great efforts in keeping as private as possible in order to avoid possible attacks.

TOR network

Some users prefer to connect to the Bitcoin network using proxies to protect their identity even more. In countries where Bitcoin is prohibited and its access is blocked, users still use it connecting through virtual private networks (VPNs) or networks with special encryption protocols like TOR.

The TOR project [23] (The Onion Router) is an open source project focused on improving the anonymity and privacy on the Internet. Tor network is a distributed peer-to-peer network. The privacy of a user is achieved thanks to the path that the data sent will follow until it reaches its final destination. In every jump from a node to another, the data is encrypted and routed to another node, in order to avoid a man-in-the-middle attack. Instead of using standard IP addresses, the Tor project identifies its nodes with *.onion* addresses (*onion* because of the similarity with its encrypting method that adds a layer in every jump). As the IP of the nodes is unknown and the data can exit the network from any node, it is very complex to identify and locate an onion address.

The easiest way to use TOR is through a web browser developed by its contributors

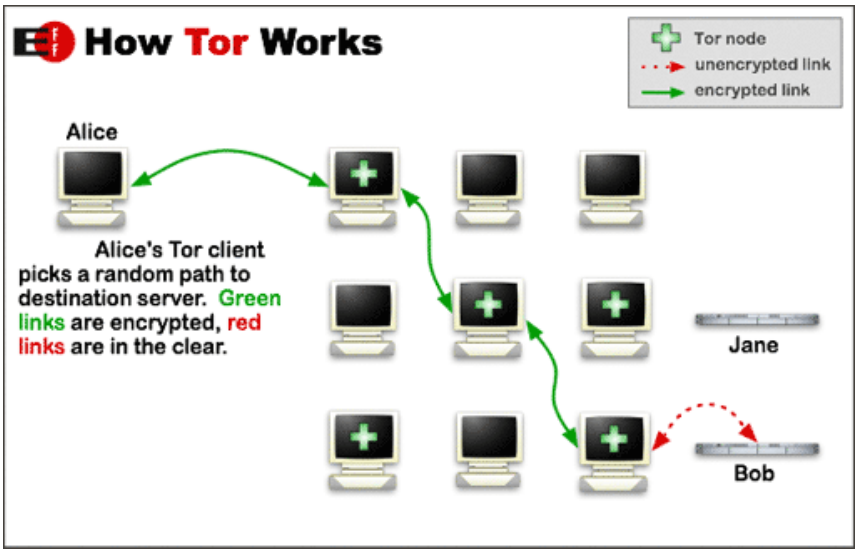


Fig. 4.4. TOR network structure. Diagram originally contributed by the Electronic Frontier Foundations and under a Creative Commons Attribution 3.0 United States License.

and maintainers. It converts the computer executing it in a TOR node and allows the user to access websites using *.onion* domains.

Most Bitcoin clients implement the functionality of using a proxy server, and the user can easily download the software from the Tor project website to connect to the Tor network and route its traffic through it.

5. REGULATORY FRAMEWORK

5.1. Bitcoin regulation

As discussed before, the main attractions of cryptocurrencies are the anonymity and the no need for a financial institution to regulate it. Bitcoin is also known to be one the currency used to support illegal activities in the "Dark Net" and tax evasion because of its anonymity [24].

In addition to these reasons, the difficulty of applying regulations to a distributed public network is making the governments start thinking about new laws and ways to control this new currency. Most of the countries permit users to use Bitcoin and make payments, but some of them make a distinction treating cryptocurrencies as digital goods instead of legal currency. However, other countries, like China, have prohibited the payments and exchanges of Bitcoins with legal currencies [18].

This experiment is based in Spain, where the Bitcoin transactions are permitted. Virtual coins are not considered a currency as they are not issued by an official monetary authority (European Union does not consider Bitcoin as a legal currency also). Instead, they are considered digital goods and are regulated by the Civil Code [25].

However, the laws regarding the transactions of cryptocurrencies do not apply to this work, as the method just connects to other nodes without doing any other operations like transactions or mining. Also, the method does not store or manage sensible data, only public parameters of clients that are needed by the Bitcoin protocol to operate.

5.2. Ethical matters

Regarding ethical issues, some users can see the discussed technique as a flooding or denial-of-service attack, as connecting to a big number of nodes goes against the decentralized nature of Bitcoin. Also, it adds overhead to the network for running a task that was not initially intended to be performed.

5.3. Software licenses

Java Development Kit (JDK)

JDK used in this project is owned by Oracle America, Inc. It is distributed under Oracle Binary Code License (BCL) that allows the user to use JDK to compile and run programs [26].

Python

Python is distributed as Open Source [27] software and its releases are GPL-compatible [28].

GeoLite2 databases

This project includes GeoLite2 data created by MaxMind [7]. The GeoLite2 databases are distributed under the Creative Commons Attribution-ShareAlike 4.0 International License [29].

bitcoinj

Bitcoinj is distributed under the Apache License 2.0 [30]. Allows the user to modification, distribution, and usage with commercial purposes.

OpenHeatMap

OpenHeatMap service was used to generate the world heatmap of Bitcoin nodes population. It is licensed under GNU GPL [28].

Microsoft Excel

Microsoft Excel by Microsoft was used under an Office 365 University license.

[7] https://www.maxmind.com

6. SOCIO-ECONOMIC ENVIRONMENT

This chapter contains the planning and budget used to carry out this project. After that, the possible socio-economic impact is discussed.

6.1. Planning

The project was divided into different stages (fig.6.1) that were developed in four months (83 days), starting in February 2019 and finishing in June 2019.

The first step was to gain general knowledge about Bitcoin and its network protocol. Once the Bitcoin protocol was fully understood, the research of academic works related to the problem started. These works helped to adapt the methodology to the resources and deadlines provided. Then, the development of the method and the programming of the tool started. After that, the experimentation period started, monitoring the network to obtain all the data used for analysis and conclusions. Finally, the documentation of the project was written.

A Gantt chart was designed to provide a general view of the project stages (fig.6.2).

Start Date	End Date	Description	Duration (days)
20/2/19	28/3/19	Bitcoin general research	36
29/3/19	23/4/19	Previous works and current state of the art	25
24/4/19	12/5/19	Testing of different libraries	18
13/5/19	20/5/19	Design and implementation of the tool	7
21/5/19	25/5/19	Testing of the tool and analysis of the first data obtained	4
26/5/19	30/5/19	Monitoring period	4
31/5/19	1/6/19	Analysis of the obtained data	1
2/6/19	15/6/19	Project documentation	13

Fig. 6.1. Duration of the main tasks of the project

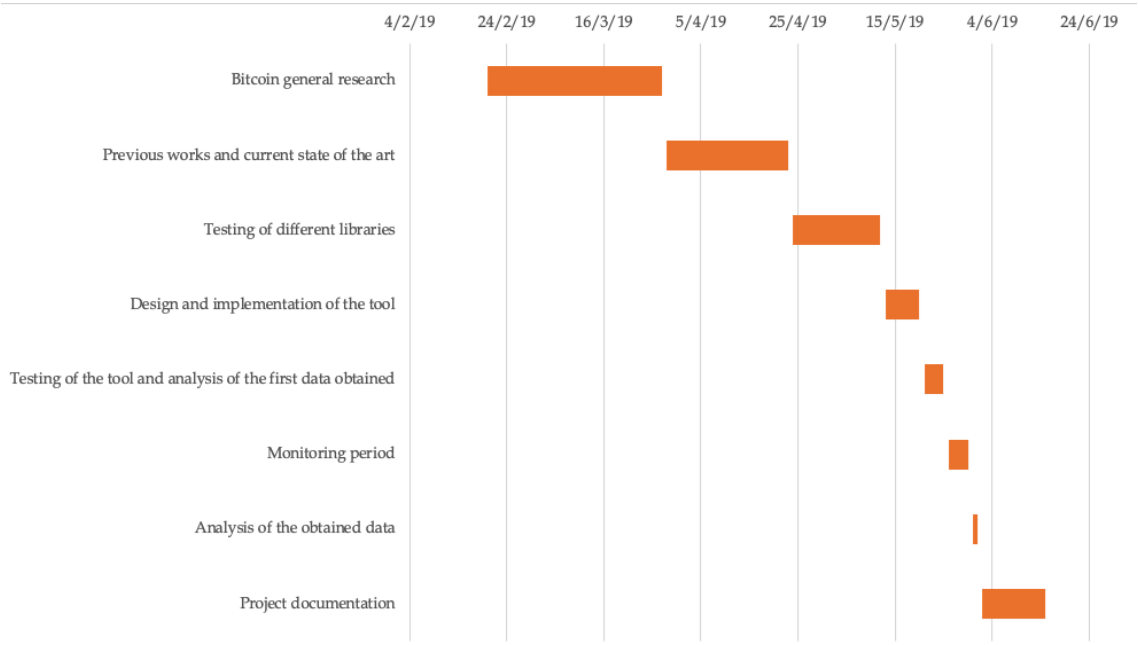


Fig. 6.2. Gantt chart of the main tasks of the project

6.2. Budget

This section lists all the resources used in the project, as well as their economic cost.

6.2.1. Hardware

The project was designed, implemented and deployed in a *Apple Macbook Pro Mid 2012* (2.9 GHz Intel Core i7, 8 GB 1600 MHz DDR3) computer. A *Askey RTF811VW* wireless router was used as the gateway to connect to the internet (provided by the ISP).

Description	Cost	Lifetime	Usage time	Total cost
Computer	1,525 €	96 months	4 months	63.54 €
Router	0 €	48 months	4 months	0 €
Total				**63.54 €**

Table 6.1. HARDWARE COSTS

6.2.2. Software

Description	Usage time	Cost per month	Total cost
Jet Brains InteliJ IDEA CE	2 months	0 €	0 €
Oracle JDK	2 months	0 €	0 €
Docker Desktop Community	2 month	0 €	0 €
Python	1 month	0 €	0 €
Pandas	1 month	0 €	0 €
Overleaf.com	1 month	0 €	0 €
GitHub.com	3 months	0 €	0 €
Microsoft Excel	1 month	0 €	0 €
Total			**0 €**

Table 6.2. SOFTWARE COSTS

6.2.3. Consumables

Description	Units	Unit cost	Total cost
Paper (500 sheets)	1	4.99 €	4.99 €
Ink cartridge	1	9.81 €	9.81 €
Total			**14.8 €**

Table 6.3. CONSUMABLES COSTS

6.2.4. Human resources

Position	Cost per hour	Dedication	Taxes (23.6%)	Total cost
Software engineer	12 €	170 hours	481.44 €	2,521.44 €
Project supervisor	15 €	20 hours	70.8 €	370.8 €
Total				**2,892.24 €**

Table 6.4. HUMAN RESOURCES

6.2.5. Indirect costs

Description	Usage time	Cost per month	Total cost
Internet connection	4 months	32.64 €	130.56 €
Power supply	4 months	42.78 €	171.12 €
Transport	4 months	20 €	80 €
Per diem	4 months	112 €	448 €
Total			**829.68 €**

Table 6.5. INDIRECT COSTS

6.2.6. Total costs

Section	Total cost
Hardware	63.54 €
Software	0 €
Consumables	14.8 €
Human resources	2,892.24 €
Indirect costs	829.68€
Total cost of the project	**3,800.26 €**

Table 6.6. TOTAL COSTS

6.3. Socio-economic impact

The economic value associated with Bitcoin makes the users treat its network as a different one, as real profit can be obtained participating in it. This accelerates its development and expansion, but also attracts the interest of potential attackers.

The tool implemented in this project can be used in future applications for monitoring and detecting possible patterns and anomalies on the Bitcoin network. Also, if the monitoring of the network is maintained over a large amount of time, some interesting data will be obtained and relations with the price of Bitcoin may be found.

However, the high popularity and economic interests associated with Bitcoin impulsed a big inversion in research and projects, like mining pools, trading sites or online wallets, that have powerful systems that could be used to monitor the network and obtain the data by themselves.

Apart from technical analysis, the tool developed and this report can be used to help people to understand Bitcoin from the computer network view and see how the different agents and protocols of a distributed network can affect to others.

7. CONCLUSIONS AND FUTURE LINES OF WORK

7.1. Final conclusions

In this project, a wide part of the Bitcoin network was monitored. However, not all nodes are reachable, like mining pools, so this is not a full representation of the network. Most of the explored nodes contributed to the network properly, although some of them only implemented the essential services and did not reply to peers requests to save computation resources. Also, some nodes used different parameters, like custom clients or protocol versions.

7.2. Future lines of work

In order to obtain more relevant data about specific Bitcoin protocol parameters, like blocks and transactions propagation, several functionalities can be implemented to improve the monitoring of the network and analysis of parameters:

- Deployment of the crawler in the cloud, with different gateway locations to improve the speed of the nodes discovery.
- Implementation of an alert system that allows monitoring the value of a specific parameter and raises an alert when it fulfills the rules defined by the user.
- Implementation of a web application showing the metrics of a recent network snapshot.
- Integration with an edge-inferring method, some of them were discussed on section 2.3, to find relations between both sources of data.
- Monitoring of block propagation and transactions received.

BIBLIOGRAPHY

[1] *"bitcoin" - Google Trends*. [Online]. Available: https://trends.google.com/trends/explore?date=2009-05-12%5C%202019-06-12%5C&q=bitcoin.

[2] *Bitcoin Market Price (USD)*. [Online]. Available: https://www.blockchain.com/es/charts/market-price.

[3] D. Chaum, "Blind signatures for untraceable payments", in, ser. Advances in Cryptology - Proceedings of Crypto 82. 1983, p. 199.

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

[5] A. M. Antonopoulos, *Mastering Bitcoin*, 2nd ed. O'Reilly Media, 2017.

[6] The Bitcoin Core developers, *Bitcoin protocol documentation*. [Online]. Available: https://en.bitcoin.it/wiki/Protocol_documentation (visited on 04/30/2019).

[7] A. Miller *et al.*, *Discovering bitcoin's public topology and influential nodes*, Accessed: 2019-04-12, May 2015. [Online]. Available: http://www.cs.umd.edu/projects/coinscope/coinscope.pdf.

[8] S. Delgado-Segura *et al.*, "Txprobe: Discovering bitcoin's network topology using orphan transactions", *CoRR*, vol. abs/1812.00942, 2018. arXiv: 1812.00942. [Online]. Available: http://arxiv.org/abs/1812.00942.

[9] T. Neudecker, P. Andelfinger, and H. Hartenstein, "Timing analysis for inferring the topology of the bitcoin peer-to-peer network", in *2016 Intl IEEE Conferences on Ubiquitous Intelligence Computing, Advanced and Trusted Computing, Scalable Computing and Communications, Cloud and Big Data Computing, Internet of People, and Smart World Congress (UIC/ATC/ScalCom/CBDCom/IoP/SmartWorld)*, Jul. 2016, pp. 358–367. DOI: 10.1109/UIC-ATC-ScalCom-CBDCom-IoP-SmartWorld.2016.0070.

[10] The Bitcoin Core developers, *Bitcoin Core 0.10.1 release notes*, 2015. [Online]. Available: https://github.com/bitcoin/bitcoin/blob/v0.10.1/doc/release-notes.md (visited on 04/30/2019).

[11] *Bitnodes crawler repository*. [Online]. Available: https://github.com/ayeowch/bitnodes.

[12] *Java programming language*. [Online]. Available: https://www.java.com/.

[13] *Bitcoinj library*. [Online]. Available: https://bitcoinj.github.io/.

[14] *Python programming language*. [Online]. Available: https://www.python.org/.

[15] *Btcd repository*. [Online]. Available: https://github.com/btcsuite/btcd.

[16] *Bcoin repository*. [Online]. Available: https://github.com/bcoin-org/bcoin.

[17] *Bitcore library*. [Online]. Available: https://bitcore.io/.

[18] G. L. R. C. The Law Library of Congress, *Regulation of Bitcoin in Selected Jurisdictions*, 2014. [Online]. Available: https://www.loc.gov/law/help/bitcoin-survey/ (visited on 06/10/2019).

[19] Blockchain.com, *Distribution of hash rate between mining pools*. [Online]. Available: https://www.blockchain.com/pools?timespan=4days.

[20] *Btc.com mining pool homepage*. [Online]. Available: https://pool.btc.com/.

[21] *Antpool mining pool homepage*. [Online]. Available: https://www.antpool.com/.

[22] *F2pool mining pool homepage*. [Online]. Available: https://www.f2pool.com/.

[23] *Tor project*. [Online]. Available: https://www.torproject.org/.

[24] M. Tsukerman, "The block is hot: A survey of the state of bitcoin regulation and suggestions for the future", *Berkeley Technology Law Journal*, vol. 30, no. 4, pp. 1127–1170, 2015. [Online]. Available: https://scholarship.law.berkeley.edu/cgi/viewcontent.cgi?article=2084&context=btlj.

[25] Boletín Oficial del Estado, *Código Civil arts. 335, 337 and 345*, 1889. [Online]. Available: https://www.boe.es/eli/es/rd/1889/07/24/(1)/con (visited on 05/29/2019).

[26] *Oracle Binary Code License Agreement for the Java SE Platform Products and JavaFX*, 2017. [Online]. Available: https://www.oracle.com/technetwork/java/javase/terms/license/index.html (visited on 06/01/2019).

[27] *The Open Source Definition*, 2007. [Online]. Available: https://opensource.org/osd.

[28] Free Software Foundation, Inc., *Gnu general public license*, Accsessed: 2019-06-01, Jun. 2007. [Online]. Available: https://www.gnu.org/licenses/gpl-3.0.html.

[29] *Attribution-ShareAlike 4.0 International (CC BY-SA 4.0) Creative Commons license*. [Online]. Available: https://creativecommons.org/licenses/by-sa/4.0/.

[30] *Bitcoinj license (Apache License 2.0)*, 2004. [Online]. Available: https://github.com/bitcoinj/bitcoinj/blob/master/COPYING.

[31] B. Schoenmakers, *Basic security of the ecash payment system*, 1998. [Online]. Available: http://www.win.tue.nl/~berry/papers/cosic.pdf.

[32] A. Back, *Hashcash - a denial of service counter-measure*, 2002. [Online]. Available: http://www.hashcash.org/papers/hashcash.pdf.

[33] A. Biryukov, D. Khovratovich, and I. Pustogarov, "Deanonymisation of clients in bitcoin P2P network", *CoRR*, vol. abs/1405.7418, 2014. arXiv: 1405.7418. [Online]. Available: http://arxiv.org/abs/1405.7418.