UiT
THE ARCTIC
UNIVERSITY
OF NORWAY

Faculty of Science and Technology
Department of Computer Science

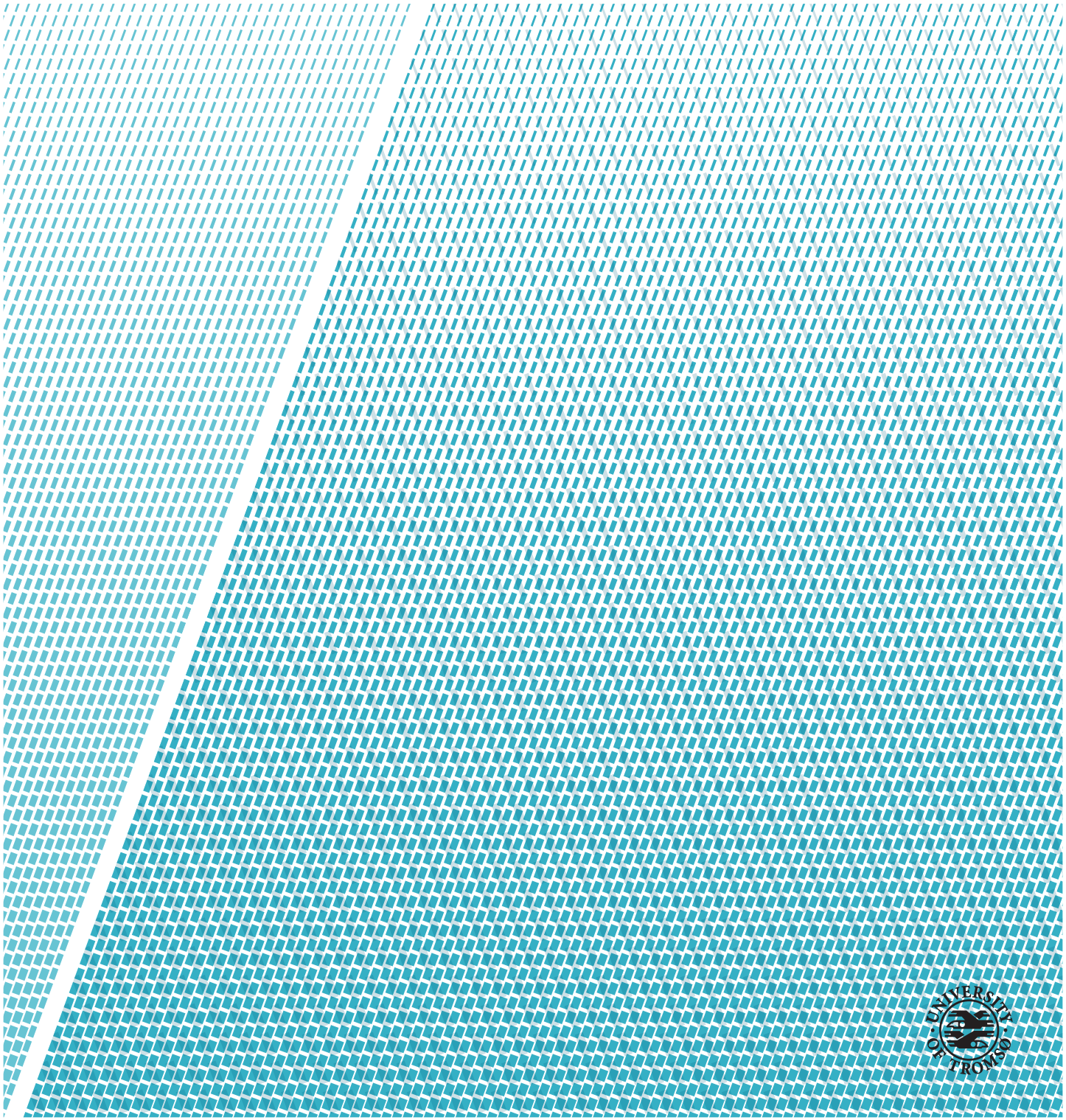
Trading Network Performance for Cash in the Bitcoin Blockchain

subtitle

—

Enrico Tedeschi

Master Thesis in Computer Science



Abstract

Nowadays blockchain systems are emerging and they are spreading each day more. Cryptocurrencies are the biggest example of a physical implementation of this protocol, having in 2012 more than 50 thousands transactions per day and reaching now in 2017 more than 350 thousands of transactions approved every day. In this thesis we evaluate the most famous blockchain system, the *Bitcoin blockchain*. Public blockchains have emerged as a plausible messaging substrate for applications that require highly reliable communication. However, sending messages over existing blockchains can be cumbersome and costly as miners require payment to establish consensus on the sequence of messages. The blockchain protocol requires an always growing size of the information stored in it so its *scalability* is the biggest problem. For that reason we collected data to be analyzed and stored in our own data frame, saving up to $x10$ space for the analysis.

This thesis will consider the network performance of the Bitcoin public ledger when used as a massaging substrate. From 2009 to 2017 a lot of analysis has been done on Bitcoin blockchain and meanwhile its block size limit changed multiple times, from 256 bytes up to 1 Mb, the Bitcoin price raised from ~ 0.7 \$ to more than 4.000 \$ and different papers were published discussing whether changing or not the block size limit or talking about the fees a miner could get from clients. We read and considered previous analysis on Bitcoin blockchain, we then present our own dataset, which contains a significant portion of the Bitcoin blockchain, updated at 09-2017, discuss our results and compare them with other evaluations from past years, then we also discuss how the fee paid to miners evolves during time and how much a client could pay for a faster approval time, plus we take into consideration transaction visibility, blockchain growth and fees paid to miners. From this we propose and evaluate, using machine learning techniques, three different cost prediction models for predicting bandwidth per Bitcoin cost of upcoming transaction. The models can be used by application to throttle network traffic to optimize message delivery. We also discuss and consider, according to the data obtained, whether the block size limit should be increased for an higher *throughput* or not.

People are using Bitcoin because it has a lower fee rate and no central authority,

we aim to find any possible relation between the fee paid from a transaction to a miner and the approval time of this transaction, plus we also noticed that the bigger is the blockchain size the more the system become centralized, since only few members, or nodes, of the Peer to Peer network can support and use the full blockchain.

Bitcoin blockchain has been analyzed with a blockchain analytics system, developed using Bitcoin's API and data were collected both by using the API and parsing `blockchain.info` HTML pages. A total of # transactions has been evaluated, more transactions than ever were considered before and useful information about the Bitcoin blockchain emerged. This thesis gives also a measurement about accuracy of data provided from `blockchain.info`.

Contents

List of Figures

List of Tables

My list of definitions

1 Introduction

In 1964, Paul Baran [22] described the differences between a centralized, decentralized and distributed network. Since then, the attention in developing systems moved from a centralized scheme to a distributed one, leaving most of the computation to every single processor in the network rather than at central coordinator. Such a change might be easy for systems that do require little security, for instance systems that do not require authentication and authorization of users. However, the more a system needs to be secure, the more the decentralization process might be tricky as it becomes very important to rely on some trusted central coordinator. Systems that more than others need to be secure are the one related to e-commerce, banking and trades, all systems that have to deal with money.

In 1983, David Chaum introduced the idea of digital cash [26]. In 1990 he founded *DigiCash*, an electronic cash company that closed because of bankrupt in 1998. After DigiCash, other systems such as *e-gold* (1996) and *PayPal* (1998) emerged. However, these systems allowed digital money transfer while they were still relying on a central authority. In 2008 Satoshi Nakatomo has presented Bitcoin [37], the first decentralized digital currency. Until 2008 e-commerce relied exclusively on financial institutions serving as trusted third parties. Those are involved in the electronic payments process and they have to guarantee consistency of the transactions and security of data.

Decentralized digital currencies are not dependent on any trusted third parties and they are built over a Peer to Peer (P2P) network where every component

has the same privileges. These systems allow money exchange without a central authority, which means lower fees, no geographical separation and global trust among users. After Bitcoin, more decentralized digital currencies emerged, in 2011 *Litecoin*, originally based on the bitcoin protocol, then in 2013 Gavin Wood has presented *Ethereum* [44] and in 2014 *Monero* currency was released.

The order of transaction is essential in any currency systems, however, establishing correct order can be problematic in decentralized cryptocurrency systems as they allow arbitrary nodes to join, including nodes that might be malicious. If arbitrary or Byzantine faults are allowed, the system might be left in an inconsistent or invalid state [33]. The ability to mask Byzantine faults has been implemented in various systems such as Byzantium [30], HRDB [42] and MITRA [34]. These protocol guarantees consistency of transactions having f faulty nodes, with a total of N nodes where $N = 2f + 1$ or $N = 3f + 1$, and a protocol like *Fireflies* [32] provides secure and scalable membership management and communication substrate in overlay network with Byzantine members.

1.1 Blockchains

To guarantee an order of transaction all these cryptocurrencies rely on the *blockchain* protocol. The need to tolerate malicious members was the reason for introducing the *blockchain* into cryptocurrency systems. A blockchain is essentially a distributed database of records or public ledger of all transactions or digital events that have been executed and shared among participating parties [28]. The fundamental principle behind the blockchain is that consensus on transaction ordering is based on contributed computational power rather than number of participants. The blockchain works by appending transactions into blocks. Every block is generated after a relevant computation, called *proof-of-work*, and each new block is appended to the public ledger of data, the blockchain, having in that way an ever growing chain of information containing every transaction ever happened. Blockchains essentially implements a distributed consensus protocols that enable a set of untrusted processes to agree on the content of an append only data structures, often referred to as ledgers. These ledgers are divided into blocks and linked together in sequence by hashes. They facilitate transactions between consenting individuals who would otherwise have no means to trust each other and deal with geographical separation and interfacing difficulties. This technology promises a highly resilient and communication substrate where messages are kept potentially for a long time.

Besides its use in cryptocurrency, blockchain technology opens up to several usages in different sectors such as trading, file storage, and identity management. Indeed it is already used by NASDAQ in its private socket market. If used in a P2P file sharing network, blockchain removes the need of a centralized data base. Moreover it allows users to create tamper-proof digital identities for themselves. Blockchain technology opens up to usages in several important sectors such as trading, file storage, and identity management.

There are several important limitations in current blockchain technologies. The most relevant is *scalability*, due to the steady growth of the blockchain. It should be also considered that decentralized cryptocurrencies operate in open (or permissionless) networks in which the ledger of data could be manipulated from arbitrary adversaries. According to the paper from University of Singapore [35], security of smart contracts have not received much attention yet. Since the only part not protected from cryptography is the *order of transactions* [10], an attacker would try to convince the network that a transaction occurred earlier than another one to gain money. The security bugs in smart contracts are classified as *Transaction-Ordering Dependence*, *Timestamp Dependence*, *Mishandled Exceptions* and *Reentrancy Vulnerability* [35]. In this thesis we refrain from explaining Bitcoin and its terminology in detail and refer the reader to already existing high-level [41, 25] or technical [37, 10]. description.

1.2 Problem Statement

While doing research, studying and reading papers related to blockchains, it turned out that the most urgent concerns are related to its scalability, performance and a profit optimization from participant in the whole system. In 2015, Möser and Böhme [36] said that Bitcoin may not be as cheap for consumers as it appears, and that Bitcoin users are encouraged to pay fees to miners, up to 10 cents (United States Dollar (USD)), per transaction, irrespectively to the amount paid. Always in 2015 Rizun [39] writes that the block size limit was set at one megabyte, corresponding roughly to three transactions per second but the transaction rate is over three hundred times larger than when the block size limit was introduced, and rising the limit is now being seriously considered. In 2016 Croman [27] announce that the current trend of increasing the block sizes on Bitcoin protends a potential problem where the system will reach its maximum capacity to clear transactions, probably by 2017.

These problems need to be taken into consideration. In our thesis we propose a longitudinal study on Bitcoin blockchain, analyzing most recent data. In particular, we discuss the scalability of the blockchain, how it affects the throughput and we present performance observations of the Bitcoin blockchain,

analyzed with a blcockhain analytic system developed for this purpose. We provide detailed insights and analysis on how Bitcoin's characteristics, such as fee, block size and reward to different miners involved have changed over time, and provide an updated view from the one proposed from Möser and Böhme [36] about the tolls and fees in the system. Furthermore, we analyzed the correlation between the fee paid from a transaction and its *latency*, or the time it takes to be visible in the whole network. Three different models are proposed to describe how applications best can spend money to improve network charactertics, this affects average bandwidth available to an application. By doing transactions-wise and block-wise experiments and analysis on the Bitcoin blockchain we state and focus on three major problems:

1. Scalability
2. Performance
3. Fees and Tolls

1.2.1 Scalability

Scalability and network performances are urgent concern in existing Blockchain-based cryptocurrencies [27]. According to Ethereum white paper [10], if Bitcoin would have the same amount of transactions of a VISA circuit, its blockchain would grow about 1 MB every 3 seconds, ~ 28GB per day, instead of the actual growth of ~ 0.12GB per day. In this thesis we discuss how much scalability affects centralization in Bitcoin network and how will be its impact in the next couple of years. Furthermore, with our analysis we aim to study how this growth is affecting the system's performance and fees and how much they change if the system capacity is filled with too many transactions to approve.

1.2.2 Performance

Centralized schemes, like VISA, are immediate while having a throughput of 2000 transactions/sec up to 56 thousands transactions/sec [27]. It is true that Bitcoin has lower fees than centralized currency schemes, but these properties come at a performance and scalability cost. In the paper from Croman [27], they claim that Bitcoin achieve a throughput of 7 transactions/sec constrained by the block creation time and the block size limit. In this thesis we want to analyze with new data from 2017 the network performance when it comes about throughput and transaction latency and see how much a change in the block size or block creation time might influence the system efficiency.

1.2.3 Fees and Tolls

Bitcoin strength always been the low fee if compared to the other centralized systems. But is this still true nowadays and are the fees still low and the system that cheap? The actual costs of the system are not extensively studied yet and Bitcoin might not be as cheap for costumers as it appears. During our analysis we could observe a remarkable increase of the fee paid from users. In our thesis we aim to find and prove the reasons that leaded to this scenario and they involve also scalability and performance problems. Our first concern while studying fees and tolls is to analyze whether is possible or not for applications to control available bandwidth given from the system by paying higher fees to miners.

1.3 Method / Context

In order to get enough information to analyze performance, make assumptions on scalability and fees in the Bitcoin network, data need to be retrieved from a source and then stored. We define three different approaches of data retrieval and all three have their pros and cons:

1. **Real time analysis:** It might be done by sniffing the traffic on the Bitcoin network in order to get real time data. For this analysis at least one full operating node needs to be implemented in the Bitcoin network. Pros are that, if you set two or more nodes distributed around the world, you can get a lot of useful information regarding the inner-node communication (otherwise impossible to obtain) and the block propagation time plus the orphaning rate in the system. Cons are that you need huge disk space to download the full ledger of data and the set up it extremely time consuming, plus, you have physical and geographical limitations in order to get up and running multiple nodes in different part of the world, without considering bandwidth and electricity costs related to it.

2. **Historical data retrieval through Bitcoin Core:** Another solution might be to use the client app of Bitcoin, Bitcoin Core. You can set up your node containing the full ledger locally, storing all the necessary data and have an historical view of the entire blockchain. Pros are that you have the whole view of the system. Cons are that, this whole view, still does not include the propagation time nor any miner's information, having to retrieve these knowledge separately. Plus, to set up the full node might take up to 4 days and the disk space required is in the order of hundreds of GB.

3. **Historical data retrieval through Application Programming Interface (API):** Third choice is to build a system locally and use Bitcoin APIs to gather data and generate information using those. Pros are that you have more flexibility when retrieving data, allowing you to store only useful information, saving a considerable disk space, also, the blockchain might be stored in intervals, which gives a faster retrieval, more disk space saved and still provides enough information to have a whole view of the system. Cons are that still you don't have information about block propagation time and orphaning and it might be difficult to analyze the blockchain in real time.

In this thesis we focus on the third method and we aim to analyze a considerable part of the blockchain and by doing a longitudinal study on it. A similar work was done in 2015 by Möser and Böhme [36], by analyzing tips and tolls in the Bitcoin blockchain, collecting data until 2014 and then analyzing more than 9 million of transactions. At that time there were a total of 100 thousands transactions per day, while today we count about 350 thousands on daily bases, so the retrieving part turned out to be more time consuming than expected. Despite that, we aim to collect even a larger portion of the blockchain, storing data smartly in a *data frame*, which allows us to save up to 10 times of disk space the blockchain actually requires. Then we analyze our collected data and with *machine learning* techniques we define models, discuss about the results and how much they can be reliable in a future-wise implementation. In our data frame we store more than 120 millions of transactions occurred in between April 2013 and September 2017. We used for the information retrieval APIs from `blockchain.info` combined with a HTML parsing on the same website for the information missing in the API. Our assumption is that we can get enough information about the Bitcoin blockchain by retrieving and analyzing only a portion of the blockchain, but having in that way a finer granularity than data represented on the Bitcoin website. In that way we aim to gain more information out of it. Moreover, sampling data from a single node in the blockchain gives statistics representative of the whole system.

1.4 Outline

2

Background

This chapter gives a brief introduction to the blockchain systems, with a particular focus on Bitcoin. The list of symbols used is showed in Appendix B. For further detail we refer to already existing high-level [25] or technical [44, 18, 7, 39, 37, 10] descriptions, but we will briefly explain the concept of mining and introduce the notion of payments and fees in Bitcoin system, how miners profit from mining, how a user could choose whether including fee or not in its transactions and how performance and scalability are changing the way miners select transactions. We will also describe about some machine learning techniques and data structures, plus we will take into consideration the most relevant studies and works that has been done in Bitcoin system regarding scalability, performance and fees.

Differences between are *centralized* and *decentralized* digital currencies are summarized in Table 2.1. As mentioned in Chapter 1, more decentralized digital currencies are emerging nowadays, and they aim to facilitate transactions between consenting individuals who would otherwise have no means to trust each other and deal with geographical separation and interfacing difficulties. According to Nakamoto [37] an electronic coin is defined as follows:

The Definition 1. An *eletronic coin* is a chain of digital signatures. Each owner transfers the coin to the next by digitally signing a hash of the previous transaction and the public key of the next owner and adding these to the end of the coin.

Table 2.1: Centralized vs Decentralized digital currencies

Centralized	**Decentralized**
Under the control of a central authority	Requires consensous among users to make changes
Users are dependent from the control given from the Certificate Authority (CA)	Users have full control over their assets/data
No need to enforce cryptography or to create keys	User's data is uniquely identified by private key
Only users allowed by CA can participate	Anyone can join and use ledger
Historic transactions can be changed by the CA	Historic transactions are unalterable and permanent
Can be run very efficiently using relational databases that fit the need of the applications	Inefficient for the cost of mining

Transactions are registered in *blocks* and all these systems use the *blockchain* technology to maintain a serial order of them. The blockchain is a consensus protocol that users run to maintain and secure a shared ledger of data and it is formed of all blocks, ordered by time and connected between each other with the *previous block* attribute. A transaction is approved only when it is included in a block, and the miner including these transactions is the one who has first solved the *proof-of-work*. Difficulty of proof-of-work is increased according to keep an average $\mathcal{T}$ of 10 minutes.

2.1 Mining

In a decentralized cryptocurrency network, transactions need to be approved by all the participant before being validated and confirmed. This validation requires high computational time and it might be expensive in matter of electricity and resources used.

The Definition 2. *Mining* is how transactions are validated and confirmed by the network.

To validate and confirm transactions, the network needs *miners*. According to the Bitcoin miner document [17], Bitcoin miners create new blocks by solving a proof-of-work problem that is chained through cryptographic proof of the previous block. The work and the effort spent to create new blocks is often

referred as mining [17]. The mining process involves identifying a value that when hashed twice with SHA-256, begins with a number of zero bits. If the amount of zeros needed for the creation is raised, the average work required increases exponentially, while a hash can always be verified by executing a single round of double SHA-256 [17, 40]. Each miner choose whether include a transaction into a block or not, and we believe that nowadays this inclusion strongly depends on the fee that this transaction has to offer. The first miner who solves the proof-of-work gains the sum of the fees (t_f) offered by each transaction included in the block, M, plus the block reward, R, which is now set at 12.5 B. After the new block is mined, this information is spread through the whole network and if this solution is propagated to the 50% + 1 of the nodes in the system then the block is validate and all the transactions in it are accepted and confirmed. Finally, miners in the network express their acceptance by moving to work on the next block, incorporating the hash of the accepted block.

2.1.1 Proof-of-Work

Proof-of-work is defined as follows [20]:

The Definition 3. *Proof-of-work* is a piece of data which is difficult (costly, time-consuming) to produce but easy for others to verify and which satisfies certain requirements.

Mining a block is difficult because the SHA-256 hash of a block's header must be lower or equal to a certain *target value* in order for the block to be accepted by the network. Producing proof-of-work is a random process with low probability of hitting the target value, so that a lot of trial and error is required on average before a valid proof-of-work is generated.

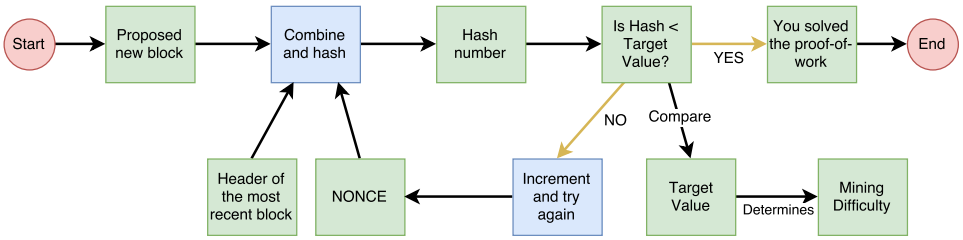


Figure 2.1: Diagram that shows how proof-of-work works.

Proof-of-work is a part of the mining process and a valid proof-of-work is determined by incrementing a nonce until a value is found that gives the block's hash the required number of leading zero bits. Once the hashing has produced a valid result, the block cannot be changed without redoing the work.

As later blocks are chained after it, the work to change the block would include redoing the work for each subsequent block [17]. The best chain is the longest one, so is the one that requires the greatest amount of effort to be produced, and majority consensus in Bitcoin is represented by this longest chain. In this way if the majority of the computing power is controlled by honest nodes, the honest chain will grow fastest and outpace any competing chains. Diagram in Figure 2.1 represents how the proof-of-work is performed, and it is inspired to the model showed in `bitcoinmining.com` [18].

2.1.2 Miners & Mining Pools

With paper money a government decides when to print and distribute money. Bitcoin does not have a central government and it uses a special software to solve math problems to get a certain amount of coins in exchange. This provides a smart way to issue the currency.

The Definition 4. A *miner* is a computer specifically designed to solve problems according to the proof-of-work algorithm and they are required to approve transactions.

Since miners are required to approve transactions, more miners means a more secure network. To become a Bitcoin miner nowadays you need to buy highly specified chips called Application Specific Integrated Circuits (ASIC). Informations about this hardware can be found at `bitcoinmining.com` [18]. In early days of Bitcoin (2009-2013) it was possible to mine with your computer CPU or high speed video processor card. Today that is no longer possible since custom Bitcoin ASIC chips offer performance up to 100 times the capability of older systems [18] and another reason is the coming of mining pools, bringing then a constant growth of the *mining difficulty*.

The Definition 5. A *pooled mining* combines the work of many miners toward a common goal.

Pools of miners or mining pools find solutions faster than individual member and each miner is rewarded proportionally with the amount of work it provides. Mining is an important and integral part of Bitcoin that ensures *fairness* while keeping the Bitcoin network *stable*, *safe* and *secure* [18].

2.2 Bitcoin & Fees

Miners generate new blocks by appending transactions in them. Every miner has a *mempool*, $\mathcal{N}$, containing the new, unapproved, n transactions. Miners are free to choose whether to accept or refuse a certain transaction $t \in \mathcal{N}$. Competitive miners will include transactions as long as the fee exceeds the marginal cost of inclusion. Production costs are fixed per block (but may vary between miners depending on access to technology and energy cooling) and the protocol defines a maximum block size Q. Because of that, the marginal cost of inclusion is zero if there are fewer unconfirmed transactions than the capacity left in the block. Competitive miners make positive expected profits only if transactions compete for space in the blockchain. Houy [31] argues that a maximum block size is necessary for the stability of Bitcoin. However in that way big mining pool might take advantage from less competition, having a centralization problem, since a transaction is willing to compete only if the capacity is reached. In that way if the system never reaches the capacity, raising the fee will not be helpful at all. Before 2015 transactions almost never required to compete for space in blocks, since the demand was less then the offer, but nowadays we have a totally different scenario. In 2012, the number of transactions confirmed per day reached peaks of 50,000, with an average of 15,000 transactions per day [7]. Nowadays we have an average of 250,000 transactions approved per day, with peaks of 350,000. Since the throughput limitations in the system, due to the block size limit Q and the average interval $\mathcal{T}$ set at 10 minutes and with such big increment of transactions to approve every day, miners could start to choose transactions with an higher fee density ρ or simply transactions that are willing to offer an higher fee rather than zero-fee transactions. The fact that the reward R has a 50% reduction every 210,000 blocks creates a market mechanism to find the price of Bitcoin transactions.

2.3 Previous Works

The main problems analyzed in the past years concerning the Bitcoin blockchain were related to scalability [39, 27], performance [27, 29] and fees [39, 36]. Already in 2014 researchers believed that these "fees" users were paying to miners were supposed to substitute miners' minting reward in the long run [36]. In 2015 discussions about how a rational Bitcoin miner should select transactions from his node mempool raised, and ideas about maximizing miner's profit when creating a new block emerged. Equations that mine to calculate the cost of mining $\langle C \rangle$ and the miner's profit $\langle \Pi \rangle$ were presented [39] and the concept of *orphaning* was introduced. Scalability is still nowadays a huge concern, causing relevant bottlenecks in Bitcoin which limit the ability of its current peer-to-peer overlay network to support substantially higher throughputs and

lower latencies [27].

2.3.1 A Transaction Fee Market Exists Without a Block Size Limit

A pressing concern exists over the ramifications of changing (or not) a Bitcoin protocol rule called *block size limit*. This rule sets an upper bound on the network's transactional capacity, or *throughput* (γ). The limit is currently set at 1 MB, corresponding roughly to three transactions per second. This limit set by the blockchain protocol, allows miner to include up to 1 MB of transactions selected from their memepools. When this limit was set, it was over eight hundred times greater than what was required. However in 2015, blocks were filled near capacity and users experienced delays. In 2015 the transaction rate was over three hundred times larger than when the block size limit was introduced. To improve performance then a raise in the block size limit should be implemented, but in that way transactions will not compete anymore for space in the block, creating an unhealthy transaction fee market. Miners should then include transactions in a manner that maximizes the expectation value of their profit [39]. For that reason *Miner's Profit Equation*, *Mempool Demand Curve*, *Block Space Supply Curve* and the concept of fee density were introduced. Every time a block is mined, the miner expects to generate a revenue $\langle V \rangle$ at hashing cost $\langle C \rangle$, having then a profit $\langle \Pi \rangle$ as follows:

$$\langle \Pi \rangle = \langle V \rangle - \langle C \rangle. \tag{2.1}$$

where the hashing cost is represented in Equation 2.2:

$$\langle C \rangle = \eta h \mathcal{T}. \tag{2.2}$$

So the hashing cost $\langle C \rangle$ is directly dependent from the miner's individual hash rate, h, the cost per hash, η, and the creation time, $\mathcal{T}$. The revenue $\langle V \rangle$ is calculated with the amount he would earn if he won the block (reward plus fees, $R + M$) multiplied by his probability of winning (ratio between miner's hash rate, h, and hash rate of Bitcoin network, H), considering also the orphaning rate showed in Equation 2.4.

The Definition 6. A block B_1 generated at a timestamp t_1 is *orphan* if it does not get approved by the network because another block B_2 with a greater timestamp $t_2 > t_1$ which propagates faster, gets approved instead.

So the expected revenue is showed in Equation 2.3:

$$\langle V \rangle = (R + M) \frac{h}{H} \left(1 - \mathbb{P}_{orphan}\right). \tag{2.3}$$

Where $\mathbb{P}_{orphan}$ increases with the amount of time a block takes to propagate in the network. Indeed, if τ is the block propagation time, the probability of orphaning is defined as:

$$\mathbb{P}_{orphan} = 1 - e^{-\frac{\tau}{\mathcal{T}}}. \tag{2.4}$$

We can define now the miner profit equation as follows:

$$\langle \Pi \rangle = (R + M)\frac{h}{H}e^{-\frac{\tau}{\mathcal{T}}} - \eta h \mathcal{T}. \tag{2.5}$$

A *rational miner* selects which transactions to include in his block in a manner that maximizes the expectation value of his profit. This selection is explained with the Mempool Demand Curve and the Block Space Supply Curve. According to the size limit, a block can select a $b \leq n$ transactions from $\mathcal{N}$ to create a new block $\mathcal{B} \subset \mathcal{N}$. Studies on inclusions brought to the conclusion that a block first includes transactions with an higher *fee density*, ρ, which is the ratio between the *transaction fee*, t_f, and the *transaction size*, t_q. To construct the mempool demand curve, is necessary to sort the mempool from greatest fee density to least and its formula is showed in Equation 2.6.

$$M_{demand}(b) \equiv \sum_{i=1}^{b} t_{fi}, \tag{2.6}$$

and the sum of each transaction's size in bytes, form the block size Q:

$$Q(b) \equiv \sum_{i=1}^{b} t_{qi}. \tag{2.7}$$

The mempool demand curve represents then the maximum fee, $M_{demand}(b)$ a miner can claim by producing a given quantity $Q(b)$ of blockspace. The size of the block a miner elects to produce controls the fees he attempts to claim, $M(Q)$, and the propagation time he chooses to risk, $\tau(Q)$. The block space supply curve represents the fees a miner requires to cover the additional cost of supplying block space Q. This cost grows exponentially with the propagation time. The equation which represents this curve is the one showed below:

$$M_{supply}(Q) = R\left(e^{\frac{\Delta\tau(Q)}{\mathcal{T}}} - 1\right), \tag{2.8}$$

where $\Delta\tau(Q) \equiv \tau(Q) - \tau(0)$. In order to have a transaction fee market without a block size limit, to maximize his profit, the miner constructs a mempool demand curve and a space supply curve. The block size Q^* where the miner's surplus, $M_{demand} - M_{supply}$, is largest represents the point of maximum profit.

2.3.2 Trends, Tips and Tolls

The Bitcoin protocol supports optional direct payments from transaction partners to miners, also called *fees* (t_f). Acknowledging their role for the stability of the system, the right level of transaction fees is a hot topic of normative debate. The actual costs of the system are not extensively studied yet and Bitcoin may not be as cheap for consumers as it appears. The definition of transaction fee is encoded as difference between the sum of all inputs and the sum of all outputs of a transaction (Equation 3.3). Previous analysis on the Bitcoin blockchain such the one presented by Möser and Böhme in 2015 [36], show that transaction fees are lower than 0.1% of the transmitted value, which is significant below the fees charged by conventional payment systems, and at the time of analysis the hard size limit did not (yet) significantly drive the level of transaction fees. However trends for the fees paid per transaction over time noticeably changed. The trend of 0-fee transactions had a drop after April 2012 leaving space to 0.0005 Ƀ fee until May 2013. After that, until the beginning of 2015 the trend for fees was of 0.0001 Ƀ and 0.0002 Ƀ, with peaks of 0.001 Ƀ. Generally, there seem to be two main reasons for shift in trends: changes to the Bitcoin reference implementation and actions by large intermediaries in the ecosystem. The emergence of 0.0005 Ƀ fees in June 2011 can be mapped to the release of version 0.3.23 of the Bitcoin Core client, which reduced the default transaction fee from 0.01 Ƀ to 0.0005 Ƀ. The raise of these last transaction fees in the second quarter of 2012 is probably due to the launch of the gambling website *SatoshiDice* [16]. On May 2013, version 0.8.2 of Bitcoin Core was released. In the past years, during a period in between 2011 and 2015 there is a small share of transactions that did not offer fee to miners, most of them offered default fee amount but some of them were even willing to pay a higher fee, a tip. A plausible reason is that paying more in fee leads to a faster confirmation. Analysis on the blockchain in 2015 [36] state that half of all zero-fee transactions had to wait more than 20 minuets for their first confirmation. In contrast to that, paying a 0.0005 Ƀ fee lead to an inclusion into a block in half of the time. 10% of all zero-fee transactions took almost 4 hours to confirm, in contrast to 40 minutes for transactions paying a 0.0005 Ƀ fee. The difference between paying 0.0005 Ƀ or 0.001 Ƀ fee is not as pronounced, but the difference in medians are still statistically and economically significant. Analysis on pool behavior regarding a possible systematic exclusion of zero-fee transactions has been done. Shares have shifted between pools quite extensively. In 2013, BTC Guild had a market share of up to 40%, in 2014 both GHash.IO and Discus Fish ousted this pool. Also, the share of other pools has risen in 2014. Previous incumbents like Slush or 50BTC have lost popularity. Possible reasons include economic and technical factors, like pool fees, service availability, or robustness against attacks. Given the dominance of a few mining pools, evaluations whether some pools systematically enforce fees has been made. The results show that two pools, Discus Fish and Eligius,

have a considerably higher share of blocks without any zero-fee transaction, with 30.6% for Eligius and 62.5% for Discus Fish, in contrast to an average of 14.4%. Over than that though, there is no clear evidence for enforcement of strictly positive transactions fees.

2.3.3 Bitcoin Performance Limitation

Can decentralized blockchains be scaled up to match the performance of a mainstream payment processor? What does it take to get there? In 2016 the Bitcoin blockchain took 10 min or longer to confirm transactions, achieving 7 transactions/sec maximum throughput. Visa credit card confirms a transaction within seconds and processes 2000 transactions/sec on average with peaks of 56, 000 transactions/sec. Bitcoin community has put forth various proposals to modify the key systems parameters of block size and block interval. Anyway, Croman state that such scaling by reparametrization can achieve only limited benefits [27]. This because because Bitcoin generates a lot of network traffic, due to its decentralization, this leads having a lot of peers in the network and they all have to interact. To ensure that most of the nodes in the overlay network have sufficient throughput we follow two guidelines:

- **Throughput limit.** The block size should not exceed 4 MB given 10 minutes average block interval. Corresponding at maximum 27 transactions/sec.
- **Latency limit.** The block interval should not be smaller than 12 seconds.

The maximum throughput of 3-7 transactions/sec is a number constrained by the 1 MB block size Q and the 10 minutes creation time $\mathcal{T}$. About the propagation time τ, measurement in 2016 [27] show that 10%, 50% and 90% block propagation times are 0.8 seconds, 8.7 seconds and 79 seconds respectively, with an average block size of 540 KB. Projecting to a 1 MB block size, τ would be 2.4 min, 15.7 sec, and 1.5 sec respectively. In that way an effective throughput on the network could be calculated as follows:

$$\text{X\% effective throughput} = Q/(\text{X\% block propagation delay}).$$

Having for X = 50% an effective throughput of 496 Kbps, equals to 248 tx/sec, and for X = 90% an effective throughput of 55 Kbps, equals to 26 tx/sec. It follows that Q, and interval $\mathcal{T}$, must satisfy Equation 2.9:

$$\frac{Q}{\text{X\% effective throughput}} < \mathcal{T} \tag{2.9}$$

Consequently, for a 10 minutes (or shorter) block interval, the block size should not exceed 4 MB for X = 90%; and 38 MB for X = 50%, corresponding to a throughput of at most 27 transactions/sec. To improve the system's latency it

could be enough to reduce the block interval. To maintain effective throughput that would also require a reduction in the block size. Propagating a block smaller than 80 KB would not make full use of the network's bandwidth, as latency would still be a significant factor in the block's propagation time. To propagate a 80 KB block to 90% of the nodes would take roughly 12 seconds. In conclusion, to retain at least 90% effective throughput and fully utilize the bandwidth of the network, the block interval should not be smaller than 12 seconds.

2.4 Machine Learning

Once a new node first joins the Bitcoin network, it has to download all the useful information to process transactions and verify them. These information consist in a huge quantity of data, nowadays ~ 125 GB, and it grows continuously over time. A full node might require more than four days to download the entire ledger. Once all data are collected is necessary to have a way to store them and to get the right information out of them. Many data structures allow to store and have ordered huge amount of data and several machine learning techniques grant to infer more information out of them. While applying machine learning techniques to your dataset might be recommended to use data structures such as *trees* or *data frames* which are more likely to be used in machine learning algorithms.

2.4.1 Training Dataset

Having a huge dataset implies having a big amount of data with different type of attributes. These attributes might have *numerical* or *categorical* values.

Numerical: Data that can be measured, quantified. For example the fee t_f paid to a miner in Bitcoin Currency (BTC).
Categorical: Data which represent characteristics. For example the name of the miners in the Bitcoin system.

These data might be saved in a data structure such as a data frame, having then a sample for each row and every column representing a different attribute for this sample. Once data are collected, more information could get out of them, and this data might be trained to get predicting models or other useful material. Training and predicting models involves different types of variables:

Target variables: the variable that you are attempting to predict, represented with Y;

Predictors: variables that you can use to make the prediction, represented with X.

$$X = \begin{bmatrix} x_{11} & x_{12} & \dots & x_{1n} \\ x_{21} & x_{22} & \dots & x_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ x_{m1} & x_{m2} & \dots & x_{mn} \end{bmatrix} \qquad Y = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_m \end{bmatrix} \tag{2.10}$$

When the targets are real numbers, the problem of training those data is called *regression problem*. If the targets are two-valued the problem is called *binary classification problem*. We need to predict then every $y_i \in Y$ using every row $x_i \in X$ and evaluate the performance of our predictions. Good performance means using the attributes x_i to generate a prediction that is *close* to y_i. For a regression problem where y_i is a real number, performance is measured in terms like the mean squared error (MSE) (Equation 2.11) or the mean absolute error (MAE) (Equation 2.12) [23].

$$\text{MSE} = \frac{1}{m} \sum_{i=1}^{m} (y_i - pred(x_i))^2 \tag{2.11}$$

$$\text{MAE} = \frac{1}{m} \sum_{i=1}^{m} |(y_i - pred(x_i))| \tag{2.12}$$

However, since MSE is in squared units, the Root Mean Square Error (RMSE) is usually a more usable number to calculate. If the problem is a classification problem, then other measure of performance must be used. One of the most used is the *misclassification error*. Classification problems generally revolve around misclassification error rates and, usually, algorithms for such kind of problems can present predictions in the form of a probability rate for the attributes rather than an attribute itself.

Before and during the analysis we made assumptions on how attributes and values might change if other attributes were increased or decreased. These assumptions are showed in Table 3.2. To be able to see whether an attribute is related to another one we might measure it by using *Pearson's correlation*. Pearson's correlation coefficient is defined for two equal length vectors u and v:

$$u = \begin{bmatrix} u_1 \\ u_2 \\ \vdots \\ u_n \end{bmatrix} \qquad v = \begin{bmatrix} v_1 \\ v_2 \\ \vdots \\ v_n \end{bmatrix} \tag{2.13}$$

First subtract the mean value of u from all the elements of u, and the same with v, then we generate Δu and Δv as follows:

$$\overline{u} = avg(u)$$

$$\overline{v} = avg(v)$$

$$\Delta u = \begin{bmatrix} u_1 - \overline{u} \\ u_2 - \overline{u} \\ \vdots \\ u_n - \overline{u} \end{bmatrix} \qquad \Delta v = \begin{bmatrix} v_1 - \overline{v} \\ v_2 - \overline{v} \\ \vdots \\ v_n - \overline{v} \end{bmatrix} \tag{2.14}$$

The Pearson's correlation between u and v is defined as follows in Equation 2.15 [23]:

$$corr(u, v) = \frac{\Delta u^T \times \Delta v}{\sqrt{(\Delta u^T \times \Delta u) \times (\Delta v^T \times \Delta v)}} \tag{2.15}$$

2.4.2 Pandas Data Frame

In machine learning is important to pick one data structure and store your data in it for the analysis and testing. We focus on explain Pandas data frame [15] since is the data structure we chose for this thesis. Pandas is a Python package providing fast, flexible, and expressive data structures designed to make working with "relational" or "labeled" data both easy and intuitive. It aims to be the fundamental high-level building block for doing practical, real world data analysis in Python. One of the primary data structure that Pandas provides is *DataFrame*. A data frame is a two-dimensional size-mutable, potentially heterogeneous tabular data structure with labeled axes (rows and columns). Arithmetic operations align on both row and column labels [15]. In our case each sample is a transaction T, and if we consider a data frame D like a set created from the union of all m transactions retrieved, $D = \{T_1 \cup T_2 \cup \cdots \cup T_m\}$, then we have the following representation of D:

$$D = \begin{bmatrix} T_1 \\ T_2 \\ \vdots \\ T_m \end{bmatrix} \tag{2.16}$$

and if every transaction T has n attribute, $T = \{a_1, a_2, \ldots, a_n\}$, we can finally represent our data frame as follows:

$$D = \begin{bmatrix} a_{11} & a_{12} & \ldots & a_{1n} \\ a_{21} & a_{22} & \ldots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & \ldots & a_{mn} \end{bmatrix} \tag{2.17}$$

Furthermore, operations such as groupby, mean, median, sum are well managed in Pandas data frame making it easy to manage your data. In Appendix C code for these operations is showed. This is just a theoretical explanation of data frame and we focus on a practical explanation on our data structure in Chapter 3.2.2.

2.4.3 Visualizing Data

When analyzing big datasets, finding a smooth and nice way of representing information of them might be tricky. For example, if you have millions or billions of samples it will result almost impossible to get any information by simply plotting these data. When analyzing longitudinal data for example, information might be showed daily, monthly or even yearly-wise, then the *mean* for every portion is applied. However, bigger is the dataset and most likely it will contain *outliers*. In this case, calculating the mean on those data might be misleading, that is why in a longitudinal study should also be considered whether to apply a *median* value for every portion rather than the mean, since the mean is particularly susceptible to the influence of outliers.

3

Blockchain Analytics System

To understand digital cryptocurrencies, in particular Bitcoin, and to study existing systems, a complete data analytic system was designed and implemented. Using this system, we were able to run a longitudinal study on the Bitcoin blcokchain, collecting useful information from 2014 to 2017, analyzing more than 120 millions of transactions corresponding to the 50% of the entire blockchain in between that period. Previous studies [36, 29] never analyzed such large portion of the blockchain and we think that our analysis might be a breakthrough for studying cryptocurrencies as a stand alone subject. This chapter details our experimental setup and explains how our studies on the Bitcoin blockchain took place, clarifying how we analyzed the APIs from `blockchain.info` to get the best out of them and illustrating how we collected and manipulated our data. Then we describe how we decided to plot our data and why. In Chapter 4 again we point out our solutions compared to our assumptions and our conclusions.

As mentioned in Chapter 1.3 we opted for building a small data analytics system locally rather than using Bitcoin Core client or running full nodes in the Bitcoin network. By doing that we had the freedom and the flexibility to retrieve and store only the information we needed for our purpose, and considering how much the blockchain scales it turned out to be one of the most valid points. Furthermore we thought that the effort for setting one or more nodes up and

running in the system was far more than the benefits we would have gained by only allowing us to have a better esteem of the block propagation time while we are using the one evaluated first in 2013 by Decker and then in 2015 from Croman [29, 27], and we can consider them reliable enough for the purpose of our work. Plus we excluded from the beginning the idea of using Bitcoin Core, since it forces you to download and keep updated the ledger of data still without providing information about propagation time nor miners.

3.1 System Architecture

The blockchain analytics system architecture is showed in Figure 3.1. Information coming from the Bitcoin network are collected in data bases and web services such as `blockchain.info` [7] or `coindesk.com` [5]. We use then their web services, REpresentational State Transfer (REST) APIs and HTTP parsing to get these information. Our system then collects and gathers data locally and saves them in our data frame D. After that data are ready to be trained and plotted.

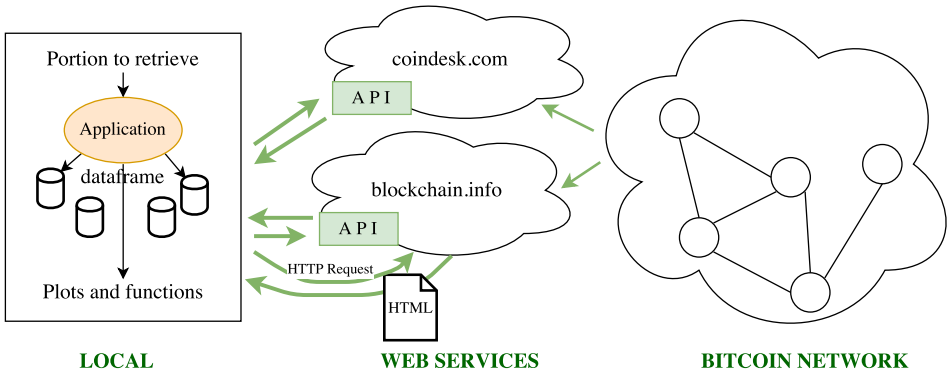


Figure 3.1: Overall architecture of the blockchain analytics system, considering how information and data travel in the network.

As Figure 3.2 shows, the information retrieval works in a combined way, *block-wise* and *transaction-wise*, then, information gathered from both sources are written and stored in D. The analytic system collects information from:

1. **JavaScript Object Notation (JSON)** file sources: files processed block by block and for each block transaction by transaction, then information are added to D.
2. **HTML** file sources: files obtained via an HTTP request. The parsed information are directly added to D.

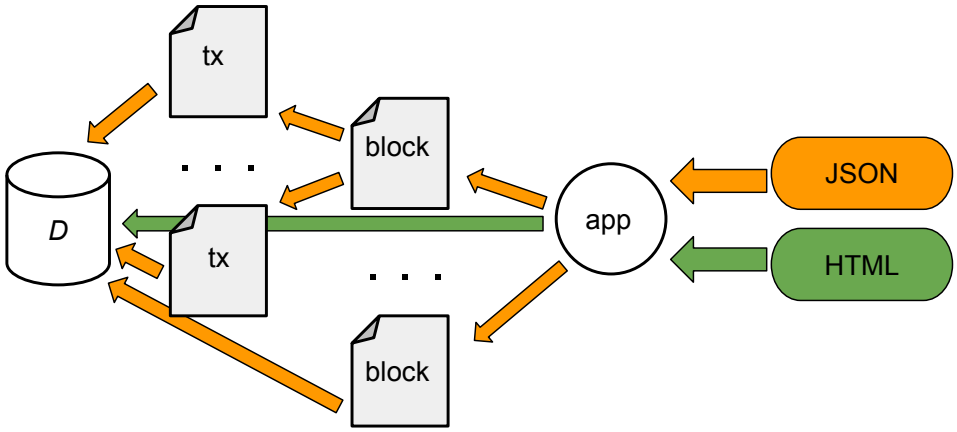


Figure 3.2: How data and informations are gathered in our data frame D.

3.2 Blockchain Data Sources

Many websites like `tradeblock.com` [21] or `blockchain.info` [7] are observing the Bitcoin blockchain every day. These websites provide also remote API that allows you to retrieve information about blocks and transactions directly from the website through REST APIs. Furthermore, websites like `coinbase.com` provide useful information about the money exchange price and they arrange API along with libraries like the one we used called *forex-python* [12]. One of the most popular system for analyzing Bitcoin data is *Bitcoin Core* [3]. However, the client requires users to download the full ledger of data before any analytics can take place, which at time of writing is 125 GB, with a total disk space required of 250 GB. Since we wanted to optimize and be able to select data we wished to analyze and we did not have the need to run a full node, we developed a system for blockchain retrieval and analysis. This system allows to fetch a portion of the blockchain by storing data in Pandas data frame [15]. This allows to save one order of magnitude disk space while still storing all the information needed for our purpose. Finally, the system displays all the knowledge acquired, analyzing data and applying machine learning technique on those, using Python libraries such as *matplotlib* [13] and *seaborn* [43]. Even though we know Python is not common for its computational speed, it provides nice access to the Bitcoin blockchain by using APIs arranged by `blockchain.info`, allowing the retrieval of all the information stored in a block and transaction, plus its enormous amount of libraries grants a vast flexibility on which machine learning algorithms to apply or whatever plotting system to use.

Table 3.1: Data sources and information gathered

Source	Entity	Information
Blockchain.info APIs	Block, Transaction	B_{ha}, t_{ha}, B_h, B_h, $\mathcal{T}$, Q, $t_{in}, t_{ou}, t_q, B_{epoch}, t_{epoch}$
Blockchain.info HTTP parsing	Miner	B_{mi}
Coindesk.com	Price	USD and Ƀ value
Data frame D	Block, Transaction, Miner, Price	τ, t_f, t_l, γ, ρ, $t_{\%}$

3.2.1 Data Retrieval

The architecture showed in Figure 3.1 gives informations on how different data are gathered in our system. The system imports Bitcoin APIs and uses them for data retrieval. The website `blockchain.info` also provides REST APIs, in that way requests made to a resource's Universal Resource Identifier (URI) will elicit a response that may be in eXtensible Markup Language (XML), HTML, JSON or some other defined format. In our case, REST APIs from Bitcoin return JSON data. The website `blockchain.info` is monitoring 24-7 the blockchain, producing graphs and statistical analysis about the actual Bitcoin network. The local application is monitoring these data as well, producing graphs that are not taken into consideration from this website, using a finer granularity that represents the data, allowing us to an in-depth analysis on those. For our analysis we select a time range from #April 2013 to September 2017, considering more than 120 millions of transactions and 100 thousands of blocks. Earlier than 2013, analysis on the system were already performed and evaluated [27, 31, 36, 39] plus the popularity of the system before 2011 was low and interpreting this early data would be not useful and self-defeating in order to understand system behavior. To have an overall of the entire Bitcoin network we combine data from different sources. As Table 3.1 shows, we use both, `blockchain.info` APIs and HTTP parsing on it to gather information about blocks, transactions and miners, plus we used `coindesk.com` APIs [5, 12] to have information about the pricing of USD and Ƀ. Unfortunately APIs from `blockchain.info` do not allow us to retrieve every information we need, knowledge regarding miners is not included in the blockchain, so we need to get these information by parsing web pages on `blockchain.info`. Finally, other useful information and derived data such as throughput γ or transaction fee t_f are obtained using our data structure D (Chapter 3.2.2), containing both, these new information and the previously collected ones. An example of data retrieval is showed in Appendix C.6 and the retrieved block's and transaction's structures by using RESTful APIs are showed in Appendix C.7, C.8.

First idea, Blockchain Splitted in Portions

During our analysis we evaluate different possible ways of data retrieval. Because of its relevant size and its fast growth, we considered the blockchain like an object and we thought that analyzing different parts during time following a certain pattern would give us an accurate overall of the whole system. We started retrieving data by dividing the blockchain in k portions. Let's suppose that m is the last block's height, then we divide all in k portions, having $p = m/k$. While retrieving b blocks, they are retrieved and stored in the following way:

$$\langle 1 \dots b \rangle, \langle p \dots p + b \rangle, \langle 2p \dots 2p + b \rangle, \dots, \langle kp \dots kp + b \rangle.$$

Retrieving blocks in such portions gives statistics representative of the whole system. Like Figure 3.3 shows, every new b blocks added to the blockchain will be added respectively to the indexes:

$$ip + b \ \Big\{ \text{for } i = 0 \text{ to } k. \tag{3.1}$$

Retrieving data in this way gave us a big space saving and still relevant infor-

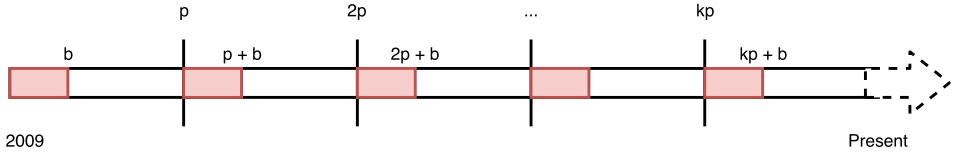


Figure 3.3: Science of Bitcoin blockchain, the entire blockchain considered as an object and divided in k portions. One of the first attempts of data retrieval on Bitcoin blokchain.

mation about the whole system. However, data collected using this methods weren't enough for our purpose of longitudinal study and analysis, but this technique gave us a spark for the next data retrieval attempt.

Fragmented Blockchain

In the other idea, to collect data we use a way smaller interval between each portion retrieved, called also jump or J. Plus we do not retrieve the earliest part of the blockchain, the one from 2009 to 2013. With retrieval of 10 blocks per time, according to the granularity of J we can have a more or less precise analysis. To balance disk space and precision we choose a $J = 10$ blocks and a fixed $b = 10$ of blocks retrieved each time, which means that every 10 blocks retrieved there is a jump of other 10 blocks, ~50% of the entire blockchain but analyzed daily. Figure 3.4 shows how our last data fetching has

been implemented, so D is generated accordingly, collecting transactions for ~100 min then ~100 min of gap. By doing that we manage to collect relevant information from 2013 to 2017 by only using ~25 GB of disk space. Then, the

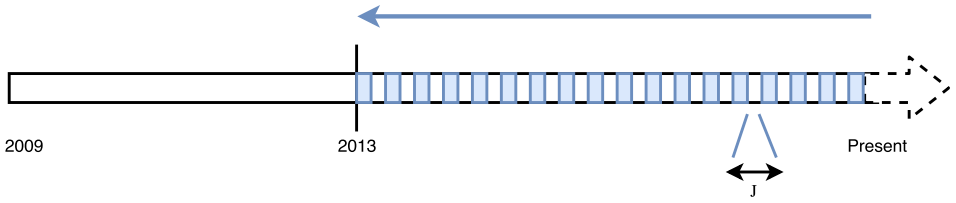


Figure 3.4: Fragmented blockchain divided in smaller portions with a jump of J blocks.

pattern we followed to store transactions it is the following:

$$\langle m \ldots m{-}b\rangle, \langle m{-}b{-}J \ldots m{-}2b{-}J\rangle, \ldots, \langle m{-}(k{-}1)b{-}(k{-}1)J \ldots m{-}kb{-}(k{-}1)J\rangle.$$

And we can write the indexes for each portion that has to be retrieved according to Equation 3.2, following the format $\langle \text{start} \ldots \text{end}\rangle$:

$$\langle m - ib + b - kJ + J \ldots m - ib - iJ + J\rangle \left\{ \text{for } i = 1 \text{ to } k. \right. \tag{3.2}$$

Having m as the height of the most recent block retrieved and k number of portions. An example of this retrieval method is showed in Appendix C.9.

3.2.2 Data Organization

Figure 3.2 shows how data, once retrieved, are collected and gathered in our analytics system. The Pandas package makes possible to read data into a specialized data structure called *data frame* [15]. As argued by Michael Bowles [23], you can think of a data frame as a table or matrix-like structure oriented with a row representing a single case or observation and columns representing particular attributes. We decided to use a data frame structure and not a matrix one becuase even though the data frame structure is matrix-like, its elements in various columns may be of different types. For statistical problems, the matrix is too confining because statistical samples typically have a mix of different types, plus, Pandas package makes it possible to automate the steps of calculating *mean*, *variance*, *median*, *group-by* elements and it offers a nice and smooth control over your data.

Finally, once data are retrieved and collected, they are saved in a data frame structure that we call D. Derived data such as γ or ρ, useful for the blockchain analysis, are obtained using D. This data structure is divided in multiple files with an overall size of ~25 GB and it contains both numerical and categorical

variables, it saves an average of 5 times disk space compared to the same information you get by downloading the full ledger, reaching peaks of 10 times for some blocks. In D information are stored in a bi-dimensional table, with transaction samples represented in rows and their attributes in columns. If we consider a set of attributes as a transaction, T, and a set of transactions D, then we have $T = \{a_1, a_2, \ldots, a_m\}$ and $Y = \{T_1 \cup T_2 \cup \cdots \cup T_n\}$. Our data frame D will be then a table with $|D| = (m \times n)$ and structured in the way showed in Equation 2.17.

Once defined the structure for our dataset D, we consider a set T having the following attributes (Appendix B):

$$T = \{t_{ha}, t_{in}, t_{ou}, t_f, t_q, t_{\%}, t_l, Q, \mathcal{T}, B_h, B_{epoch}, B_t, B_{ha}, B_{mi}\}.$$

We structure the way our data are stored for doing subsequential sampling. This is the reason why we stored transactions ordered by block and each block ordered by height/epoch, in that way we have the data set ordered by block creation with all its transactions in it and it will be much easier to make analysis or sampling data following a chronological order. Summarizing, our data frame allows us to collect all the useful information without storing the whole raw blockchain like current systems for blockchain analysis do, see *BitcoinCore* [3]. Raw data are collected in JSON format, analyzed, printed in our data frame and then deleted. This gave us up to $x10$ space saving on the blockchain, storing in a 1 GB data frame what the raw files from `blockchain.info` store in 10 GB space. Plus, we believe that analyzing a small percentage of the blockchain everyday would give us relevant information about the whole system. Due to time limitations we could collect only 50% of the data, every day.

3.2.3 Data Visualization

Once data are retrieved and saved in our data structure D, is finally possible to get the information we need out of them. Even though Pandas offers a pretty plotting, visualizing big data might be time consuming and, if the right information are not considered, poorly useful. Once data are ready to be analyzed, the first step is to determine the outliers. A spontaneous question that came up in our minds is: *how to deal with outliers?* We could segregate them out and train on them as a separate class or easier, instead of calculating the mean value, which is extremely sensitive to outliers, we calculate the median value for our data. Dealing with categorical attributes instead, as the number of attributes grows, the complexity of handle them mounts, also, most of binary tree algorithms, which are the basis for ensamble methods, have a cutoff on how many categories they can manage [23]. Random Forest package written by Breiman and Cutler has a cutoff of 32 categories [24]. Our only

categorical variable is B_{mi} so we did not deep in the analysis on categorical attributes.

To visualize our data we use Pandas libraries as well as *matplotlib* and *seaborn* [15, 13, 43]. We mostly use seaborn to plot data considering our categorical values since it offers nice methods for this kind of analysis. For example, we use it to calculate how major mining pools change their number of transactions approved over time, or to calculated the ρ for every mining pool. We also use it to calculate the linear regression and the Pearson's coefficient for every attribute $a \in D$. Pandas was useful to use area plots and pie plots on attributes, so wherever was necessary to manipulate our data frame D even further.

One example of data manipulation and visualization on our dataset D regards the calculation and classification over time of the fee density ρ. From D we collected data we needed and we generated another data frame D'. In that way we have a new data frame with a new set of attributes X', generated accordingly to the transformation we need, that for convenience we call $\xrightarrow{\lambda}$, having then

$$D \xrightarrow{\lambda} D',$$

$$T \xrightarrow{\lambda} X'.$$

So the transformation $\xrightarrow{\lambda}$ brings a new set of attributes with new data in it. This new X' contains a set of numerical values categorized, regarding ρ, plus the related date and the total number of transactions approved in the following way:

$$X' = \{0, <50, <100, <200, <300, >300, \text{date}, \text{total}\}.$$

Another use of data manipulation for visualization is the one we apply to the epoch. We want, accordingly to make our data more readable, to transform our epoch in date time and then get only the information we need regarding the day, the month or the year. By doing that we have to apply a function to our data frame D that hits one attribute (in this case the epoch, B_{epoch}), converts it in date time and then parses it to get information about the day, month or year. The listing is showed in Appendix C.5.

3.2.4 Derived Data

As shown In Table 3.1 our data frame D gives us information about derived data from the other data collected. This section aims to explain how they are calculated and why they are so important. We first focus on calculating the

exact amount a transaction has to pay in order to get the payment processed and approved, the transaction fee, t_f. This fee is a difference between the sum of all input t_{in} and the sum of every output t_{ou} of a certain transaction t. In that way, if n is the number of input and m the number of output, t_f is calculated according to Equation 3.3 and measured in BTC (B).

$$t_f = \sum_{i=1}^{n} t_{in_i} - \sum_{i=1}^{m} t_{ou_i}. \tag{3.3}$$

Next, we evaluate for every transaction, the time it takes since its creation, to be visible in the whole network and ready to be spent again. We call it transaction latency, or t_l. This value is calculated following Equation 3.4 and is the difference between two epochs, the first is the block creation time in which a transaction t is approved, the second is the transaction timestamp, so when it first was created. t_l is measured in seconds.

$$t_l = B_{epoch} - t_{epoch}. \tag{3.4}$$

Another relevant derived information using D is the throughput of the Bitcoin network, or how many transactions it can approve per second. We measure it according Equation 3.5, we call it γ and the unit used is tx/sec.

$$\gamma = \frac{t_B}{\mathcal{T}}. \tag{3.5}$$

As shown by Rizun [39] information regarding fee density is important. This value, that we represent with ρ, could be an important factor for miners to chose whether include or not a transaction in their next block and indicates how many BTC per bytes a transaction t has to offer. Equation 3.6 shows how ρ is calculated.

$$\rho = \frac{t_f}{t_q}. \tag{3.6}$$

We also calculate and insert in D the percentage of fee paid compared to the total input of a transaction t. This value might be useful to have a reference when the Bitcoin price increase or decrease drastically, so we are able to monitor if the fee is steady or has some changes during time. Then we calculated $t_{\%}$ as showed in Equation 3.7:

$$t_{\%} = 100 - \frac{f_{ou} \times 100}{f_{in}}. \tag{3.7}$$

The last derived data is the block propagation time, τ. The calculation for this value follows the papers from Decker and Croman [29, 27]. From 2012 to 2015 block propagation time had an increment. When it was tested from Decker and Wattenhofer in 2012 the median and 90-percentile time for Bitcoin nodes to

receive a block was 6.5 seconds and 26 seconds respectively. Considering that at the time of their measurement, the average block size was 87 KB, a full 1 MB block would have taken 5 minutes to be visible from the 90% of the nodes in the network. In 2015, last measurements from Decker and Croman show that the 10%, median, and 90% block propagation times are 0.8 seconds, 8.7 seconds, and 79 seconds respectively. Further, the average block size was at the time roughly 540 KB. Projecting to a 1 MB block size, the 90%, median, and 10% block propagation times would be 2.4 min, 15.7 sec, and 1.5 sec respectively. We consider this last measurement our reference for the block propagation time τ.

3.3 Assumptions

Our focus and studies are related to blockchain scalability, performance and see whether the constant increasing amount of transactions per day affects the way miners include transactions or the way users offer fees to miners. We assume that analyzing the last 5 years of transactions might produce interesting results about how this (not anymore) new but still cryptic system is evolving and changing its properties according to achieve better performance even after it scaled drastically in the last couple of years and still be able to compensate miners and users. We studied and analyzed previous works on the Bitcoin blockchain [27, 39, 36, 29] and we made some assumptions of how the following properties of the Bitcoin systems may vary if other attributes are changing as well. We tried to classify the main properties of the system and make assumptions on how they may vary. This attributes are Q, $\mathcal{T}$, τ, γ, t_f, $\langle C \rangle$, $\mathbb{P}_{orphan}$ and t_l and our assumptions are summarized in Table 3.2. For each property, its row is telling what happens to other attributes if the attribute considered is increased/decreased. White space means that there might be no relation or we do not have any assumption yet. For example, if we are going to increase the block space Q, then we have a relevant increase on the propagation time, τ, the chances of orphaning are way bigger, so will be the cost to mine a block, but we have a better throughput, γ and we might have less fee from transactions, t_f, since they will have less competition while trying to be included in a block.

Table 3.2: Possible Relations Between Major Attributes

	Q	$\mathcal{T}$	τ	γ	t_f	$\langle C \rangle$	$\mathbb{P}_{orphan}$	t_l
$Q \uparrow$			$\uparrow\uparrow$	$\uparrow$	$(\downarrow)$	$\uparrow$	$\uparrow\uparrow$	$(\downarrow)$
$\mathcal{T} \uparrow$	$(\uparrow)$			$\downarrow$				$\uparrow$
$\tau \uparrow$	$\uparrow\uparrow$			$\downarrow$	$(\downarrow)$	$(\uparrow)$	$\uparrow\uparrow$	$\uparrow$
$\gamma \uparrow$	$\uparrow$	$\downarrow$			$(\downarrow)$	$\uparrow$	$(\uparrow)$	$\downarrow$
$t_f \downarrow$			$(\uparrow)$			$\uparrow$		$\uparrow$
$\langle C \rangle \downarrow$	$\downarrow$				$\uparrow$			$(\downarrow)$
$\mathbb{P}_{orphan} \downarrow$	$\downarrow$					$\downarrow$		

[1] $\uparrow\uparrow/\downarrow\downarrow$: more than linear or even exponential increase/decrease
[2] $\uparrow/\downarrow$: increase/decrease.
[3] $(\uparrow)/(\downarrow)$: might increase/decrease.

4

Blockchain Observations

In this chapter we present and discuss observations of the Bitcoin blockchain captured by the analytic system outlined in Chapter 3. To evaluate and discuss our problem definition in Chapter 1.2, we focus on considerations related to performance, scalability, fees and tolls. To answer our questions and focus on the problems and possible soultions a large number of test has been evaluated, more than 100 million of transactions over 100 thousands blocks were collected, stored and analyzed. Graphs and relations between attributes which mine to verify our assumptions in Table 3.2 are represented and discussed. We can state finally that scalability brought to a centralization problem concerning miners, since nowadays there are less individual miners and more mining pools, an unlikely and ill-judged increment of $\mathcal{T}$ that might lead to an excessive high throughput γ, and an urgent problem whether increase the block size Q or not, due to the massive scale of transactions to approve; performance studies brought to consider carefully an increment of γ since it may leads either to a raise on costs $\langle C \rangle$ or a growth of $\mathbb{P}_{orphan}$, while a user-side performance study concerning the transaction latency t_l, enhanced that after 2015, t_l became strictly dependent from t_f; and finally a study on tips and tolls brought us to analyze miner's profits and users' benefits and thanks to our data collected we were able to infer the miner profit function $f^2_{\langle \Pi \rangle}$, which establish a relation between a block creation time and a miner's profit $\langle \Pi \rangle$ and to analyze how the fee paid to the miner changed during time, studying also the fee density ρ.

4.1 Scalability

Scalability may concern the number of nodes in the network but also the amount of transaction's requests per second that the system faces. The constant growth of the cryptocurrencies' popularity rises not few concerns regarding the scalability of the system. The reward for miners brought a lot of people being interested in mining, and the number of nodes for this purpose highly increased during years.This brought the emerge of *mining pools*, which consists in having a lot of miners connected together to find as fast as possible the solution of the puzzle and share the earned reward afterwards. This creates a centralization problem though, only bigger mining pools nowadays, such as *AntPool* [2], *F2Pool* [11], *BTCC Pool* [9] or *BitFury* [6], will find a solution to mine new blocks, discouraging a lot of small miners to join the network.

The other scalability problem is related to the growing amount of transactions that need to be approved every day. It is obvious that with an average t_q of 500 bytes, a fixed estimated $\mathcal{T}$ =10 minutes and a maximum block size Q fixed at 1 MB, the outside number of transactions the system can approve per second is roughly 3-4 txs/sec. Compared to systems like VISA that can easily approve 2000 transactions per second, Bitcoin or any other digital cryptocurrency using the blockchain protocol are still far in taking over digital systems such as VISA or Master Card. However we want to analyze this boundaries and understand what it possible to do according to improve the scalability.

4.1.1 Miners & Mining Pools

In our data frame D are stored all the information regarding every single transaction and which miner approved it. We analyzed and tracked down every active miner in the system from 2013 to 2017. With our 50% of the total information we evaluate a total number of 6137 miners, where 6066 are just occasional nodes not belonging to any mining pool. In Figure 4.1 the number of monthly active miners from 2013 until now is displayed. Even if the number is relative to the 50% of the blockchain, we analyzed every block mined during time and we state that after December 2014 until August 2015 there was a relevant drop, connected with the growth showed in Figure 4.2, mainly caused by the come of mining pools, KnCMiner, AntPool and BitFury in late 2014, and the massive mining power growth of SlushPool and F2Pool in mid 2015. This is the main cause for occasional miners' drop in the system, and more mining pools will join the network, more it will be disheartening for smaller nodes to start mining, since the costs will be probably higher than the revenue. This creates a sort of centralization problem, since only the bigger mining pools could take part in the whole process of transactions approval, forcing small and occasional miners to join big mining pools. From our analysis we state

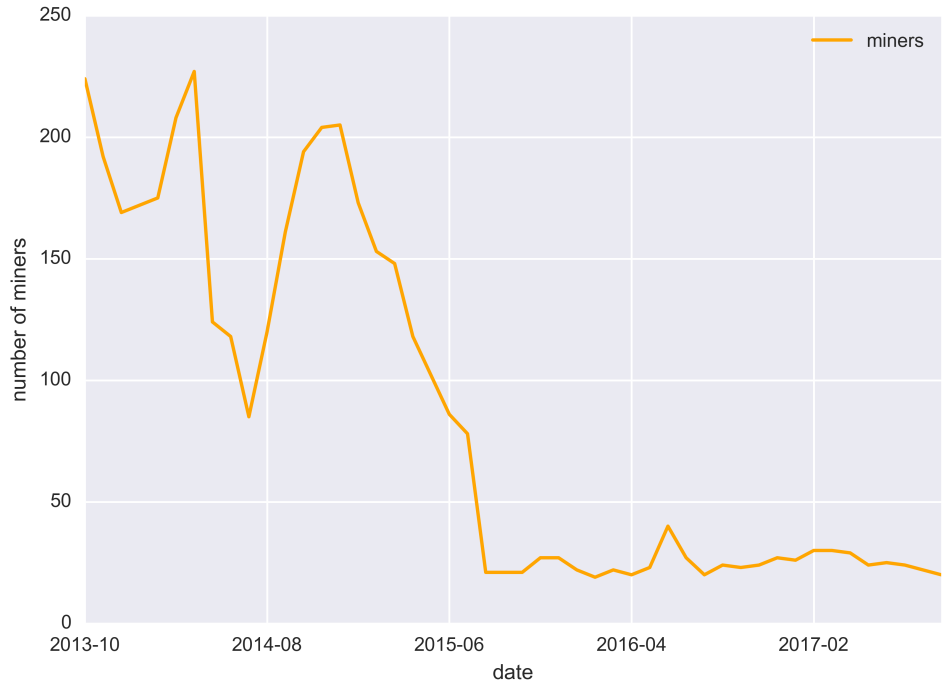


Figure 4.1: Monthly number of active miners from 2013 to 2017.

that there are different mining pools joining and leaving the Bitcoin network during these years. As Figure 4.2 shows, the most relevant example is the *GHash.IO* pool which was one of the major mining pool until mid 2015, then its mining power started to decrease slowly until its stop in October 2016. After some research, we refrain to an article from `coindesk.com` [4] which says that Bitcoin miners ditch GHash.IO pool over fears of 51% attack. Indeed in 2014, according to `blockchain.info`, GHash.IO accounted for more than 42% of Bitcoin mining power. Nowadays as the Table 4.1 shows, AntPool has 19.53% of the mining power, and there is no centralization risk. We analyzed then the major mining pools from 2013 until 2017, we divided dates analyzed in two eras, 2013-2015 and 2016-2017, selected the first 10 miners regarding their mining power and then represented in Tables 4.2-4.1. Table 4.2 shows that a big percentage of the mining power comes from occasional miners and for the whole period between 2013 and 2015 GHash.IO has only a 13.3% of mining power. We can see indeed in Figure 4.2 that this mining pool had only a short time peek in between 2013-2015, but anyway in an overall scheme it managed to collect the 13.3% of the whole share. Table 4.1 shows that AntPool, F2Pool and BTCC Pool are dominating the mining power in Bitcoin network from 2016 until now with values respectively of 19.53%, 16.43% and 10.5% of share. If compared with Table 4.2, AntPool is the one which is growing faster and mining pools such as GHash.IO and BTC Guild stopped their activity. We

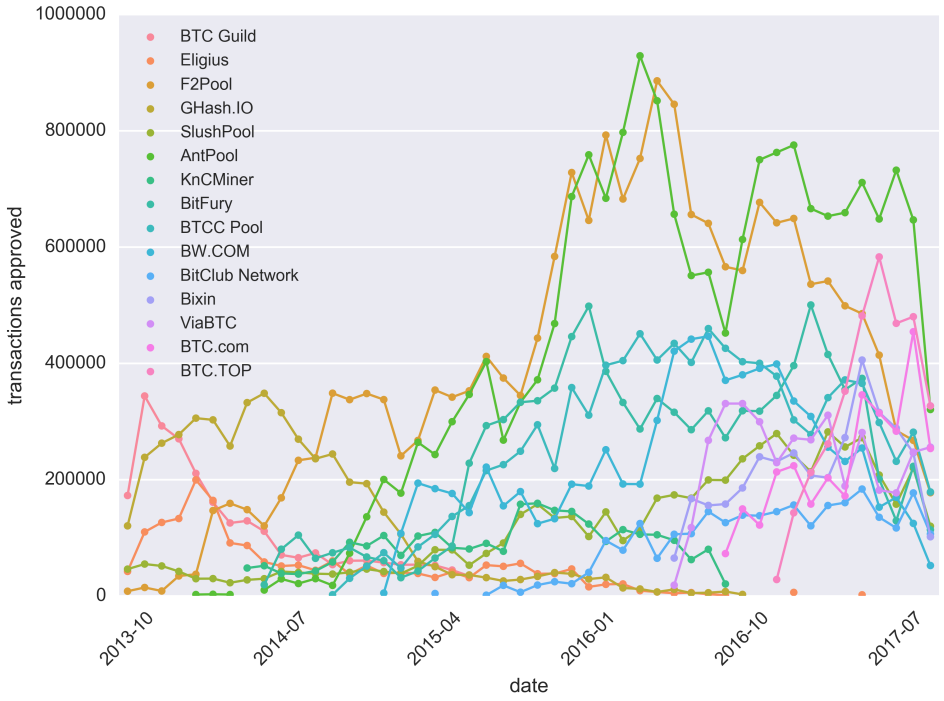


Figure 4.2: Top 15 mining pools from 2013 until 2017. In total, they approve the #% of the total Bitcoin's transactions ever occurred.

state finally that because of these results, no miner seems to get even close to the GHash.IO share in 2014 and it seems like we will not have a centralization problem in the very near future. However the mining power during the years changed from a distributed architecture to a decentralized one. We can see in Figure 4.3 that the occasional miners almost disappeared in the last two years, leaving their space to mining pools, while they were contributing for almost the 100% at the end of 2013. This brought a downside for individual users which want to buy their own hardware and start mining for themselves. Since Bitcoin always claimed its distribution as a system's strength, this scalability issue might result a centralization problem in the long run, excluding small miners from the system and letting only the biggest mining pools to have informations about the whole blockchain. Bitcoin is useful only if is decentralized because centralization requires trust and Bitcoins value proposition is trustlessness.

4.1.2 Unconfirmed Transactions

Another relevant concern about scalability in blockchains is the number of transaction's requests the system can accept. As we state in Chapter 4.2.1,

Table 4.1: Mining power distribution in a scenario from 2016 until 2017, considering only the 10 major mining pools.

Mining Pool	Mining Power (%)	Blocks Mined
AntPool	19.53	9,078
F2Pool	16.43	7,635
BTCC Pool	10.5	4,874
BitFury	8.71	4,050
BW.COM	8	3,717
SlushPool	5.26	2,447
ViaBTC	4.8	2,222
Bixin	4.12	1,947
BTC.TOP	4	1,857
BTC.com	3.55	1,648
Total Analyzed	84.9	39,475
Other Miners	15.1	6,996

Table 4.2: Mining power distribution in a scenario from 2013 until 2015, considering only the 10 major mining pools.

Mining Pool	Mining Power (%)	Blocks Mined
F2Pool	13.53	10,548
GHash.IO	13.3	10,370
BTC Guild	7.37	5,749
AntPool	6.82	5,323
Eligius	4.99	3,889
BitFury	4.73	3,693
BTCC Pool	3.9	3,042
SlushPool	3.37	2,627
KNCMiner	3.12	2,432
BW.COM	2.67	2,064
Total Analyzed	63.8	49,737
Other Miners	36.2	28,225

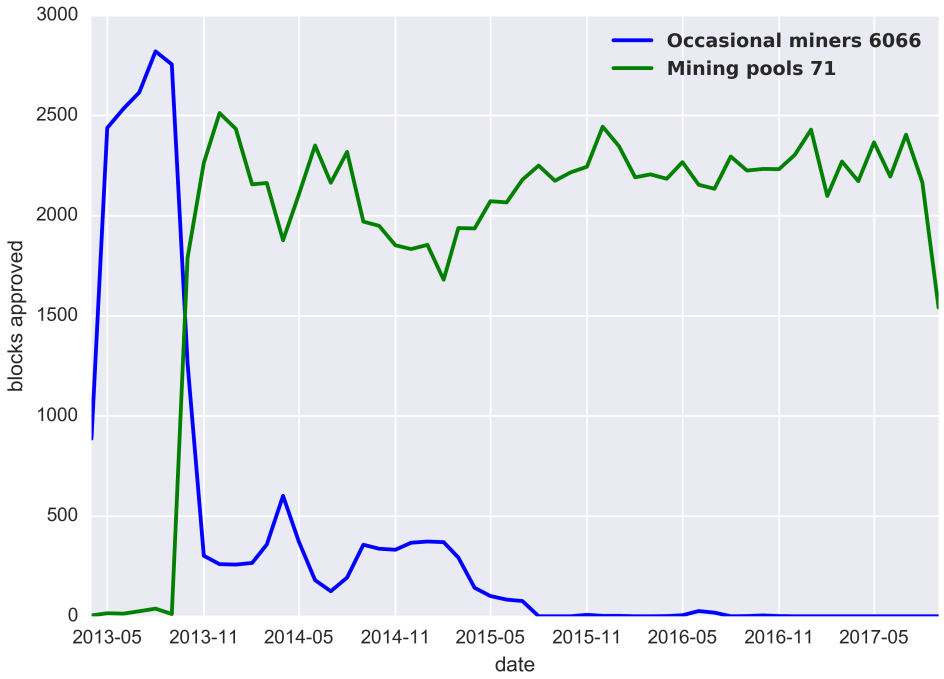


Figure 4.3: Number of transactions approved from occasional miners and mining pools from 2013 until 2017.

throughput (γ) may vary according to Q, $\mathcal{T}$ and t_q. The difficulty of the Bitcoin proof-of-work is adjusted according to the hashing power of the entire Bitcoin network so that the average time to mine a block, $\mathcal{T}$, will always be around a certain value. In our analysis we prove that $\mathcal{T}$ stays stable for the whole time of testing and this value is set at 10 minutes. In this interval of time, since the block size Q is set to a limit of 1 MB, the system can approximately approve, if we consider an average t_q of 500 bytes, 3.3 transactions per second. This value is calculated considering the maximum block size possible of 1 MB. If we consider an interval of time in between 2013 and 2017, the number of daily transactions that need to be approved scaled from 50 thousands in 2013 to 300 thousands in 2017 and all blocks are filled over their maximum capacity since April 2016, as Figure 4.4 shows. Because of this scaling limitation, transactions might be willing to pay more fee in order to get accepted in miner's mempools. To improve the number of transactions that can be accepted per second by the system, we consider an eventual change of $\mathcal{T}$ and Q, since we analyzed that the average t_q is constant and equal to 500 bytes for the whole period of testing. We have to consider though, that even if is possible to improve the number of transactions that could be accepted from the system every second, it will be anyway difficult to get better results than circuits such as VISA or Master Card and the scalability from this prospective is still far to be reached.

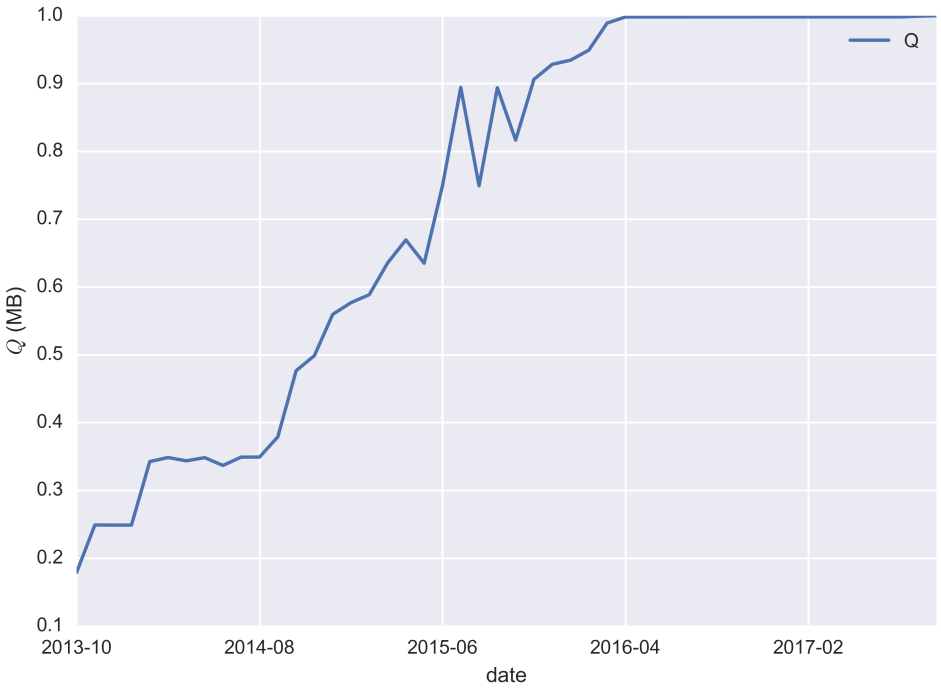


Figure 4.4: Average block size for every month, from 2013 until 2017.

Block Creation Time $\mathcal{T}$

In our assumptions showed in Table 3.2 we argue that an increment of $\mathcal{T}$ may lead to a γ drop and an increment of t_l, for that reason we think might be a better choice to adjust the difficulty according to decrease the block creation time and not to increase it. Furthermore, according to Rizun [39], in Equation 2.2 the cost for mining a block is directly proportioned to the block creation time, which means that if the $\mathcal{T}$ is decreased, the cost for mining a block is less. Thanks to our data frame D we managed to prove that there is an inverse proportionality between γ and $\mathcal{T}$. Only in blocks with a creation time lower than 10 minutes the throughput reaches peaks of 10-14 txs/sec, while in cases from 10 to 20 minutes of creation time, the throughput is in between 2-4 txs/sec and in blocks where the $\mathcal{T}$ is higher than 20 minutes, throughput never reaches 2 txs/sec.

Our assumptions about transaction latency were partially right. it is true that t_l tends to grow with the growing of $\mathcal{T}$ but this is verified only if $\mathcal{T} \geq 10$ minutes. If the block is mined before that time then we do not have any relation between transaction latency and block creation time. We suspected that this uncertain relation regarding blocks mined before 10 minutes might be somehow connected with who mined that particular block. While mining pools may have

different criterion (higher transaction fee) while appending transactions to be mined, occasional miners might have none, or less. This is the reason why the t_l might be so random if t is mined by some occasional miner, since transactions were not selected according to any criterion, so a transaction t who did not offer any fee and is waiting to be approved since longer time, might be included by some occasional miner if it luckily wins the proof-of-work before the 10 minutes time. After some tests, we show the results in Figure 4.5. As suspected, for $0 \leq \mathcal{T} \leq 7$, the occasional miners take the 21.3% of mining share, while in the other portions they never reach the 18% of the total blocks mined.

In conclusion, an increment of $\mathcal{T}$ is ill-judged, while should be considered a decrement of it, acknowledging according to Equation 2.4, that an increased bandwidth for a better inner node communication is needed, otherwise we might have an exponential growth on $\mathbb{P}_{orphan}$. Furthermore, in Chapter 4.3.1, we state that with an higher $\mathcal{T}$ the profit $\langle \Pi \rangle$ of a miner is significantly lower.

Block Size Q

If the creation time $\mathcal{T}$ tends to have a fixed value then Q plays a big role in throughput limitations. Discussions whether increase or not the block size limit are still ongoing and this issue has been brought up in several papers [39, 31]. Increasing the block size limit leads to a bigger amount of transactions that could be accepted every block creation time, so we may think that this could be the right solution to adopt according to increase system's performance. Anyway there is a big controversy on this topic, and even the biggest entities in Bitcoin system do not agree to a solution. Some such as CoinBase, Blockchain.info or BTC Guild, think is good to increase the block size, some others like F2Pool or Ethereum are neutral and other entities, for example GreenAddress, Bitrated or Bitcoinpaygate, think that this is just pushing the problem a little further and is not a good solution. In our assumptions we also state that transaction fee t_f may decrease with an increment of the block size limit since transactions do not have to compete anymore for space in the block. So a small block size will require higher fees for fast confirmation, which will bring the following pros (+) and cons (−) [8]:

+ It will no longer be cheap to spam transactions such as Satoshi Dice bets [16].

+ Fees will not be zero. This is eventually a necessity in order to incentivize miners and secure the mining ecosystem.

− Bitcoin may look unattractive to new users with high fees.

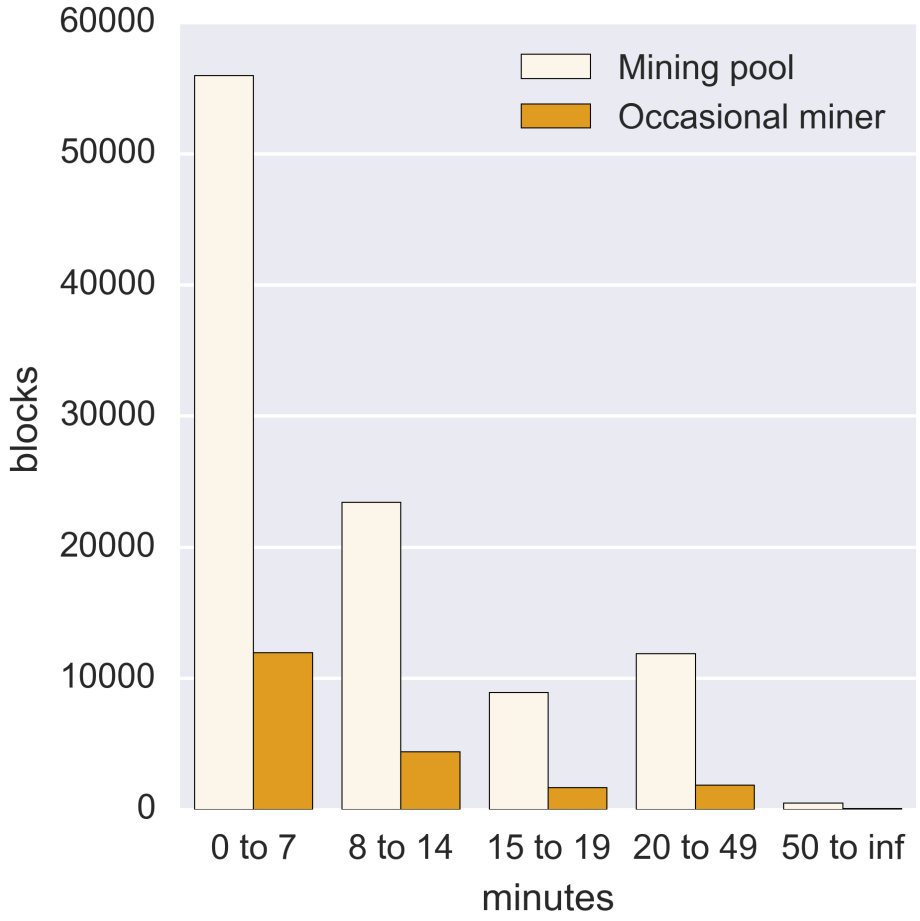


Figure 4.5: Relation between the creation time $\mathcal{T}$ and number of block mined according to miner's category.

– Users pay higher fees.

We tested that not only the t_f is related with the increment of the block size limit, but thanks to our evaluations showed in Figures 4.11-4.12, we can state that from when the blockchain started to become saturated in mid 2016 (Figure 4.4), t_f but also the fee density ρ had a big increment, sign that the blockchain needs an urgent change according to avoid scalability problems. Increasing the block size limit is one of those possible solutions, but we have to consider that it might be good for users but also bad for miners. The cost for mining bigger blocks is higher, so receiving no more fees from users would be costly effective and may leads to a centralization problem, since larger blocks make full nodes more expensive to operate, having then less hashers running full nodes. If we consider now the statement that block size limit should be increased, these are

the arguments against it:

- Orphan rate amplification.
- Damage to centralization.
- "Congestion" concerns can be solved with mempool improvements including transaction eviction.
- No amount of max block size would support all the world's future transactions on the main blockchain (various types of off-chain transactions are the only long-term solution).

In conclusion, even if we think that an immediate increment of the block size is not the final solution to the scalability problem, we believe that an immediate and temporary solution should be considered to avoid network congestion, too high t_l and drastic increment of t_f. This temporary solution might be the block size limit increment, especially if big mining pool such as F2Pool claim that they could support a maximum block size of 20 MB [8].

4.2 Performance

Together with scalability, performance is one of our main study subject on Bitcoin blockchain and we can say that it is highly related with scalability. Indeed in Chapter 4.1.2 we already talked about throughput γ and how it is related with the block size Q and block creation time $\mathcal{T}$. We want to focus now on Bitcoin network performance from both, user side (t_l) and system side (γ). Moreover, our target by studying transaction latency t_l is to verify our assumptions whether there is any relation between t_f and t_l during the whole period analyzed. These tests are strictly related with the fees and tolls problems (Chapter 4.3) and are important to make users aware whether is possible to pay for bandwidth in Bitcoin system and if yes, how to optimize their fee according to get faster t_l for a relatively low price t_f.

4.2.1 Throughput γ

As we state in Chapter 4.1.2, γ is highly dependent from Q and t_q, but also related to $\mathcal{T}$. Block size limit can set the number of transactions the network can confirm per block, so the throughput of the whole Bitcoin network. In Figure 4.6 we show how throughput changed during years. We want to clarify that in 2013 the system wasn't less performing, there were just less transactions

to process, roughly 50 thousands per day and only when the blocks started to get saturated in mid 2016, then the system began to use its maximum throughput capacity of ~3-4 txs/sec. However, if compared with centralized trust-oriented system, throughput is extremely low. We said that circuit like VISA or Master Card could approve 2000 transactions per second which is still unthinkable for blockchain distributed systems. Our assumptions on throughput (Table 3.2) turned out to be right about the block size and the block creation time, indeed according to have an higher throughput in the system, Q needs to be bigger, $\mathcal{T}$ lower, or both, with all the advantages and disadvantages that this could bring enhanced in Chapter 4.1.2. However, an increment on γ will bring a raise on the cost $\langle C \rangle$ to mine a new block if Q is changed, and a growth of $\mathbb{P}_{orphan}$ with also repercussions on $\langle C \rangle$ if $\mathcal{T}$ is lowered.

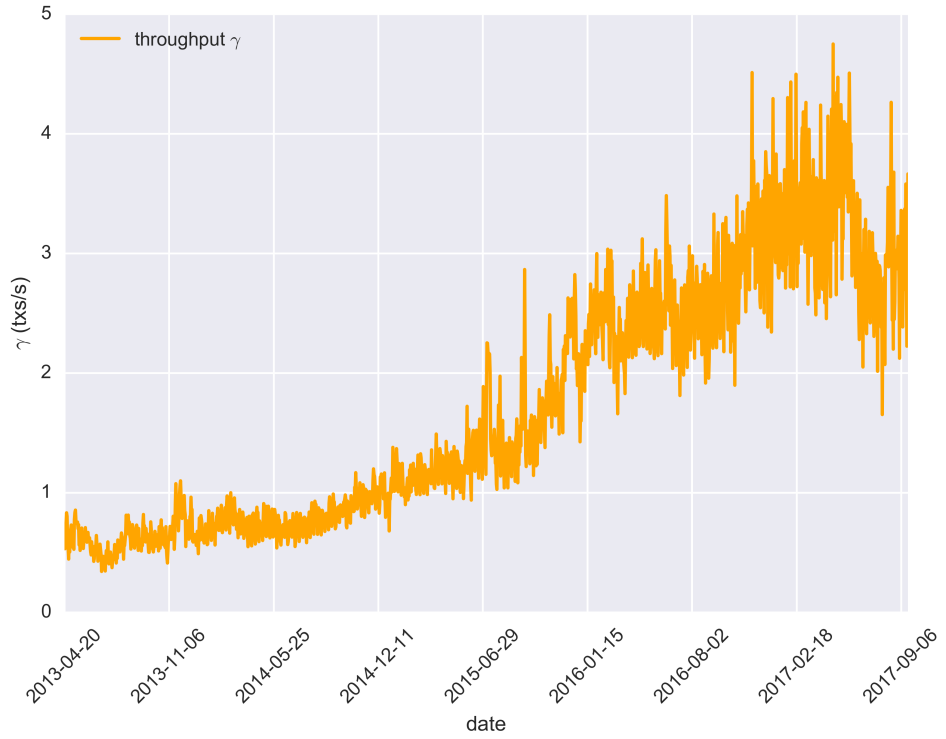


Figure 4.6: Throughput (γ) of the Bitcoin blockchain calculated in a period from 2013 until 2017.

4.2.2 Transaction Latency t_l

Another important aspect of performance regards the bandwidth Bitocin could offer to users. We considered then the latency t_l of every transaction analyzed, which is the time necessary for t to be approved by the network. We want to

Table 4.3: Table representing the results in Figure 4.8, showing how t_l might vary for each category of fee.

Fee category	t_f **value (Ƀ)**	t_l **variation (h)**
0	$t_f = 0$	1 to 33
<0.0002	$0 \leq t_f < 0.0002$	0+ to 5+
<0.0004	$0.0002 \leq t_f < 0.0004$	0+ to 2.5
<0.0006	$0.0004 \leq t_f < 0.0006$	0+ to 1.5
<0.0008	$0.0006 \leq t_f < 0.0008$	0+ to 1
<0.001	$0.0008 \leq t_f < 0.001$	0+ to 1
<0.01	$0.001 \leq t_f < 0.01$	0+ to 1
<0.1	$0.01 \leq t_f < 0.1$	0+ to 1.5
>0.1	$t_f \geq 0.1$	0+ to 2

find causes about how and why t_l and t_f changed during this years pointing out, if any, the relation between these two data, so the users are aware while paying a certain t_f, in how much time approximately their transactions will be approved and confirmed by the network. We observed that from 2013 to 2017, while the average transaction latency did not vary much and was stable at ~15 minutes, the fee paid from users to the miners has undergone a strong raise, from an average of 0.0001 Ƀ to 0.0004, up to 0.0012 Ƀ in 2017, where the Bitcoin price is also 2000 times higher than 2013. In our assumptions (Table 3.2) we show how t_l could be affected by other changes in the system. While assumptions on τ and γ are immediate, so the transaction latency will increase if the propagation time grows and will decrease if the throughput gets higher, other might not be that direct and they require a deeper study. We analyzed then the relation between block size Q and transaction latency t_l. We categorized numerical data from Q according to the different block sizes during the years showed in Figure 4.4 and plotted them yearly to show any possible relation with t_l. As we suspected, during the years, whenever there was an increment on the block size, transaction latency had a drop. In 2016 when we had the final increment to 1 MB, the latency started to become incredibly low, while in 2017 when the blocks started to get saturated we have a latency up to 9 times higher, plus, smaller blocks in 2017 verify a much higher transaction latency since they include less transactions than the number required, as Figure 4.7 shows.

Not much intuitive and more difficult to study was also the relation between t_l fee and t_f. According to analysis made on t_f we opportunely categorized the fee paid from users to miners and then we split the data frame per year. Table 4.3 enhance this categorization and Figure 4.8 shows the plot generated while analyzing more than 120 million of transactions from 2013 to 2017. As Table 4.3 and Figure 4.8 enhance, the fee paid from users has a bigger impact year by

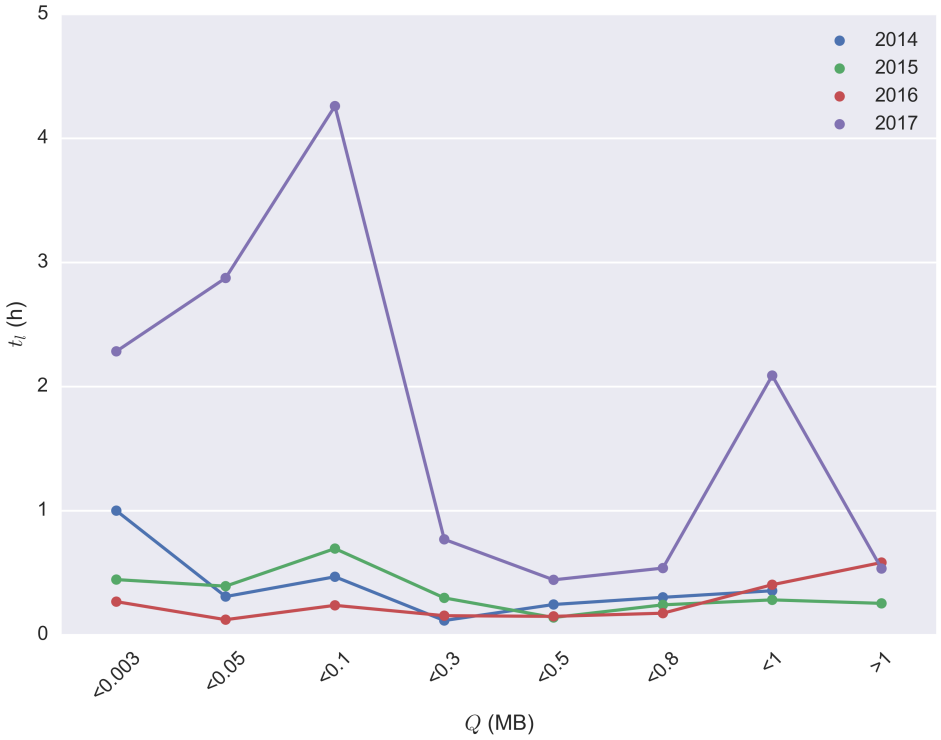


Figure 4.7: Relation between t_l and Q from 2014 to 2017.

year on the available bandwidth the system offers. Our assumptions about this relation were right, and especially after the blocks started to get saturated in mid 2016 we had a big change in the way miners include transactions. If in 2013-2014 miners and mining pools tended to include in blocks a compromise between fee and waiting time, in 2017 miners tend to completely ignore 0-fee transactions. This could be good to avoid transactions spam and it is good for miners which are gaining more out of mining, since the block reward is halved every 210, 000 blocks. On the other hand, impatient users need to pay an higher fee to see their transaction approved faster. Furthermore, we can state that in 2017, 0-fee transactions have a latency of ~33 hours, 4 times bigger than 2016, 8 times than 2015 and more than 16 times bigger than 2013-2014, which means that the problem of block saturation has taken into consideration by excluding 0-fee transactions. These leads us to the fees and tolls problem in the system and to explain how we studied them and what we concluded out of it.

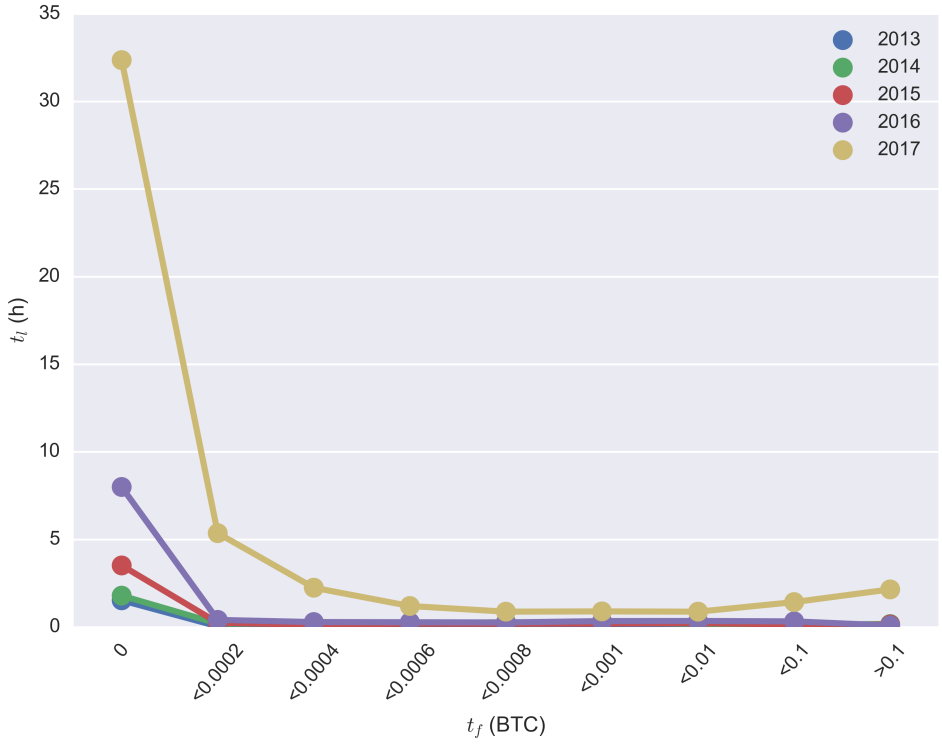


Figure 4.8: Relation between t_l and t_f during time.

4.3 Fees and Tolls

This Section aims to show the changes of tips and tolls in Bitcoin system from 2013 to 2017. In the paper from Möser and Böhme [36] they claim that the hard size limit still does not affect the level of the fee paid. We want to enhance instead, that nowadays the saturation problem it significantly drive the fee paid from users in order to get more bandwidth or to be accepted at all. Furthermore, at the time of analysis (2015), Bitcoin claimed that the fees were lower than 0.1% while nowadays we analyzed (Figure 4.15) to have an average percentage of 0.25%, which is more than the double. Plus we want to consider the mining cost for miners, from what it is driven and how it will change in the past years. When we talk about fees and tolls we might refer both to miner's profit and users' benefits.

4.3.1 Miners' Profit

The study of miner's profit is complex and a lot of data analysis and manipulation is required. Knowledge regarding electricity consumption, mining

hardware and cost of hashing is needed and a deep study and data analysis on how the hashing Bitcoin rate and Bitcoin price in USD changed during the years has to be taken into consideration. Making assumptions on mining costs/profits might be difficult due to the information withheld from mining pools. We think though, that this is an important concern and it has to be considered and studied since the block reward is even halved every 210, 000 blocks and miners should find another way to profit out of mining. Solutions for miners might be to additionally charge users with higher fees or to optimize their mining profit $\langle \Pi \rangle$ by analyzing the costs $\langle C \rangle$ and the revenue $\langle V \rangle$ during time. We refer to Equations 2.2, 2.3, 2.5 in order to calculate $\langle C \rangle$, $\langle V \rangle$ and $\langle \Pi \rangle$. For the reckoning we had to adapt our data and formulas to the Bitcoin price and Bitcoin total hash rate H at the time when data were generated. To do so, we extracted our time information from D, collected JSON data regarding H and Bitcoin price from `blockchain.info` and finally fitted them in order to get $\langle \Pi \rangle$, $\langle C \rangle$ and $\langle V \rangle$. Equation 2.5 gives us information in order to calculate $\langle C \rangle$ and $\langle V \rangle$, and we use a propagation time of $\tau = 15.7$ seconds, considering in that way a scenario with an average block size of 1 MB and a 50% propagation; in addition, we contemplate an electricity cost of 0.04 \$/kWh (Chinese price, since AntPool is a Chinese mining pool) and examining the most powerful mining hardware on the market for Bitcoin, AntMiner S9 [1], having an hashing power of $h = 14,000,000$ MH/s and a consumption of 1375 W, with an overall cost per hash of $\eta = 1.091 \times 10^{-18}$ \$/H. A similar analysis on electricity consumption and cost of mining was made by Croman [27] in 2016. For our calculations we only considered transactions from late 2016 and 2017, when Bitcoin's price was already over 1000 \$, before that, data might contain too many outliers due to huge differences in Bitcoin's hashing rate and Bitcoin price, and they could drive the analysis to an incorrect status. Figure 4.9 shows how the miner's profit is related with the block creation time and how after 15-20 minutes the miner starts to decrease its profit. This is caused by the constant growth of the cost necessary to mine a block, $\langle C \rangle$, which is directly proportional to the cost per hash η and the block creation time $\mathcal{T}$. Is important to enhance that Figure 4.9 shows the profit $\langle \Pi \rangle$ for a single miner using AntMiner S9 and that in a real case scenario we might have hundreds of computers working together, so the possibilities to find a new block are much higher and the revenue $\langle V \rangle$ is accordingly bigger, having then much greater profit that needs to be shared among all the miners.

Furthermore, as we will see in Chapter 4.3.2, miners most likely include transactions with an higher fee density (ρ), so if users are not willing to pay more fee, once all the best ρ-wise transactions are already included in the new upcoming block but Q is not reached yet, miners have to include lower fee density transactions, having in that way a loss in their profit. So if we look at it from a miner's prospective, the increment of Q might not be the best choice in the long run, unless the fees are standardized in a way that every miner has a guaranteed

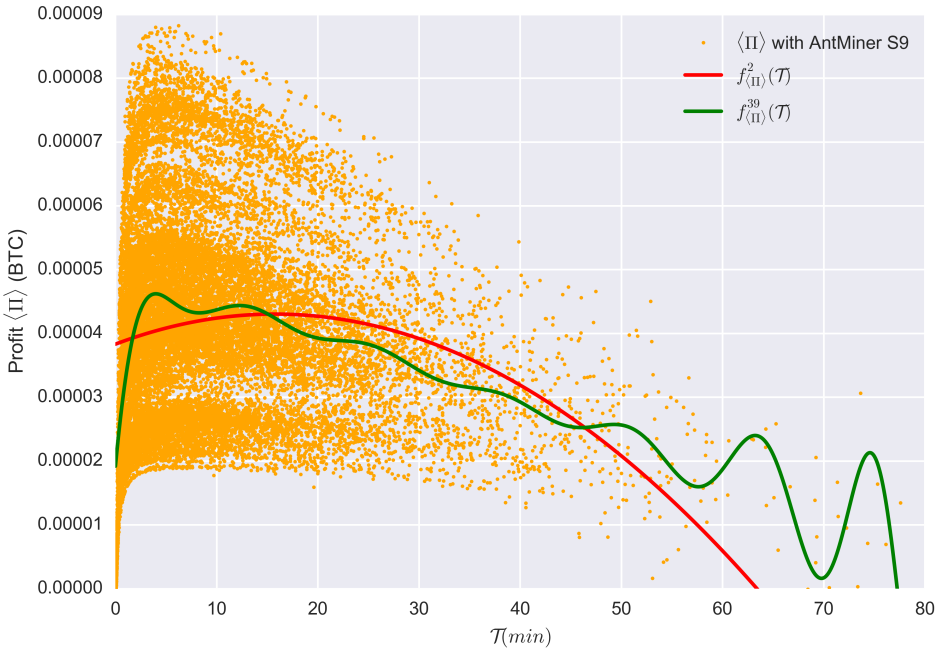


Figure 4.9: Profit ⟨Π⟩ mining with AntMiner S9 from 2015 until 2017 according to block creation time $\mathcal{T}$.

profit. If back in 2009-2013 the block revenue was poorly influenced by user's fees and it relied more on the reward R, we can see instead in Figure 4.10 that R is decreasing while the fees are getting higher and we expect M to overcome R by 2020 when the reward is halved again at 6.25 ʙ. As we have seen, the block profit ⟨Π⟩ is also highly influenced by the users' fees t_f, but is obvious that more a miner will request for fee, more users are discouraged to pay with Bitcoin. Furthermore we noticed a strong link between the halving of R in mid 2016 and the increment of t_f. Because of that we analyzed transaction fees offered by users to see how they are influenced in paying more or less ʙ to the miners during they years.

Miner Profit Function

To have a mathematical idea of what is happening between miner's profit and creation time we define $f^{2}_{\langle\Pi\rangle}$ as the *Miner Profit Function* in Equation 4.1 which is inferred after our analysis on almost 25 thousand blocks, considering a single miner using AntMiner S9, and having $\mathcal{T}$ in minutes as input.

$$f^{2}_{\langle\Pi\rangle}(\mathcal{T}) = -\frac{1.896}{10^{8}}x^{2} + \frac{5.977}{10^{7}}x + \frac{3.831}{10^{5}} \quad \Big\{ \text{with } 0 \leq \mathcal{T} < 80 \tag{4.1}$$

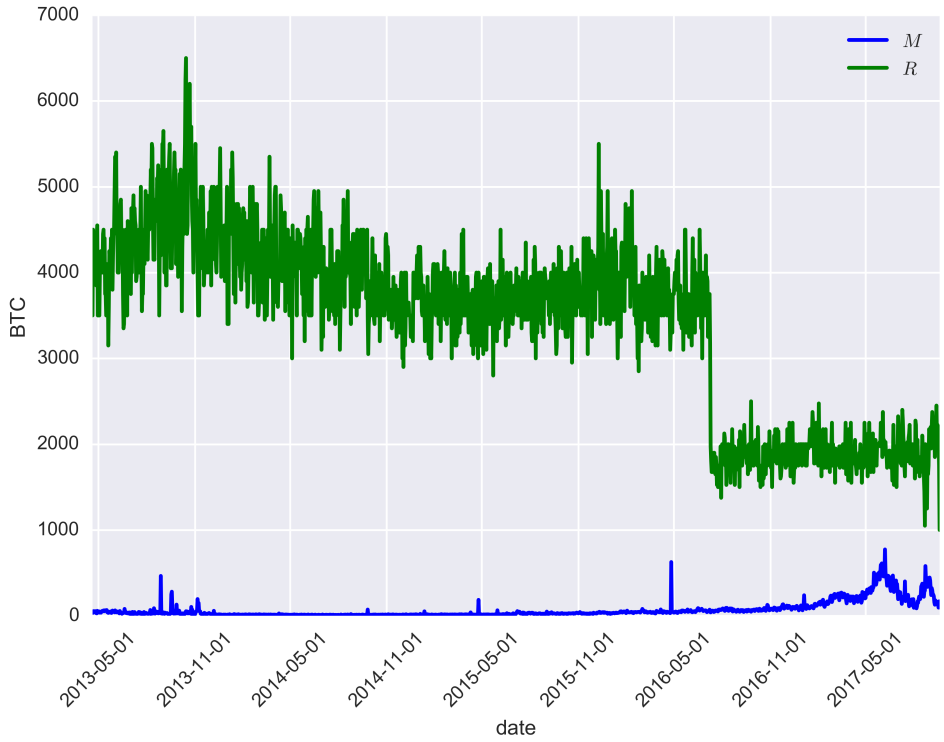


Figure 4.10: Miners revenue $\langle V \rangle$ divided in block reward R and sum of all t_fs in a block, M, analyzed in between 2013-2015. Data are represented daily, and the R and M values are sums of every specific day.

To optimize $\langle \Pi \rangle$ according $f^2_{\langle \Pi \rangle}$ the difficulty should never be increased in a way to have a creation time $\mathcal{T} >>$ 20 minutes otherwise miners have a loss in their profit, and for miner's sake, should be even better to slightly decrease $\mathcal{T}$ according to avoid useless computation that leads to an higher cost and waste of electricity. We also inferred, to have a more accurate trend, $f^{39}_{\langle \Pi \rangle}$, which is also showed in Figure 4.9 but not mathematically represented, being a 39 degrees polynomial. We could use this trend though to assume that the real profit occurs in between 3 and 8 minutes, while having less or more creation time $\langle \Pi \rangle$ will not be optimized.

4.3.2 Users' Benefits

In order to make assumptions and conclusions on how users should use their fee to optimize their bandwidth in the system, we first analyze how the t_f changed during time. Figure 4.11 shows the numerical attribute t_f divided into categories and represented in percentage for each category. We can notice that after the first half of 2016 fees in between 0 Ƀ and 0.0002 Ƀ almost disappeared from

the system, and considering that the Bitcoin price raised from less than 1000 $ to more than 5000 $ in between mid 2016 and second half of 2017 this results to be as a huge increment on the fees paid to miners, especially if we consider that, after plotting the Bitcoin price and the fee paid in USD we see substantial co-movement which indicates that BTC is the dominant unit of account when deciding about the fee offers and not USD, and is possible to see this massive increment of M in Figure 4.10, indeed the total t_f paid from miners almost reaches 1000 Ƀ in 2017. We can also see that we had higher fees early in 2013, but at the time the USD price for Bitcoin was less than 120 $ so respectively the fees were much lower. We believe that events happening on the Bitcoin blockchain might influence fee, tolls and the way blocks choose transactions, and to confirm our assumptions we see that the biggest increment in t_f was in late 2016, which coincides with the halving of R and the blocks saturation. To

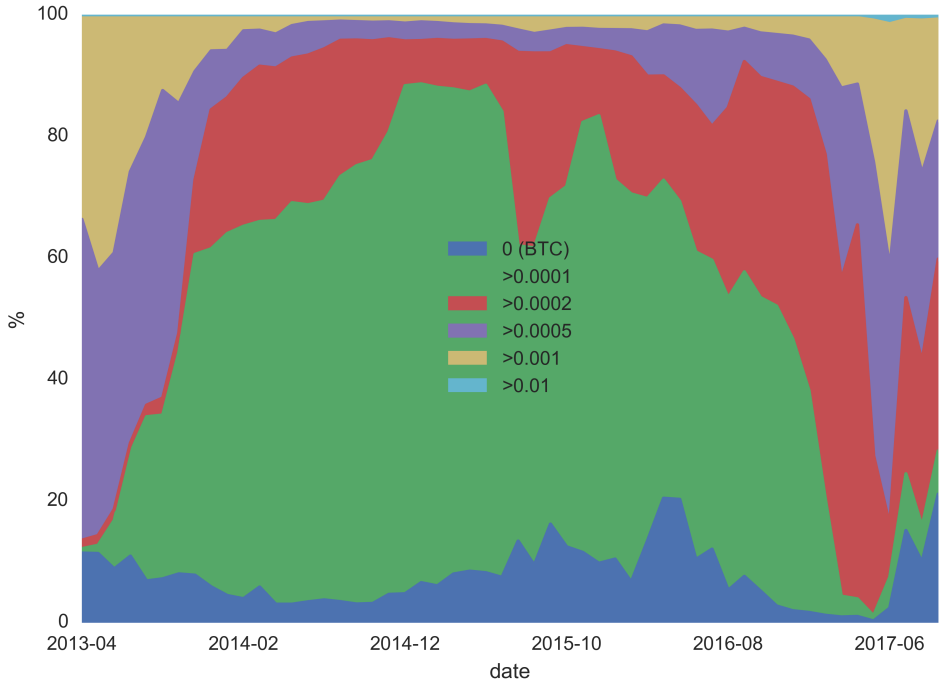


Figure 4.11: Transaction fee (t_f) distribution during the years from 2013 until 2017.

see how miners change their behavior according to events happening on the Bitcoin system we study the fee density ρ, defined in Equation 3.6 and showed in Figure 4.12. Fee density and fee are directly connected, since the transaction size t_q has an average of 500 bytes, but at the end of 2017 even if we have some fees with < 0.0001 Ƀ, we almost never have transactions with a $\rho = 0$, which means that recently, miners might have changed from having constraints on fees to having constraints on fee density. This drastic change in t_f and ρ lead us to think that there is another factor which is influenced by it, and in our

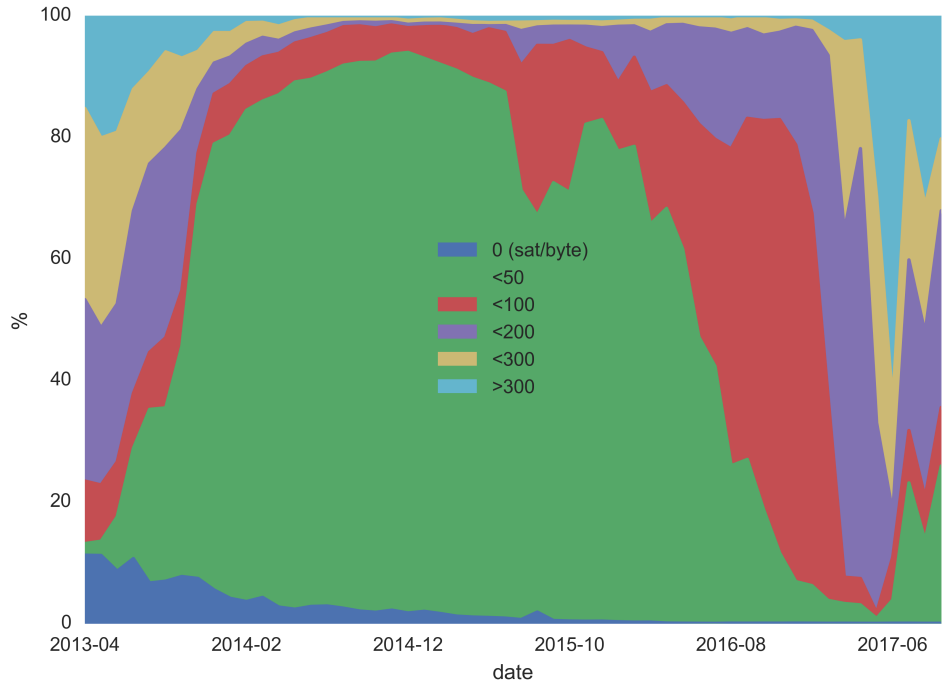


Figure 4.12: Fee density (ρ) distribution during the years from 2013 until 2017.

assumptions we thought it was the transaction latency t_l. This is the reason why we analyzed t_l in function of t_f from 2013 to 2017 (Figure 4.8) for an overall trend and then t_l in function of t_f and ρ for transactions occurred in 2017 to generate our prediction models. As also Figure 4.8 shows, is not good to invest in so much high fees, since after a certain threshold, the latency would be even higher. With our models we want to define these thresholds and give users some ideas on how they might invest wisely their money, according to optimize their bandwidth. With data regarding transactions evaluated in 2017 we generate two models, *Fee Density - Latency Function*, $f_{t_l}(\rho)$, represented in Equation 4.3, and *Latency Function*, $f_{t_l}(t_f)$ showed in Equation 4.2. In Figures 4.13, 4.14, due to our interpolation function's boundaries, we show two different degrees of polynomial interpolation, $f^2_{t_l}$ and $f^{39}_{t_l}$. The purpose of $f^2_{t_l}$ is to have a general equation to refer on while talking about fees and latency, while the $f^{39}_{t_l}$ gives us more strict thresholds about the trend our data tend to follow, for example in Figure 4.8 we see that in 2017 if you pay more than 0.001 Ƀ, you most likely get an increment in your transaction's approval time, and in Figure 4.13, $f^{39}_{t_l}$ shows this threshold while $f^2_{t_l}$ does not.

$$f^2_{t_l}(t_f) = 6248x^2 - 555.8x + 1.42 \tag{4.2}$$

$$f^2_{t_l}(\rho) = \frac{5.416}{10^8}x^2 - \frac{2.215}{10^3}x + 1.598 \tag{4.3}$$

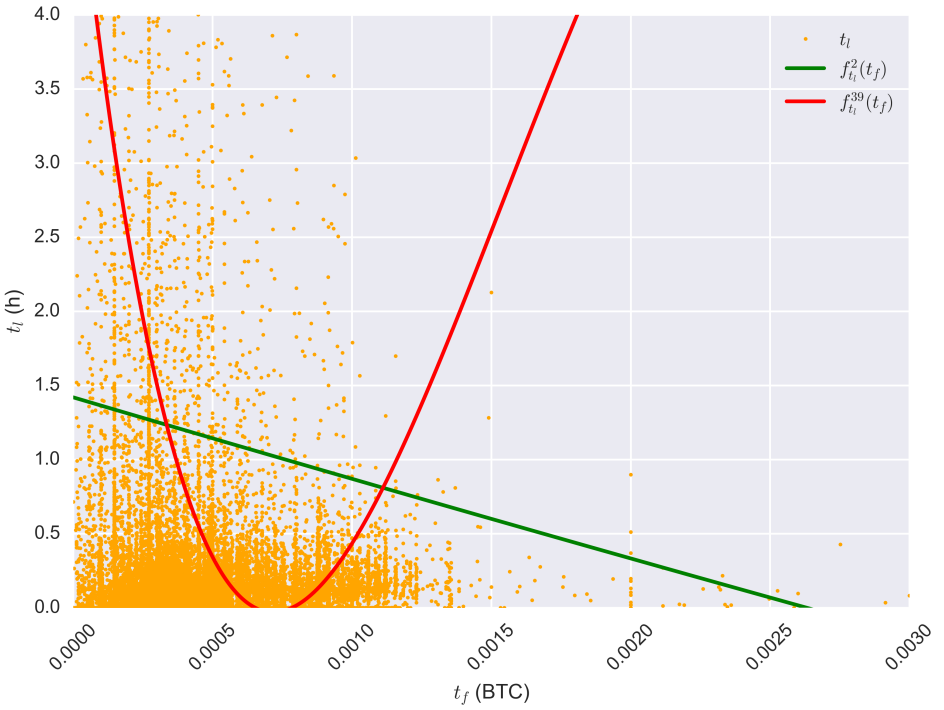


Figure 4.13: Interpolation with a 2 and 39 degrees polynomial of the relation between t_f and t_l, for transactions analyzed in 2017.

We can define in that way the thresholds in which our functions might work and they might be the following:

$$0 \leq t_f \leq 0.0011,$$

and

$$0 \leq \rho \leq 460.$$

If the $f^2_{t_l}$ function is used to make predictions and your t_f and ρ are outside this thresholds, then predictions might not be accurate and plus, you might be losing more money while getting even an higher latency. In conclusion, with data collected for the whole 2017, we can state that a user can definitely pay for bandwidth in Bitcoin's applications but it shouldn't be willing to offer more than 0.0011 Ƀ or it might waste those money for no reasons. In addition, if nowadays miners tend to consider ρ rather than t_f, is good for users to know that an optimal ρ according to get higher latency would be in between 200 and 300 sat/byte.

Now if we consider our t_f as a percentage of the total output t_{ou}, where $t_f\% = (t_f \times 100)/t_{ou}$, also the overall percentage $t_f\%$ had a consistent increment during the years and from all mining pools, like Figure 4.15 shows, going from

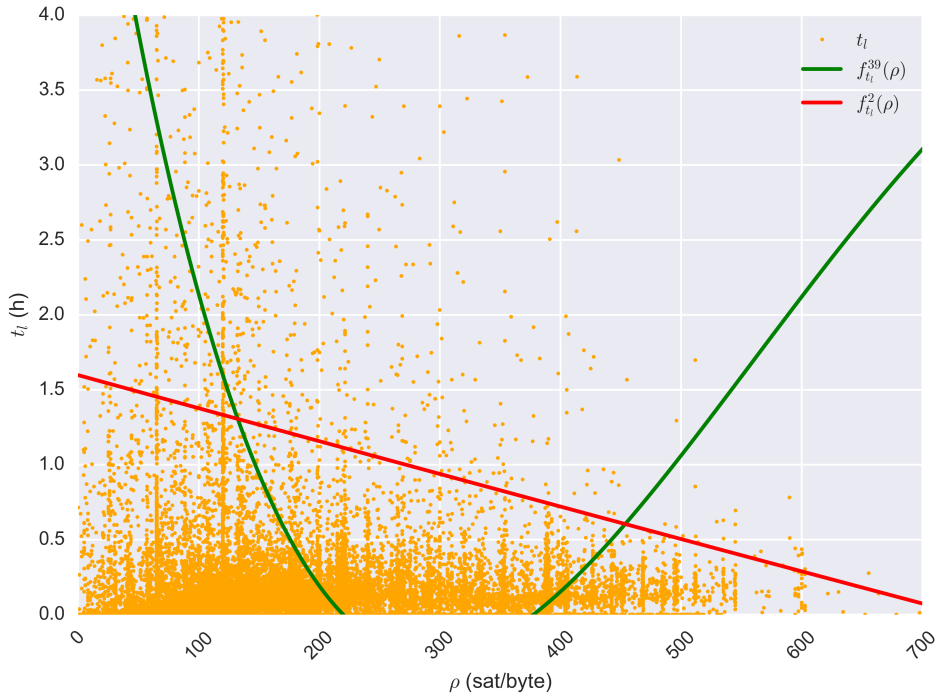


Figure 4.14: Interpolation with a 2 and 39 degrees polynomial of the relation between ρ and t_l, for transactions analyzed in 2017.

less than 0.05% between 2013 and 2015, to more than 0.2%, reaching also 0.3% in 2017. Considering now that the Bitcoin price raised from 1000$ to more than 7000$ in the last two years, this increment in the t_f% results as a huge addition to the costs for users, and Bitcoin might not be as cheap as it claims to be.

4.4 Is Bitcoin a Green-Wise Choice?

After calculating the miner's consumption related to the revenue we are concerned to know which is the impact of Bitcoin network on the environment. To know the electricity cost of mining in the Bitcoin network is necessary to know exactly how many miners are running and how much is their consumption. Unfortunately no one really knows how many miners are active and running in the network and this number changes every day, plus we don't have information about which miner's hardware is used but we could esteem they are in the order of hundred thousands, they group in mining pools and we could consider they are using the most powerful mining hardware on market at the time of analysis which is Antminer S9. The problem of consumption is that, even if

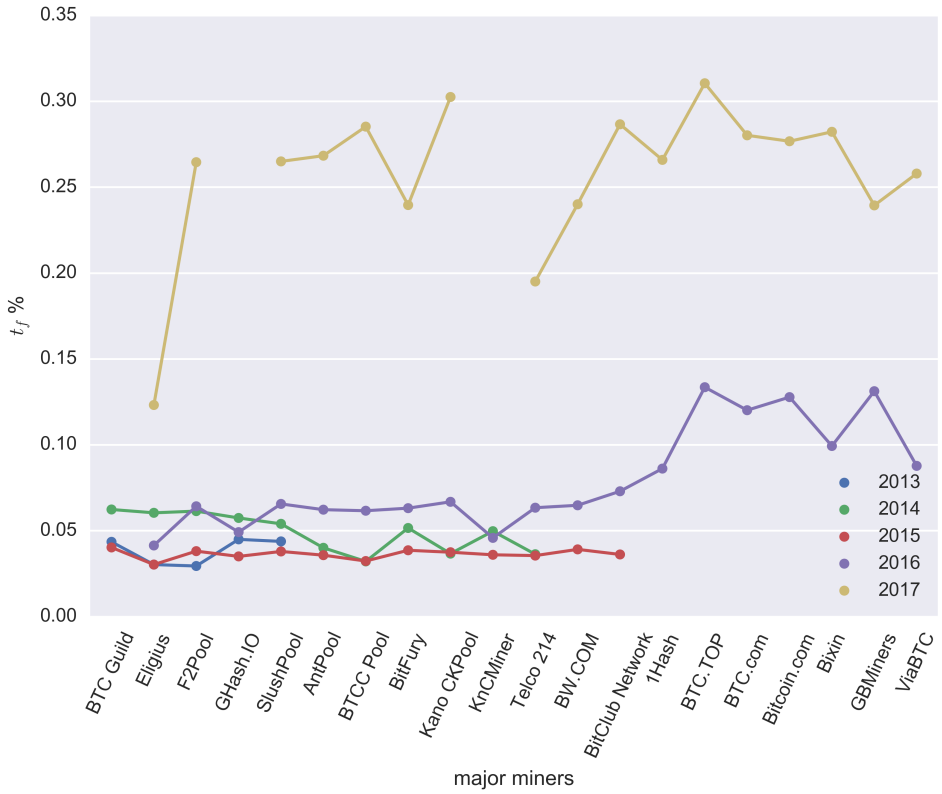


Figure 4.15: Average % of t_f paid from users to the top 20 miners of all times, divided by year.

there are hundreds, thousands or ten thousands of miners in the network, they will approve anyway 3-4 txs/sec but they will use, accordingly, more consumption of electricity. Now, if we make assumptions on transaction's consumption, considering a relatively low number of miners, with peaks of 150000 and all of them using AntMinerS9, we would have a consumption in between 400 and 500 Kwh per transaction, and if we grant that an average consumption of a house in the United States, which is one of the highest in the world, is of 11000 Kwh per year, consuming then about 30 Kwh per day, we can say that Bitcoin is anything but a green choice. The problem of Bitcoin consumption is that it only depends on how many miners are active in the network and if the Bitcoin price is rising then there could be more eager individuals willing to join, causing an increment of the difficulty to solve the proof-of-work so the system is anyways able to produce blocks every 10 minutes with a defined maximum size of 1 MB, but the overall consumption of the network will be higher. So every time the difficulty is raised also the consumption will increase accordingly. It might be good then to mine in places where the electricity is produced using renewable energy and to avoid the inclusion of stand alone

miners which would work without ever mining a block, consuming a lot of energy. Even if our evaluation shows a downward esteem, the Bitcoin economy has a consumption only of a thousandth if compared to the whole oil industry per hour. We estimate for Bitcoin a consumption of 4.32 Gwh per hour while in the Statistical Review of World Energy from BP [19] they show that the oil industry has a consumption of 5600 Gwh per hour. Until Bitcoin would be that competitive for eager miners, then the difficulty is intended to grow leading the cryptocurrency into a really costly and consuming scenario, plus always less miners will be able to enjoy reward benefits and they will end up mining for nothing with more electricity consumption and consequence of that would be the increment of the fees, but that would attract even more miners and here we are at the earlier scenario. A solution to that might be to regulate the miners traffic by controlling and auto-adjusting the number of miners which can join the network, in that way the fees are contained and regulated, having also a content consumption.

5

Conclusions

This Chapter aims to highlights the most relevant observations while running our blockchain analytics system and to proof whether our assumptions were accurate or not, providing an explanation and also an esteem of how the system might change in the next years. We compare previous results with ours and we give an opinion about the hot topics in Bitcoin, regarding the block size, the fees and miner's profit and then we discuss and test the accuracy of our prediction models. We evaluate that scalability issues brought to a centralization problem, the Bitcoin system, when it comes about mining, from distributed become decentralized due to mining pools. Then we state that to increase the system performance and scalability an eventual change on $\mathcal{T}$ and Q should be considered with all the pros and cons connected. Finally we inferred prediction models about the miner's profit and the transaction's latency according to the fee paid and the fee density for each transaction, and this is good either for miners and users so the firsts need to have a guaranteed gain in mining while the seconds need to get a faster confirmation time by optimizing their fees.

5.1 Results

Our main results are focused on fees and tolls but we evaluate also the scalability and the performance of the system by showing our analyzed data and our assumptions about how the system might improve performance and scales

more. When Bitcoin was first implemented, one of its strength was the decentralization. Miners could join the network all over the world and more miners meant a more secure network. During the years though, the reward enticed always more people around the world and new miners kept joining the network increasing in that way the difficulty to solve the puzzle, since the creation time should be kept fixed at 10 minutes. Consequently, the Bitcoin hashing power kept growing, so it became more difficult for single individuals to mine new blocks, decreasing in that way the chances for miners to gain their reward and leaving space in the network to mining pools, since they can gather up miners as well as their hashing power. This led to a scenario where singular individuals need to join bigger mining pools and the whole network became from distributed to decentralized since mining pools are controlled by third parties society so they control a big portion of the hashing power needed to mine new blocks. Furthermore, large mining pools nowadays withhold information about number of miners, the hardware used or they profit, making the centralization in the system even more enhanced. Being one of Bitcoin value proposition trustlessness, Bitcoin is only useful if is decentralized, since centralization requires trust. The other problem in scalability regards the amount of transactions to be approved every day by the Bitcoin system, and we have seen that this number is constrained by Q, $\mathcal{T}$ and t_q and since the transaction size will always be around 400-500 bytes, we consider a change in the block size Q and in the block creation time $\mathcal{T}$. Table 5.1 shows the results and our assumptions after a longitudinal study on the Bitcoin blockchain. Decreasing Q might appear a good solution for the number of pros it has, but the only two cons are critical for both performance and scalability matters. For that reason a decrease on the block size is ill-judged, while it could be much easier to deal with the orphan rate amplification. We have instead an opposite scenario with the block creation time. An increment would be ill-judged since the system will not be scalable, will be less performing and miners will have less profit while mining with the only pro to have a lower orphaning rate.

As we can observe from Table 5.1 throughput γ increase when either the block size Q is raised or the creation time $\mathcal{T}$ is lowered, a good compromise of both might be the solution to part of the scalability and performance problems in the Bitcoin network. While we can not define a relation between Q and t_f, since they are related only when a drastic change in the block size is made in the network, we can state that, from 2013 to 2017 the relation between t_f and t_l is more and more noticeable, having almost an inverse proportionality in the interval which interest us like showed in Figure 4.8 and that we represent with the parable in Equation 4.2. In 2017 0-fee transactions almost disappeared from the system and that is due to the incredibly high latency they were facing, since it took an average of 33 hours to get confirmed and included in a block by some miner. If only this fee is raised less than 0.0002 Ƀ, the transaction latency drops to an average of 5 hours, and with a fee in between 0.0008 Ƀand

Table 5.1: Scalability and performance scenario and consequences if Q and $\mathcal{T}$ are increased or decreased.

	Higher ↑	**Lower ↓**
Q	+ more scalability and transactions accepted per day + less latency t_l −/+ lower fees, good for users bad for miners − Orphan rate amplification − Damage to centralization − Congestion concern solved with transaction eviction by miners − temporary solution	+ no transaction spam + no 0-fee transactions + less mining cost + less propagation time + less chance of orphaning −/+ higher fees, good for miners bad for users − less throughput − more latency t_l
$\mathcal{T}$	+ orphaning rate much lower + no physical changed needed to support faster inner node communication − lower throughput γ unless Q is increased − system not very scalable unless Q is increased − profit $\langle \Pi \rangle$ is confined	+ higher throughput γ + system is more scalable + profit $\langle \Pi \rangle$ much higher for miners +/− not much information about t_l but for sure it will not drastically increase − faster inner node communication is required with possible changes in the physical structure − exponential increment of orphaning rate

0.001 Bthe average expected latency is less than 1 hour.

Regarding fees and tolls in the network we estimate the profit for each block of a miner using AntMiner S9 in relation with its block creation time. Once we collected enough data and calculated the profit, we inferred a possible trend for $\langle \Pi \rangle$ and called it *Miner Profit Function*, $f_{\langle \Pi \rangle}$. We used Numpy libraries [14] for the interpolation (Appendix C.10) and we used a polynomial of 2 and 39 degrees to see how the trend changes if the polynomial level is increased. Figure 4.9 shows both functions interpolated plus the samples for each block in 2017 and we define $f^{2}_{\langle \Pi \rangle}$ in Equation 4.1 while we use $f^{39}_{\langle \Pi \rangle}$ more like a reference to see where the samples tend to go in a real scenario. We state that for the first 2 minutes and after 20 minutes is not profitable to produce new blocks, while is good to have a creation time in between 3 and 8 minutes. We finally tested the accuracy of our Miner Profit Function, $f^{2}_{\langle \Pi \rangle}$, using scikit-learn libraries [38] and calculating the MAE as shown in Appendix C.11. We managed to have an high accuracy on this function, with a median absolute error MAE = 0.00001265 B.

Equation 4.1

5.2 Discussion

5.3 Future Implementation

5.4 Comments

References

[1] Antminer s9 - bitcoin mining hardware. https://shop.bitmain.com/productDetail.htm?pid=00020171024170217293t9Sp5rm406A9.

[2] Antpool, mining pool in bitcoin network. https://www.antpool.com/,.

[3] Bitcoin client application. https://bitcoin.org/en/bitcoin-core/.

[4] Bitcoin miners ditch ghash.io pool over fears of 51% attack. https://www.coindesk.com/bitcoin-miners-ditch-ghash-io-pool-51-attack/.

[5] Bitcoin prices calculated every minute. https://coindesk.com.

[6] The bitfury group, your leading full service blockchain technology company. http://bitfury.com/.

[7] Bitocoin blockchain analytic website. https://blockchain.info.

[8] Block size controversy - bitcoin wiki. https://en.bitcoin.it/wiki/Block_size_limit_controversy.

[9] Btcc mining pool in bitcoin network. https://www.btcc.com/.

[10] Ethereum's white paper. https://github.com/ethereum/wiki/wiki/White-Paper.

[11] F2pool, mining pool in bitcoin, ethereum and litecoin network. https://www.f2pool.com/.

[12] Foreign exchange rate and currency conversion, based on coindesk.com. https://pypi.python.org/pypi/forex-python.

[13] Matplotlib for data plotting. https://matplotlib.org.

[14] Numpy libraries manual – scipy. https://docs.scipy.org/doc/numpy/

index.html.

[15] pandas: Python Data Analysis Library. http://pandas.pydata.org/, 2012.

[16] Satoshi dice - gambling with bitcoin. https://satoshidice.com/rules, 2012.

[17] Bitcoin mining process. http://bitcoinminer.com/, 2015.

[18] Bitcoin website – mining. https://www.bitcoinmining.com, 2016.

[19] Bp: Statistical review of world energy. https://www.bp.com/en/global/corporate/energy-economics/statistical-review-of-world-energy.html, 2016.

[20] proof of work wiki. https://en.bitcoin.it/wiki/Proof_of_work, 2016.

[21] tradeblock.com – analysis on the blockchain. https://tradeblock.com, 2016.

[22] Paul Baran. *On Distributed Communications: Introduction to Distributed Communication Networks*. The Rand Corporation, 1964.

[23] M. Bowles. *Machine Learning in Python: Essential Techniques for Predictive Analysis*. Wiley, 2015.

[24] Leo Breiman. Random forests. *Mach. Learn.*, 45(1):5–32, October 2001.

[25] Rainer Böhme, Nicolas Christin, Benjamin Edelman, and Tyler Moore. Bitcoin: Economics, technology, and governance. *Journal of Economic Perspectives*, 29(2):213–38, May 2015.

[26] David Chaum. Blind signatures for untraceable payments. In D. Chaum, R.L. Rivest, and A.T. Sherman, editors, *Advances in Cryptology Proceedings of Crypto 82*, pages 199–203, 1983.

[27] Kyle Croman, Christian Decker, Ittay Eyal, Adem Efe Gencer, Ari Juels, Ahmed Kosba, Andrew Miller, Prateek Saxena, Elaine Shi, Emin Gün Sirer, Dawn Song, and Roger Wattenhofer. *On Scaling Decentralized Blockchains*, pages 106–125. Springer Berlin Heidelberg, Berlin, Heidelberg, 2016.

[28] Michael Crosby, Nachiappan, Pradan Pattanayak, Sanjeev Verma, and Vignesh Kalyanaraman. Blockchain technology: Beyond bitcoin, 2016.

[29] C. Decker and R. Wattenhofer. Information propagation in the bitcoin network. In *IEEE P2P 2013 Proceedings*, pages 1–10, Sept 2013.

[30] Rui Garcia, Rodrigo Rodrigues, and Nuno Preguiça. Efficient middleware for byzantine fault tolerant database replication. In *Proceedings of the Sixth Conference on Computer Systems*, EuroSys '11, pages 107–122, New York, NY, USA, 2011. ACM.

[31] Nicolas Houy. The economics of bitcoin transaction fees. Working Papers 1407, Groupe d'Analyse et de Théorie Economique (GATE), Centre national de la recherche scientifique (CNRS), Université Lyon 2, Ecole Normale Supérieure, 2014.

[32] Håvard D Johansen, Robbert van Renesse, Ymir Vigfusson, and Dag Johansen. Fireflies: A secure and scalable membership and gossip service. *ACM Transactions on Computer Systems (TOCS)*, 33(2):5:1–5:32, May 2015.

[33] Leslie Lamport, Robert Shostak, and Marshall Pease. The byzantine generals problem. *ACM Trans. Program. Lang. Syst.*, 4(3):382–401, July 1982.

[34] Aldelir Fernando Luiz, Lau Cheuk Lung, and Miguel Correia. Mitra: Byzantine fault-tolerant middleware for transaction processing on replicated databases. *SIGMOD Rec.*, 43(1):32–38, May 2014.

[35] Loi Luu, Duc-Hiep Chu, Hrishi Olickel, Prateek Saxena, and Aquinas Hobor. Making smart contracts smarter. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, CCS '16, pages 254–269, New York, NY, USA, 2016. ACM.

[36] Malte Möser and Rainer Böhme. *Trends, Tips, Tolls: A Longitudinal Study of Bitcoin Transaction Fees*, pages 19–33. Springer Berlin Heidelberg, Berlin, Heidelberg, 2015.

[37] Satoshi Nakamoto. Bitcoin: A peer-to-peer electronic cash system," http://bitcoin.org/bitcoin.pdf, 2008.

[38] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[39] Peter R. Rizun. A transaction fee market exists without a block size limit. Technical report, 2015.

[40] Chaitya B. Shah and Drashti R. Panchal. Secured hash algorithm-1: Review paper. Technical report, Indus Institute of Technology and Engineering, Gujarat Technological University, 2014.

[41] Sarah Underwood. Blockchain beyond bitcoin. *Commun. ACM*, 59(11):15–17, October 2016.

[42] Ben Vandiver, Hari Balakrishnan, Barbara Liskov, and Samuel Madden. Tolerating Byzantine Faults in Transaction Processing Systems Using Commit Barrier Scheduling. In *ACM SOSP*, Stevenson, WA, October 2007.

[43] Michael Waskom, Olga Botvinnik, drewokane, Paul Hobson, David, Yaroslav Halchenko, Saulius Lukauskas, John B. Cole, Jordi Warmenhoven, Julian de Ruiter, Stephan Hoyer, Jake Vanderplas, Santi Villalba, Gero Kunter, Eric Quintero, Marcel Martin, Alistair Miles, Kyle Meyer, Tom Augspurger, Tal Yarkoni, Pete Bachant, Mike Williams, Constantine Evans, Clark Fitzgerald, Brian, Daniel Wehner, Gregory Hitz, Erik Ziegler, Adel Qalieh, and Antony Lee. seaborn: v0.7.1 (june 2016), June 2016.

[44] Dr. Gavin Wood. Ethereum: A secure decentralised generalised transaction ledger. Technical report, 2014.

A

Terminology

RLP: Stands for recursive length prefix. It is a serialization method for encoding arbitrary structured binary data (byte arrays).

KEC-256: Another serialization method generating a 256-bit hash.

full node: A full node in a decentralized digital currency peer-2-peer network, is a node that stores and processes the entirety of every block, storing locally the entire size of the blockchain.

light node: A light node in a decentralized digital currency peer-2-peer network, is a node that only stores the part of the blockchain it needs.

satoshi: Unit of the Bitcoin currency. 100,000,000 satoshi are 1 BTC (Bitcoin).

B

List of Symbols

t_B	number of transaction approved in a block B.
t_{in}	transaction input in bitcoin (Ƀ). All the money sent.
t_{ou}	transaction output (Ƀ). All the money received.
t_f	transaction fee (Ƀ) (Equation 3.3).
t_q	transaction size, in bytes.
t_l	commit latency of a single transaction (Equation 3.4).
t_h	transaction height. Height of the block in which the transaction is included.
t_{ha}	transaction hash.

$t_{\%}$	percentage of t_{in} paid in fee, t_f.
$\mathcal{T}$	expected block interval time (~ 10 min)
$\mathbb{P}_{orphan}$	probability that given a block is orphaned.
τ	block solution propagation time, we consider a $\tau = 15.7$ seconds according to Croman [27].
t_{epoch}	timestamp of a transaction t. Epoch of when t was first seen in the network
η	cost per hash.
$\langle \Pi \rangle$	expectation value of a miner's profit per block.
$\langle V \rangle$	expectation value of a miner's revenue per block.
$\langle C \rangle$	expectation value of a miner's hashing cost per block.
R	block reward, currently at 12.5 Ƀ, halved every 210,000 blocks.
h	miner's individual hash rate.
H	total hash rate of Bitcoin network.
Q	block size or block space in bytes.
Q^*	the block size that maximizes the miner's expected profit.
ρ	fee density, or the price per byte for block space.

M	money in ฿. Sum of all t_f in a block.
$M_{demand}(b)$	partial sum of the b transaction fees in mempool in order of descending fee density.
$M_{supply}(Q)$	miner's cost due to orphaning to produce a certain block size Q.
$\mathcal{N}$	the set of transactions in a miner's mempool.
n	number of transactions in a miner's mempool.
B	single block.
B_t	transaction root that links to every transaction in a block B.
B_{epoch}	timestamp of a block B. Epoch of when the block was included in the blockchain
B_h	block height.
B_{ha}	block hash.
B_{mi}	miner which mined the block B.
γ	throughput of Bitcoin network, measured in txs/sec.
D	data frame generated by the analytic system.

C Listing

Listing C.1: Creation of Pandas data frame

```
1 import pandas as pd
2 # a1, a2, a3 = lists of attributes
3 df = pd.DataFrame.from_items([('label1', a1), ('
    label2', a2), ('label3', a3)])
```

Listing C.2: Union of df1 and df2 in a new data frame new_df

```
1 # df1, df2 = DataFrame
2 new_df = pd.concat([df1, df2])
```

Listing C.3: Group by attributes 'a1' and 'a2'. After that the mean, the sum and the median is calculated on the other attributes. The method reset_index() returns a data frame with the original attributes before the groupby was applied.

```
1 grouped_df = df .groupby(['a1', 'a2']).mean().
    reset_index()
2 grouped_df2 = df.groupby(['a1', 'a2']).sum().
    reset_index()
3 grouped_df2 = df.groupby(['a1', 'a2']).median().
    reset_index()
```

Listing C.4: Count how many occurrences for the attribute 'a1' and save this number in a new attribute called 'size'.

```
df = df.groupby('a1').size().to_frame('size').
    reset_index()
```

Listing C.5: Function for data manipulation. It creates a new column ('date'), from another ('B_ep') containing the respective B_{epoch} transformed in date time value with days as granularity.

```
def epoch_date_dd(df):
  # get a data frame with a column of epoch 'B_ep',
     returns
  # another column with the date yyyy-mm-dd so it
  # orders the date by day
  # :param df:  dataframe in input
  # :return:    new dataframe containing the 'date'
     attribute
  df['date'] = df['B_ep'].apply(epoch_datetime)
  df['date'] = df['date'].apply(revert_date_time)
  df['date'] = df['date'].str.slice(start=0, stop
     =10)
  return df
```

Listing C.6: Data retrieval using RESTful APIs provided from `blockchain.info`.

```
import urllib2
import json
global block_hash_url
block_hash_url = "https://blockchain.info/rawblock/"

def get_json_request(url):
  # Read the url and get json data.
  # :param url: str, site where to fetch information
  # :return: str, data requested in json format
  json_req = urllib2.urlopen(url).read()
  request = json.loads(json_req)
  return request

# get a block given an hash
block = get_json_request(block_hash_url + hash)

# get the previous block through block attribute '
   prev_block'
hash = block['prev_block']
```

Listing C.7: Block object structure obtained using Bitcoin's APIs

```
class Block:
  def __init__(self, b):
    self.hash = b['hash']
    self.version = b['ver']
    self.previous_block = b['prev_block']
    self.merkle_root = b['mrkl_root']
    self.time = b['time']
    self.bits = b['bits']
    self.fee = b['fee']
    self.nonce = b['nonce']
    self.n_tx = b['n_tx']
    self.size = b['size']
    self.block_index = b['block_index']
    self.main_chain = b['main_chain']
    self.height = b['height']
    self.received_time = b.get('received_time', b['
       time'])
    self.relayed_by = b.get('relayed_by')
    self.transactions = [Transaction(t) for t in b['
       tx']]
    for tx in self.transactions:
        tx.block_height = self.height
```

Listing C.8: Transaction object structure obtained using Bitcoin's APIs.

```
class Transaction:
  def __init__(self, t):
    self.double_spend = t.get('double_spend', False)
    self.block_height = t.get('block_height')
    self.time = t['time']
    self.relayed_by = t['relayed_by']
    self.hash = t['hash']
    self.tx_index = t['tx_index']
    self.version = t['ver']
    self.size = t['size']
    self.inputs = [Input(i) for i in t['inputs']]
    self.outputs = [Output(o) for o in t['out']]

    if self.block_height is None:
       self.block_height = -1
```

Listing C.9: Portion retrieval having a jump $J = 10$ and a number of blocks retrieved per time $b = 10$.

```
global latest_block_url
latest_block_url = "https://blockchain.info/
   latestblock"

jump = 10
b = 10
if(os.path.isfile(name)):
  # retrieve data frame from name.csv file
  df = pd.DataFrame.from_csv(name, sep='\t')
  hash_list = df['B_h'].values
  # get the last height
  height_list = df['B_he'].values
  last_block = height_list[-1]
  # subtract the jump
  last_block = int(last_block) - jump
  # get the block where to start the new fetching
  b_array = get_json_request("https://blockchain.
     info/block-height/" + str(last_block) + "?
     format=json")
  blocks = b_array['blocks']
  b = blocks[0]
  #  lget the block hash that has to be fetched
     first
  block_hash = b['hash']
  # call the method for fetching the blockchain
  get_blockchain(b, block_hash)
else:
  # file does not exist
  # retrieve the last block hash
  latest_block = get_json_request(latest_block_url)
  block_hash = latest_block['hash']
  get_blockchain(b, block_hash)
```

Listing C.10: Polynomial interpolation on miner's profit, $\langle \Pi \rangle$, and creation time, $\mathcal{T}$, using Numpy libraries.

```
x = df['B_T'].values
y = df['profit'].values
new_x, new_y, f = polynomial_interpolation(x, y,
    degree=2)

def polynomial_interpolation(x, y, degree=2):
  # given two lists of data it generates two new
     lists containing the y values interpolated
  # :param x : x values of the data to interpolate
  # :param y : y values of the data to interpolate
  # :param degree : polynomial degree
  # :return : x and y interpolated values. f is the
     polynomial.
  # order lists
  together = zip(x, y)
  sorted_together = sorted(together)
  x_vals = [el[0] for el in sorted_together]
  y_vals = [el[1] for el in sorted_together]
  # calculate polynomial
  z = np.polyfit(x_vals, y_vals, degree)
  f = np.poly1d(z)
  x_new = np.linspace(x_vals[0], x_vals[-1], len(
     x_vals))
  y_new = f(x_new)
  return x_new, y_new, f
```

Listing C.11: MAE accuracy calculation on miner's profit $\langle \Pi \rangle$ calculated using scikit-learn libraries in Python.

```
from sklearn.metrics import mean_absolute_error
new_x, new_y, f = polynomial_interpolation(x, y,
    degree=2)
predicted = []
samples = df['B_T'].values
real = df['profit'].values
for float(s) in samples:
  predicted.append = f(s)
mae = mean_absolute_error(real, predicted)
```