Investigating the Bitcoin blockchain through the analysis of empty blocks

Università Ca' Foscari Venezia, Italy

Software Dependability and Cyber Security

Andrea Sina 841723

Supervisor Prof.Andrea Marin

Academic year 2018/2019

Contents

To my beloved family and my friends.

Abstract

In the recent years, cryptocurrencies have become more and more pervasive in everyday life. Bitcoin is the most famous and the most used one.

This technology is based on the Blockchain, a continuously growing public list of blocks. Blocks are made up of transactions that describe exchange of coins among entities that own a Bitcoin wallet.

Since the creation of the currency in 2008, this structure has recorded all the information about transactions and blocks.

The security of the system is based on a widespread network that prevents modification at data registered in the blockchain and the use of cryptography that secures the transactions between two parties.

Bitcoins are created through mining, a process that requires the miners (users who want to participate actively in the creation of Bitcoins) to solve the proof of work, a problem of finding a known hash starting from part of the input.

When the correct hash is found, a new block with the coinbase, or the starting, transaction is created and added to the blockchain. As a reward, a certain amount of Bitcoin is given to the miner who solved the hash.

In this dissertation, we will provide an analysis of some interesting aspects of the blockchain.

The information about blocks and transactions are retrieved by parsing, interpretation by converting raw data into readable data, a section of the blockchain. In particular we considered the time-span between 1st January 2013 to 11th October 2018.

The most intriguing aspect developed in this thesis is related to the so called empty blocks, that are characterized by not containing any transaction except the coinbase, or the starting, one.

We analyzed how empty blocks are mined and when it is profitable not to add any transaction in a mined block. Moreover, we studied the correlation of empty block generation with the propagation delay of blocks and transactions in the bitcoin network.

We propose a model that aims to predict the behaviour of the Bitcoin system in the setting of an infinite pending transaction queue and test the goodness of the prediction by comparing the results with the real world data.

Finally, we will address the drawbacks of increasing the actual size limit of a block and how this change impacts on the energy wasted by the miners and the goodput of the system.

Introduction

Bitcoin is the first and most used type of cryptocurrency, digital coins that rely on cryptography and a dense network of active members, the miners, to exist. Initially, we provide a theoretical background on Bitcoin technology, what differentiates it from the traditional type of currencies and how they have become so used nowadays. Hence, we examine the blockchain, i.e. the structure at the base of its activity, and how it is constructed and maintained.

The objective of this dissertation is to study the nature of empty blocks, odd blocks that do not contain any user transactions and to delve into the question of the block size limit.

The formation of empty blocks seems to be an unexpected and foolish behavior of miners whose sole aim is to make more profit from a single mined block. From a system prospective, empty blocks undermine the energy efficiency of the blockchain, since the computational power used to generate them does not bring any effective benefit on the reduction of the transaction backlog.

These blocks have become more and more frequent in the chain in latest times and it is interesting to understand which are the reasons that lead the miners to create them. To retrieve this information, we download the full ledger and used a parser to retrieve key statistics. We examine how vacant blocks are distributed in the blockchain and weather or not the gaps between these unique blocks are distribute in a known way.

To study the behavior of the Bitcoin system, we propose a predictive queuing model with the intention of estimating the presence of empty blocks according to the size of the blocks with the assumption of an infinite buffer for the pending transaction list.

Finally, we evaluate the drawbacks of expanding the size of blocks with a particular focus on the energy waste and the goodput of the system.

Structure of the thesis

This thesis is organized in 4 chapters: The first focuses on providing fundamental information about Blockchain and Bitcoin, and the way they work. We then provide an insight on the largest problems that are subsequently addressed: the construction of empty blocks and low system throughput. The second chapter addresses the issue of creating empty blocks. Starting with evaluating the problem, we see how information is propagated in the Bitcoin network. Then, with the assumption that the propagation delay is one of the causes that allow the creation of empty blocks, attempts to discover a good distribution to fit the empty block gaps data using a hypothesis test. A queuing model for the blockchain and its assessment is provided in Chapter 3 after seeing some of the primary queuing systems. The last chapter discusses the last issue, the drawbacks of raising the block size restriction specifically on the energy waste and the system goodput.

The topics covered in this dissertation are:

- What is the blockchain?
- What is bitcoin?
- How bitcoin works?
- The problem of empty blocks
- The issue of bitcoin throughput
- A queuing model for the BTC blockchain
- The drawbacks in increasing the blocksize

Chapter 1

Blockchain and Bitcoin

Chapter introduction

This chapter focuses on the introduction of Bitcoin technology and the framework on which its operation is based: the blockchain. We will see how blocks are produced and what is the technology that secures the blockchain. Then, we will look at a bit of bitcoins history: when they were formed and how they become such a powerful asset in the world economy. Moreover, we discuss the issue of empty blocks and see what makes them so special. Finally, we will mention the Bitcoin system throughput issue relative to other significant payment system.

1.1 Blockchain

1.1.1 What is the Blockchain

The Blockchain is a continuously growing public list of records, called blocks, which are linked and secured thanks to the use of cryptography.

Bitcoin blockchain is basically an open, distributed, ordered, public ledger that, since 2009 year in which the system was created, records all the transactions between two parties in an efficient, verifiable and permanent way. This ensures all those participating in the Bitcoin network to have complete access to all the transactions occurred and verified until that time and everyone can agree on the way each of these transactions took place.

Each block of the chain is composed by a hash pointer as link to the previous block, a time stamp and the transaction data composed by a group of one or

more transactions starting from the coinbase transaction made by miners.(see Section 1.1.3) Duplicates of each transaction are hashed, paired, hashed again until a single hash remains. This is the Merkle root of a Merkle tree, stored in the block header. When data are registered in the blockchain is unfeasible to retroactively alter block data without modify all its successive block. This would require the consensus of the majority of the network.

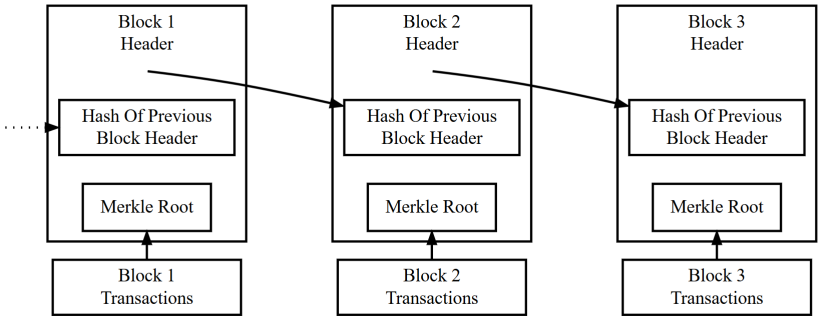

Figure 1.1: A simplified blockchain schema[1]

The use of blockchain is not limited at cryptocurrencies, instead this technology can be used for several other purposes like supply chain[2] or healthcare[3].

1.1.2 Merkle tree

A Merkle tree, or hash tree is a specific type of tree, usually binary but can be arbitrary, greatly used in cryptography. The peculiarity of this tree data structure is that:

- Every leaf node is labelled with the hash of a data block
- Every non leaf node is labelled with the hash of its respective children

The top hash, or Merkle root, is the hash of the entire tree. As Merkle tree uses hash they are used to efficiently and securely verify huge amount of data as they can be compressed in a single hash.

This structure is named after the American computer scientist Ralph Merkle, one of the co-inventor of public key cryptography, who published [4] and patented in 1979 [5]

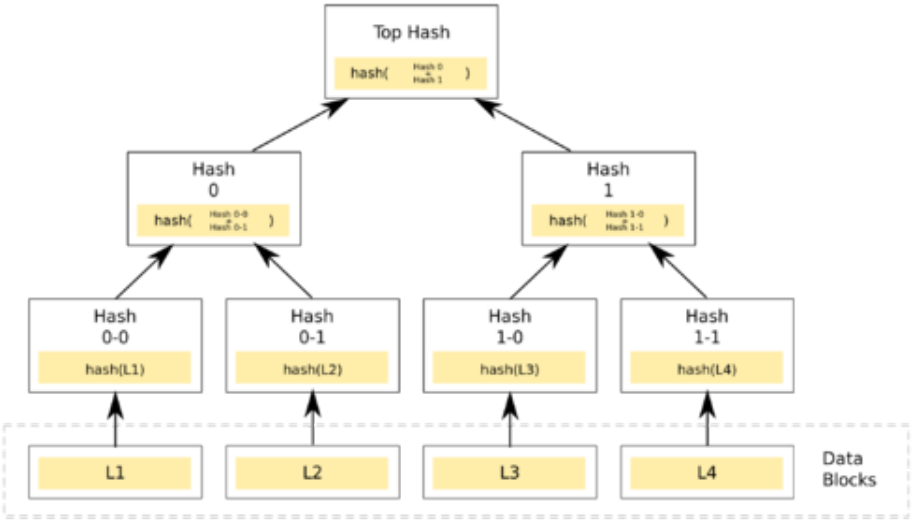

Figure 1.2: A Merkle tree

In Bitcoin, hash trees are used to store transactions in a way that optimizes the space utilization and the security. In our case, the leaves are the transactions that are consistently hashed and grouped together until the Merkle root is stored in the block header of each block.

The following Figure from Satoshi's 2009 paper [6] depicts the way Merkle trees are built and stored into each blockchain block.

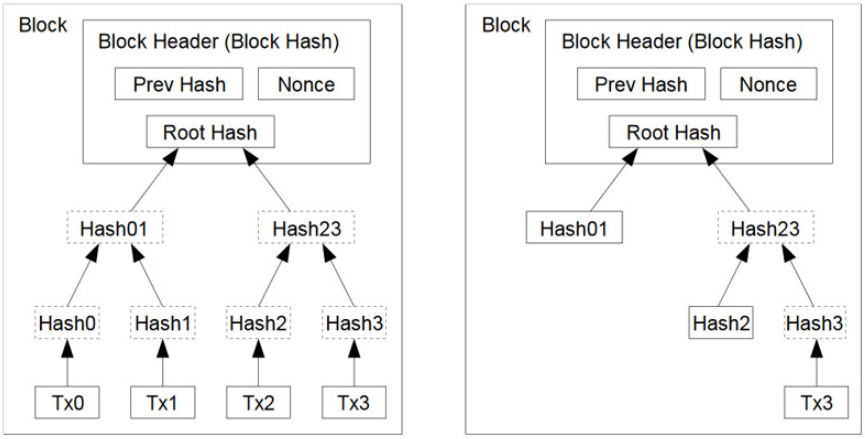

Figure 1.3: Merkle tree use in bitcoin

Using hash trees in Bitcoin serves to keep data integrity and allows unique nodes, known as Simple Payment Verification or SVP, not to download the entire blockchain to work on the network. Instead, they only need to download the valid chain block header, which is merely the shortest one. Then, SVP node can determine the status of any transaction using the hash root of a block part of the longest chain.

1.1.3 Blocks and transactions

Blocks are the "bricks" of which the blockchain is comprised. A block is mainly made up of Bitcoin transactions representing an exchange of cash (or in this case cryptocoins) between two peers.

Other components of Block are the magic number, a fixed hexadecimal value choosen by Satoshi when Bitcoin was designed, that uniquely identifies a message as a Bitcoin block; the blocksize that, as the name suggests, represents the size in byte of the block; the block header that contains vital data as the hash of the previous block; and the transaction counter that simply tells how many transactions are packed inside it.

This structure highlighted in the table below.

Field	Size	Description
Magic Number	4 bytes	Constant value 0xD9B4BEF9
Blocksize	4 bytes	Size of the block in bytes
Block header	80 bytes	List of 6 item
Transaction counter	1-9 bytes	Number of transaction in block
Tranactions	transactions size	List of transactions (non empty)

Table 1.1: Bitcoin block structure

As already said, the block header is composed itself from various parts as shown in Table 1.2.

Field	Size	Description	Updated when
Version	4 bytes	Block version number	New software update rolls out
HashPrevBlock	32 bytes	Sha-256 hash of the prevous block header	A new block comes in
HashMerkleRoot	33 bytes	Sha-256 hash of all the transaction in the block	A transaction is accepted
Time	4 bytes	Current timestamp in POSIX time format	Every few second
Bits	4 bytes	Current difficulty	The difficulty changes
Nonce	4 bytes	Incremental number used to solve the proof of work	An hash is tried

Table 1.2: Block header structure

A transaction is basically a transfer of values among Bitcoin wallets that is included after being confirmed in a blockchain block. The transaction is signed with the private key contained in the wallets, giving the mathematical proof that they come from the wallet of the owner.

Since each customer secretly stores the private key, the signature prevents some malicious customers from altering a transaction.

Table 1.3 illustrates the composition of a Bitcoin transaction.

Field	Size	Description
Version number	4 bytes	Currently 1
Flag	2 bytes array	[Optional] If present indicates the presence of witnesses
In-counter	1-9 bytes	Number of input values
Inputs	Size of inputs	List of inputs
Out-counter	1-9 bytes	Number of output values
Outputs	Size of outputs	List of outputs
Witnesses	Size of witnesses	A list of witnesses, one for each input (SegWit)
Lock_time	4 bytes	The earliest time a transaction can be added to the blockchain

Table 1.3: Bitcoin transaction stucture

1.1.4 Hash and proof of work

Hash is a cryptographic function, a mathematical algorithm mapping arbitrary length data in a fixed length binary string (the hash value or message digest). This feature is made to be unidirectional, or rather non-invertible: there is no way to get the digest from the hash value. The only way to do that is through a brute-force research / attack.

The most commonly used hashing algorithm in Bitcoin is *SHA-256*, implemented for both bitcoin address generation and proof-of-work.

With proof-of-work, we indicate the activities of miners that consist of calculating the hash of the block header with regard to certain limits. The block involves, among other things, a reference to the previous block, a transaction group hash and a nonce.

In cryptography, the nonce (number used once) denotes a number, random or pseudo random, used only once. This amount is increased constantly until the hash value calculated on the block produces value starting with a certain amount of zero bits.

This operation is performed by each miner millions of times per second. When a miner computes the nonce that satisfies the presence of the exact number of zeros, the block is flooded in the network that checks if the solution is correct. Only when the new block has been validated by certain nodes, named the validators, this eventually becomes valid and the miner earns its income.

The quantity of bitcoin that a miner gains as a reward for his job is intended to halve every 210,000 blocks, or usually four years; when bitcoin began in 2008, it was set as 50 bitcoins for each mined block, fell to 25 in 2012.

Nowadays the reward is 12.5 BTC, after the third halving that happened in 2016, correspoinding at the actual change to 102.507,56 euros.

To compensate for the natural development of hardware speed and the variation in the interest of the operating nodes over time, the proof-of-work toughness / difficulty is determined by a moving average(ref) that limits block manufacturing to about 10 blocks every 10 minutes. If blocks are produced too quickly, the difficulty will rise and the average amount of mined blocks will stay stable.

A block editing type of attack is preserved by the fact that if a malicious user wishes to edit an ancient block, he must solve the proof-of-work puzzle of that block and all its subsequent blocks until the work of legit nodes, which in the meantime continue to produce fresh blocks, is achieved and surpassed.

The probability of a malicious user reaching the job of honest nodes has been proved to decline exponentially with the increasing amount of blocks.

1.1.5 Mining

Mining is the process that adds a new block to the blockchain and at the same time makes the structure hard to change. The entities that realize it are called miners.

Mining can be solitary or pooled depending on the amount of customers involved in the process. In solo mining, the miner attempts to generate blocks himself, whereby the proceeds from the block reward and the transaction fees flow completely to him. This allows the miner to receive larger payments but implies harder work.

The software periodically surveys the network for new transactions, then builds a block and generates a block header that is sent to the mining hardware.

On the other side, in pool mining, miners are grouped into pools and collaborate to generate new blocks more frequently. However, in this case the profit is shared among the pool miners proportionally to the amount of work they each contributed. This leads every miner to have smaller payments, but more frequently.

1.2 Bitcoin

The term Bitcoin is used to indicate both the network and the currency. As mentioned before, this electronic currency was developed to fix the issues of trust, transparency and accountability between the parts in exchanging products and services online without any intermediate.

In fact, Bitcoin is the first payment network to rely on distributed peer-to-peer technology to operate without a central authority: the entire network collectively manages the transaction and the issuance of the currency.

Thanks to the distributed nature of the network, all transactions are collected to avoid double spending. All consumers have knowledge of what is happening in the network, so there is no need to have an entity that manages the transactions.

Because Bitcoin is an anonymous currency, they can be compared to cash money: it can be given to another person to buy goods or services and when spent you does not own it anymore. In addition, everything occurs without a third party acting as an intermediate.

The *bitcoin network* works thanks to a combinations of preexisting technologies: asymmetric cryptography, peer-to-peer network and proof-of-work/hash.

1.2.1 Asymmetric key cryprography

In Asymmetric key Cryptography each user own two pair of keys:

- A **public key**, used to encrypt information, that is shared with the receiver;
- A **private key**, used to decrypt received information, that remain secret

For example, if Alice wants to send a secret message to Bob, she will use Bob's public key to encrypt her message and send it to Bob. When Bob receives the message, use his secret key to decrypt the message. Since he is the only one in possession of his secret key, he is also the only one able to decrypt Alice's message. Bob's private key must be kept secret to guarantee communication security.

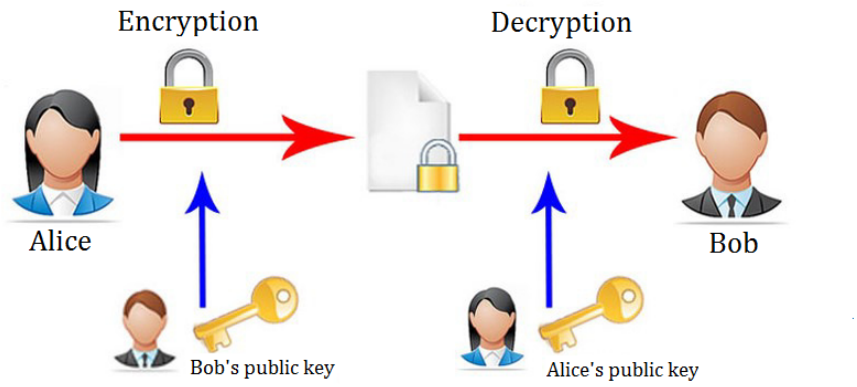

Figure 1.4: Communication secured with asymmetric key cryptografy

In our setting, the bitcoin address that must be exchanged to send and receive payments is the *public key* of the asymmetric cryptography cipher. Crypto is required in bitcoin to ensure the privacy of both the sender and the receiver making this sensitive information unrecoverable for any unauthorized user.

An example of communication between two users secured by Asymmetric Key Cryptography is illustrated in Figure 1.4.

1.2.2 Historical Background

Bitcoin is an open source project published in 2008 by Satoshi Nakamoto [6], an unknown author (or a group of authors). This paper reveals the first effort to establish a cryptocurrency, a fresh type of currency produced and regulated using cryptography instead of a central authority (such as a bank) like traditional currencies. Nakamoto allegedly started working on the Bitcoin project in 2007 while a first patent was filed on August 2008 concerning the distribution of encryption keys from N.King, V.Oksman and C.Bry [7], who, however, rejects any link with Satoshi. The domain Bitcoin.org was registered at anonymousspeech.com in the same month. That permitted the creator to remain anonymous. The first block, also known as the Genesis Block, was mined at 18:15:05 on January 3, 2009. The Bitcoin age had officially started.

To see the first transaction we had to wait until January 12 2009 when in the block 170 was added the first coins trade between Satoshi and Hal Finney, cryptography activist and pioneer of the cryptocurrency. Using an equation

that included the cost of electricity to mine a block, New Liberty Standard on October 2009 published the first exchange rate establishing a value of 1 USD = 1,309.03 BTC. The first difficulty increase hits the system on December 2009.

The first real-world transaction using bitcoin took place on May 2010 when a Bitcoin Forum user paid 10,000 BTC, which was worth 25USD at the time, to buy a pizza. On July 18, 2010, Jed McCaleb established the former largest bitcoin exchange market. Mt.Gox During that period, Bitcoin system was susceptible to several attacks, one of which triggered an inappropriate generation of 184 billion bitcoins. Slush's Pool, the oldest mining pool, and the first known to be publicly available. announced on November, 27, 2010 mined their first block later that year [8].

2011 was the year in which the difficulty achieved and exceeded the limit of 10,000 and the complete amount of bitcoins produced exceeded 25% of the predicted sum of nearly 21 million BTC. Bitcoins reached parity in exchange for US dollars for the first time in the first months of the same year at MtGox and passed parity with Euro in the middle of 2018. The cost a Bitcoin hits the all-time high of 31.91 USD in June. That was also the year of the so-called 2011 Great Bubble [9] when the bitcoins exchange rate at MT.Gox plummeted to 10 dollars just four hours after its hitting its peak of $33. In the same month, the exchange market suffered a massive security breach [10] witch led in a theft of over 60,000 accounts and a total amount of 500.000$ causing the cost to plunging at 0.01 $ per bitcoin. Meanwhile, the generation difficulty passes 1 million. The highest fee in a single transaction was recorded in the last month of the year, 171 BTC transacted in block 157235.

An FBI study [11] on May 2012 concluded that:

> *"[...]Bitcoin will likely continue to attract cyber criminals who view it as a means to transfer, launder, or steal funds as well as a means of making donations to groups participating in illegal activities, such as hactivists[...]"*

and

> *"[...]Bitcoin might logically attract money launderers, human traffickers, terrorists, and other criminals [...]"*

Highlighting the risk of using bitcoins to promote illegal purchases of weapons and narcotics. The biggest block for that time came to light in June comprising 1322 transactions. One of the main events taking place in 2012 was the first

halving at block reward, known as the Halving Day, on November 28. On the same day the price payed to miners for creating a block passed from 50 to 25 BTC. On January 2013 the price of a bitcoin reached back 30 US dollars for the first moment since the Great Bubble of 2011. The first hard fork on the blockchain took place on March 24, 2013 following a protocol rule that was originally unknown to everyone. In mid-year, the bitcoin price rises exceeding the limit of 100 US dollars achieving the all time high of 266 USD by the first days of April (the year before was only 13 USD). A big crash impacted the exchange by the middle of the same month following the hacking of Bitcoin Central, a famous BTC exchange market. On October 2, FBI closed down the notorious illegal internet drug marketplace Silk Road [12] capturing a quantity of 3.6 million USD of bitcoins. After that, the price collapsed from 139 $/BTC to 109 $/BTC in less than three hours, making it one of the highest cost crashes to date as higlited in Figure 1.5. In November, the market value skyrocketed above $1000 after a US Senate hearing about virtual currency legitimacy and challenges. By the middle of the month, Bitcoin accomplished a volume of transactions greater than those of Western Union.

2014 began in the worst possible fashion. Mt. Gox, the giant of bitcoin exchanges, filed for bankruptcy by the end of February [13] amid rumors that 744,000 bitcoins were stolen. As a result, the price dropped to 600 USD and reached the level of $340–$530 per bitcoin, the smallest value since the Cypriot crises in 2012-2013. The netework as a whole overcame the computational power record of 100 petahash / s in June. Microsoft, Dell and Newegg began adopting bitcoin by the end of the year. The cost dropped to 200-300 USD / BTC for the first few months of the next year. By the beginning of 2015, the amount of dealers that accepted bitcoin surpassed 100,000.

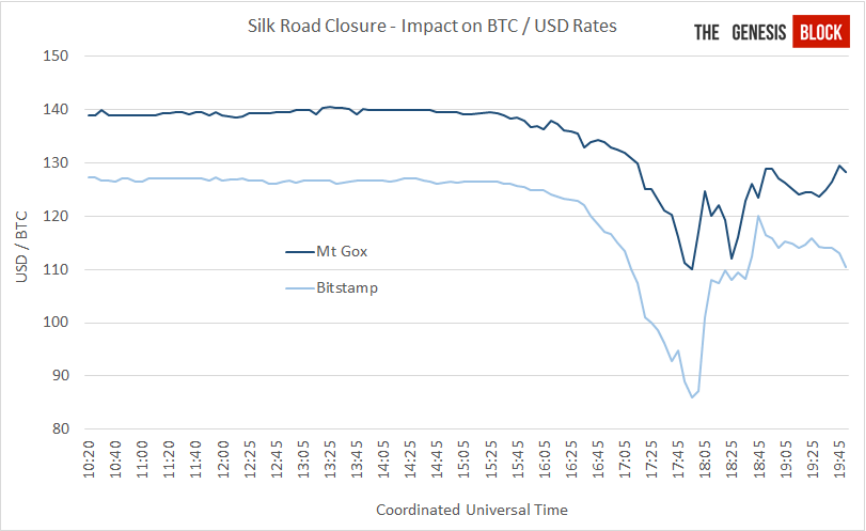

Figure 1.5: Price crash following Silk Road closure in 2013

The network speed has not slowed its growth, ending at 1 exahash / s (1.000.000 terahash/s) by the end of January 2016. The big online gaming platform Steam joined the other large enterprises that embrace bitcoin in April of the same year. August was the month Bitfindex, a leading bitcoin exchange market, after being hacked hack lost nearly 120,000 BTC, about 60 m $ given the exchange rate at the moment. Price remained near to $600/BTC. Bitcoins related academic papers rose in volume year by year from the 36 published in 2012 to over 1500 in 2016 (a detailed chart about publications on Bitcoin by year is available at [14]).

Nevertheless, the year was known as the year of the second halving. In fact, on July 10, 2016, the block reward halved from 25 to 12.t bitcoins [15].

In the first months of 2017, many nation legislators and financial companies began giving Bitcoin more credibility: Russia, Japan, and Norway began legislative processes for recognizing Bitcoin as a traditional method of payment in multiple forms. Bitcoin cash, the hard bitcoin blockchain fork that brings the block size limit up to 8 MB released on August 1, 2017 [16]. Steam announced in December that they would stop receiving bitcoin as payment for their products and services. By the beginning of the year the cost of a bitcoin reaches the abominable amount of $19,783.06. 2018 was the year in which the price began to fall to $6,200 in the first months to $3700 per bitcoin at the conclusion of the

year.

The future of Bitcoin is unsure. The price seems to rebound to lower levels at the moment of composing this dissertation, settling at around $8,000 as can be seen from figure 1.6, but any forecast is reckless. The only certainty is that with the current block mining rate, the third halving will be in 2020 [17].

Figure 1.6: Bitcoin price chart over the year

1.3 Peer to peer network

Peer-to-peer-network denotes the peculiar type of network in which all the users in the network are equally privileged. In these networks, the workload is distributed among peers, and the resources are shared without the need of a separate server machine. In fact, in contrast with the client-server architecture, each peer acts both as supplier and as consumer.

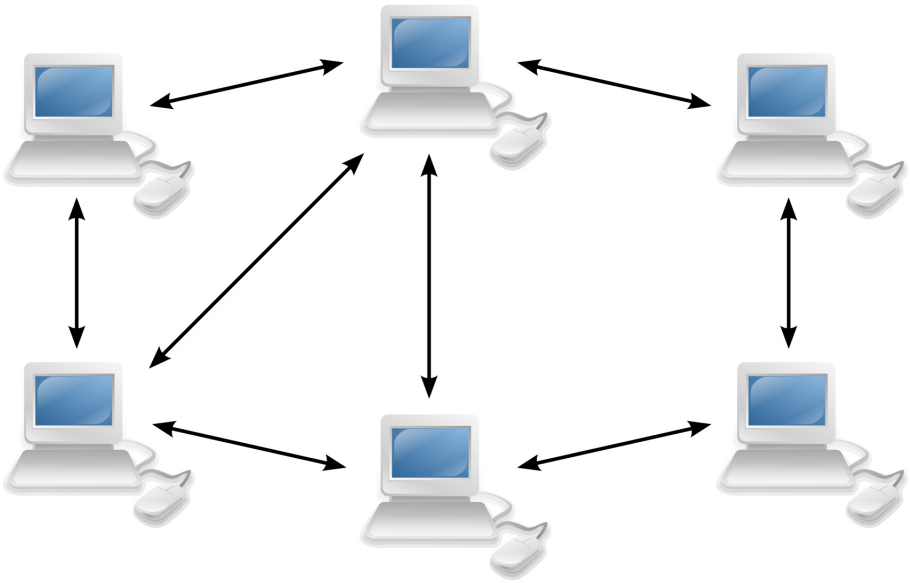

Figure 1.7: An example of peer-to-peer network

Bitcoin is the first form of a decentralized, peer-to-peer payment network. Unlike other payment systems and currencies, it does not rely on a central authority (usually a bank) to handle user-to-user transactions and to ensures security. Conversely, as specified by the Bitcoin protocol, this is granted by the structure on which the currency is based: the blockchain.

Each node is treated equally with the others and has a copy of the blockchain, physically stored on its drive. Every time a new valid block is added to the chain, each node receives the data and updates its local version of the ledger without any centralized distribution and authentication system.

1.4 Empty blocks

An empty block is a unique type of Bitcoin block sometimes found in the blockchain that is characterized by being devoid of transactions except from the *coinbase* transaction, the first transaction in a block used by miners to collect the block reward. The trivial example of an empty block is the block of genesis, the first block ever mined.

It is important to clarify that empty blocks do not lack data and the effort

to mine an empty blocks in terms of proof of work is the same as made to mine a non-empty block. Empty blocks still contain the block header which, as already seen, are 80 bytes and have the same data structure as non-empty ones. They still have their transaction list which, by its nature, includes only the coinbase transaction.

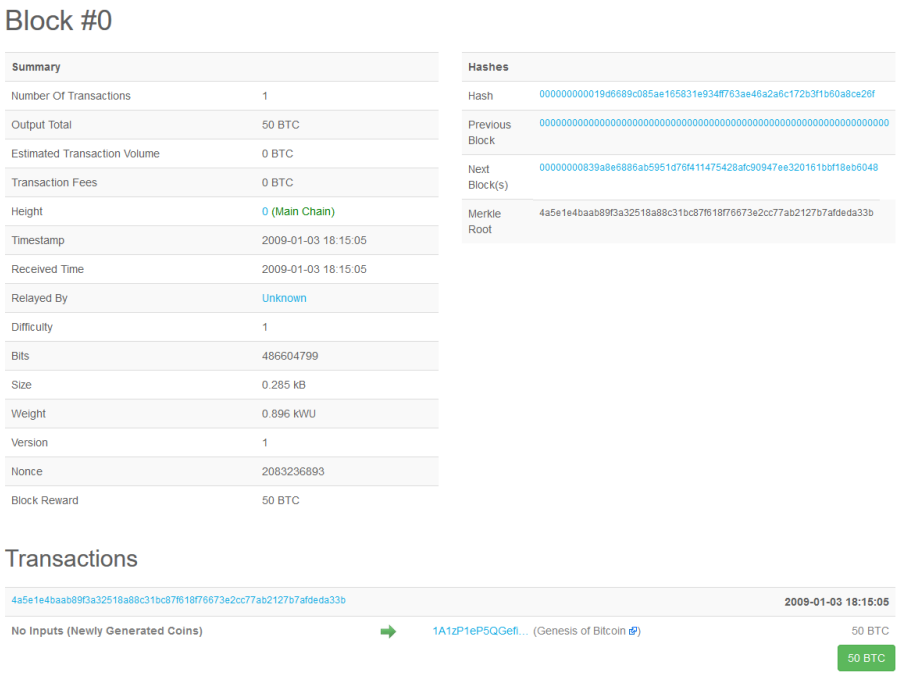

Figure 1.8: The genesis block is the trivial example of an empty block

Miners might have a disparate justification to mine an empty block instead of a full one. If the list of pending transactions is empty, miners will not just stop working. The system requires to switch on and mine of empty blocks is allowed for the blockchain to keep expanding. This occurred frequently when the Bitcoin system first started back in 2008, the network load was very small and the list of transactions was often empty. For this reason, the first mined blocks were mostly empty. The first non-empty block was mined on Jan 12, 2009 as the 170^{th}. Even if miners are advised to bundle as many transactions as possible inside a block to increase their profit from transaction fees, there is nothing to prevent them from not including any transaction inside a fresh new

block. Since both with a full block and with an empty one, they still obtain the BTC 12.5 award (at the moment of composing this dissertation).

Moreover, the creation of empty blocks is often used as a way to guarantee profits for miners. As mentioned before, the proof-of-work effort to make empty blocks or non-empty blocks is the same, but the validation time plays an important role in the transmission speed of the information in the network. With only one transaction, empty blocks are quicker to be verified and this could lead to a greater probability of solving the block's puzzle and thus gaining the reward before others. Furthermore, the calculation of the merkle-tree is much quicker if a block includes just one transaction since it is calculated only once.

Additionally empty blocks injecting new cash into the system make feasible an ever-increasing flow of the cryptocurrency among customers that leads the entire system to expand in notoriety. In parallel, their presence extend the blockchain and enhance the safety of the ledger by adding more work.

1.5 System throughput

According to many community members, the Bitcoin system has the purpose of supplanting the classical payment system with a decentralized, middleman-free model. However, due to its complexity and the restriction imposed by the Bitcoin protocol, it is certainly suffering from having a low throughput. Bitcoin protocol is designed to guarantee that blocks are mined every 10 minutes on average, for security reason, and a block has a maximum size of 1 MB (without contemplating the enhancement made by introducing *SegWit* transactions).

It follows that the Bitcoin system's peak theoretical throughput is restricted at 6MB / h with an approximate transaction processing capacity between 3.3 and 7 transactions per second.

K.Croman et al. in [18] faced the issue of small Bitcoin system throughput relative to conventional payment methods such as Visa. The payment giant has an average process rate of 2000 transactions per second and can sustain a peak of 56,000 transactions/sec in event of heavy load. The authors questioned whether or not decentralized blockchain could be scaled to at least suit the performance of a conventional payment processor and what it takes for Bitcoin to get there.

They concluded that "[...] reparametrization of the block size and interval in Bitcoin is only a first step toward substantial throughput and latency improvements while retaining significant system decentralization." Therefore, the

authors suggest that "More aggressive scaling will in the longer term require fundamental protocol redesign".

1.6 Chapter conclusion

In this chapter we have seen the basics of bitcoin technology, like how the proof-of-works make the blokchain a safe structure that needs most of the total computational power to be exploitable. We then focused on giving the definition of an empty block, which contains only the single coinbase transaction, and began to delve into the reasons that bring miners to produce them at the expense of normal blocks. This element needs further development since the amount of empty blocks in the last year has risen a lot.

We then confronted the problem of tiny system throughput that only 6MB / s made Bitcoin not even comparable to Visa and other large payment systems. On this we highlighted how according to researchers the change on block size is only a stopgap.

Chapter 2

The origin of empty blocks

Chapter introduction

We have already discussed the issue of empty blocks. This section focuses primarily on an empirical analysis this aspect of the blockchain.

Starting with a closer assessment of the issue of block size, already stated in Chapter 1.5 about system throughput, we will see how data is propagated throughout the Bitcoin network. Then, we will address the heart of this chapter, the study of empty block gaps, i.e., the gap between two empty blocks. We will examine how this data are distributed and considering a small part we will attempt to use a hypothesis test to measure how well it fits with a recognized distribution of probability.

2.1 The block size problem

The Bitcoin system is characterized by two main invariants:

- A block is mined averagely every 10 minutes
- The maximum block size is 1 MB

As mentioned in Chapter 1.5, the Bitcoin system's maximum throughput is 6 MB / h which is far too small compared to other large payment processors. For years, therefore, the Bitcoin community has been wondering about the best way to overcome this problem.

Since it is impossible to increase the number of blocks mined in 10 minutes for security reasons or to drop the average time in which a new block is found, the only possible solution is to raise the maximum block size limit. The average block time of 10 minutes, was specifically chosen by Satoshi when he introduced bitcoin [6] as a trade off between first confirmation time and the amount of work wasted due to chain splits. [19]

The restriction on block size was not always present, or precisely it was not acknowledged instantly. When Satoshi published his paper in 2008, the limit was not on the block's number of bytes, but instead on the number of database lock required to process it. Quantitatively, the average threshold was around 625k bytes. In 2010, Satoshi himself implemented a real maximum block size bound in a disguised commit. Because of the complexity of traffic in the system ago in those days, the restriction was not viewed until March 2013 when the Bitcoin Core update v0.8.0 was published to impose it on all Bitcoin nodes.

Unfortunately, it had a bad impact since the update contributed to a scenario where the upgraded nodes split the network in half. From that time on, because of this accident, the entire community was aware of this limitation. The problem was settled soon and the network members decided to build a hard fork to remove the restriction that was approved and successfully enabled in May 2013. From 2015, as the volume of transactions expands with the vast and increasingly faster adoption of the Bitcoin as a payment system, the debate on increasing block size is becoming very popular in the community and according to some exponents an hard fork to remove, or raise it seems unavoidable.

There are a lot of reasons to change the limit, as the maximum size of the block will increase more transaction would be packed inside of it and the overall number of transaction per second will increase, causing an increment of the system throughput. Moreover since a block will contain more transactions the total number of fees that miners collect will consequently increase, raising their total profit. It is important to note that as the number of bitcoins is limited and so the proof of work reward for the solver is doomed to decrease, the user fees will become the main source of earning for all the miners. In parallel as the number of transactions and then fees for block will increase, the average fee will decrease making Bitcoin system accessible to a wider audience of users.

Some other prominent community exponents claim that the size of the block should stay so [20] [21] [22]. From their subjective point of view, since the value of the single transaction charge will be lowered, miners will have fewer incentives for mine blocks. The hard fork, vital to implementing the shift, will split the

community creating a ton of trouble and expanding the network magnitude, and the consequent rise of the system's cumulative computing power along with increasing the mining difficulty, will take a handful of small miners to abandon the network. This, in their opinion, will cause the vast bulk of the network to be governed by a few large mining organizations and governments. Lastly they believe that bitcoins should not become an everyday currency and payment method, but instead stay restricted to a specific use.

SegWit, an acronym for segregated witness, is a soft fork of Bitcoin introduced in August 2017. It needs only a small software update to be adopted as a soft fork and it can still operate with nodes that use earlier protocol versions. This fork allows the limit to be raised to a theoretical maximum of 4 MB by detaching the witness (the part of the transaction certifying its validity) from the Merkle-tree, occupying bytes from a different space and allowing more transactions to be packed inside each single block. Hence, with SegWit the concept of block size has changed to the maximum size that a block can have if it doesn't validate inside any single SegWit transaction.

Segwit is now adopted from around the 40% of the total miners.

2.2 Information propagation in the Bitcoin network

2.2.1 Theoretical background

By building a random graph, Bitcoin network is where each node tries to keep a minimum number of links p to other nodes at all times. When the number of open connections become below the pool size p, (which default value is 8) the node randomly selects an address from its list of known node addresses and pursues to connect.

When a new node attempts to enter the network, it receives a batch of addresses of nodes, connects to them and retrieves information about the other participants by requesting addresses from their neighbours. If a node remains unreachable, it is flagged as disconnected from the network for several hours and its address is dropped from the list of nodes.

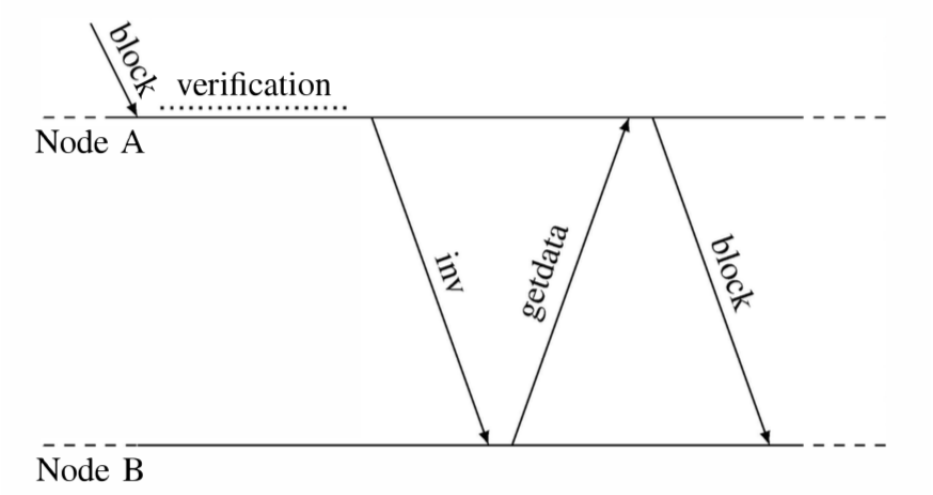

Figure 2.1: Message propagation in the network

When a node has verified a collection of blocks and transactions and is prepared to send it to its neighbours by sending a small *inv* message (most often of 61B). This message includes a set of ready-to-send transaction hashes and blocks hashes. When a node receives the message *inv* for a block or transaction that is not present in its local blockchain copy, send a *getdata* message back to the sender of the *inv* containing the hashes of the data required, whether they are blocks or transactions. Finally the real transmission takes place and messages of blocks and transactions are transmitted to the node in need.

The figure above from [23] explains the scenario completely. Node A received a block after the validation and is ready to forward to its neighbor, Node B. It then sends a *inv* message containing the block hash announcing its accessibility. Node B after receiving the message *inv*, considering that this is a new block, answers with the message *getdata*. Node A can then send the entire block message to Node B after receiving the request for *getdata*.

The message occurs in a *propagation delay* at each hop in the communication schema.

Definition 1. The *propagation delay* is described as the sum of the transmission time and the block or transaction's local verification time. *Transmission Time* consists of the time to send *inv* message, *getdata* message and the delivery time of the requested data.

As we have already seen, the size of the *inv* message is negligible and the *getdata* request also has a trivial influence on the transmission time. By contast block or transaction message could be very large, remember that a block has a maximum size of 1 MB and all the freshly mined block has a weight around this value. However, the procedure with the greatest impact on performance is the block (or transaction) verification performed before the announcement via the *inv* message. This is due to the fact that each transaction contained in the block is fully verified and a single verification involves random disk access which is the slowest element of a machine which bottlenecks the whole system.

Definition 2. Let B_h be the h height block set. A *block height* is defined as the distance between a block b and the origin (or genesis) block g, the first block in the blockchain hard-coded in the client of each node. Genesis block has a height of $h_g = 0$.

Definition 3. *Blockchain head* is described as the block with the maximum height, i.e. the last block mined in a node perspective, or formally the block with the highest distance from the origin block h_g.

Definition 4. In every node, there can be multiple heads at a time in any moment, formally this happens when $h > 1$. When this situation happens, we are in the presence of a *blockchain fork*. That implies the system's inconsistency.

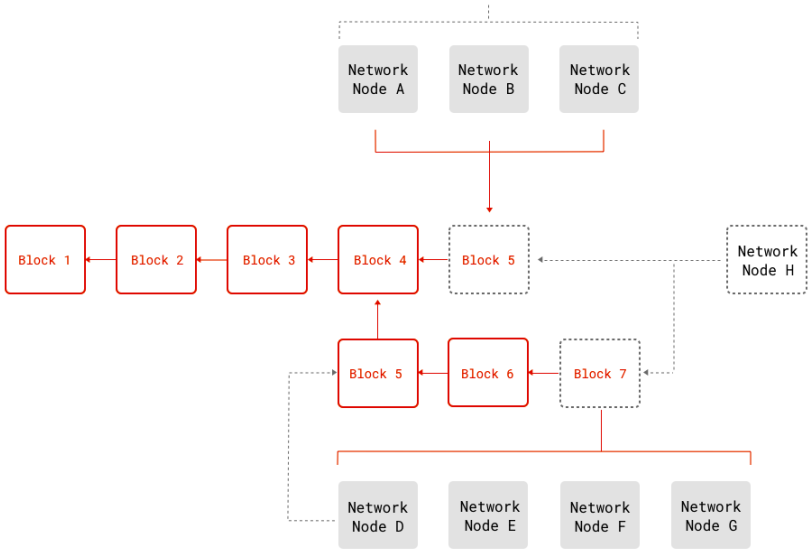

Figure 2.2: A blockchain fork

If a fresh mined block at height h' arrives at a node whose head is currently at height h with $h' > h$, the new block becomes the blockchain head for this node. If the old head, the block b_h, is in the same branch of the new one $b_{h'}$ it recovers all intermediate blocks in the branch and incrementally applies their changes. Vice versa, if block b_h is not an ancestor of $b_{h'}$, so it belongs to a different branch, they share a common ancestor. As $h' > h$, $b_{h'}$ must be in a longer chain than b_h it becomes the new head of the blockchain. From that node down to the common ancestor all modification are reversed applying the changes from the branch of $b_{h'}$.

Thanks to the partitions of the network a blockchain fork may widespread with the result of more blocks B_h+1, B_h+2 building on their respective different heads. The network will reach a point where one branch will ultimately be longer than the other branches. All the partitions with different branch adoption will switch over the new found branch. The fork is finally resolved and all ledger clones are consistent up to the same blockchain head.

Figure 2.2 illustrates the condition of a ledger fork. Due to network condition, nodes A, B and C have block 4 as their head while nodes D, E, F and G have already switched to blocks 7 as their new blockchain head since being the longest chain, composed by 7 blocks, it is considered the valid one. Once nodes

A, B and C are finally aware of the other branch with block 7, even if they have just mined block 5, they switched to the longest chain and discard block 5 that becomes an orphan node. When a fresh new node H joins the network t chooses the longest chain between the two available branch. This settled the fork and all the network peer agrees on the same blockchain head, block 7.

2.2.2 Data analysis on literature

Before analyzing the blockchain we propose some compelling works found in the literature.

The *Forschungsgruppe Dezentrale Systeme und Netzdienste*, namely the Research Group in Distributed Systems and Network Services, by the *Karlsruher Institut für Technologie* collected several blockchain data from July 2015 by adding a dummy node in the bitcoin network for research purposes only.

They have made their data and statistics available from their web platform [24] in form of plots and downloadable raw data.

With regard to the delay in the propagation of messages, we have both tthe transaction and the block propagation delay history, reported in the following plot for the last year:

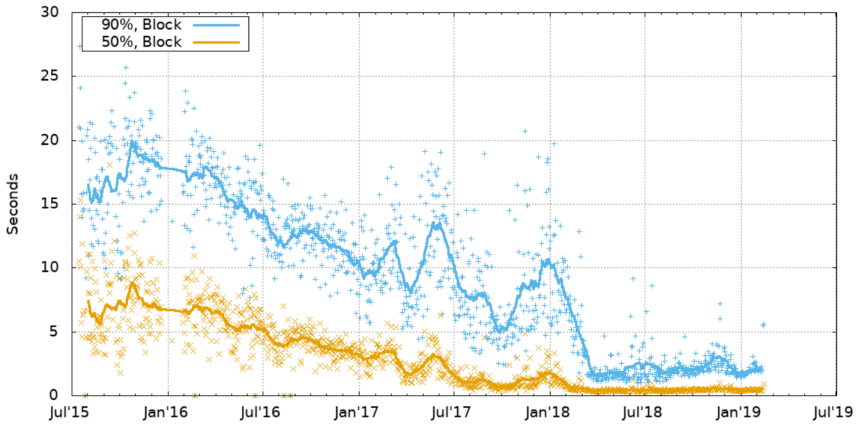

Figure 2.3: Block propagation delay history

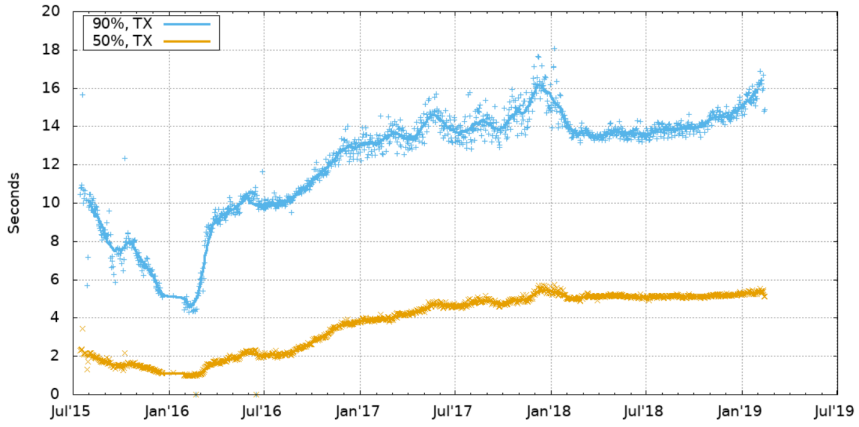

Figure 2.4: Transaction delay history

Both plots measure the time it takes until a specific block or transaction is announced by 50% and 90% of the network nodes. As we can see (for this work we only take into account data from the last year) in the first months of 2018 block propagation delay fell from 10 seconds to about 2-3 seconds. The delay fall in the same period is less noticeable on transactions passing from a propagation delay of 16-15 to 14-13.

Another aspect affected by the fall in early 2018 was the *inv* per hour as shown in the figure below.

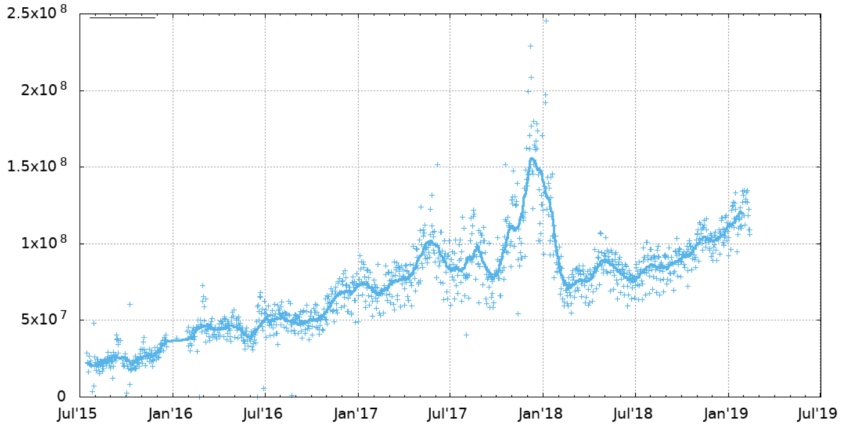

Figure 2.5: Transaction delay history

The plot above illustrates the total amount of *inv* messages received by network peers per hour, announcing the availability of a fresh set of blocks and transactions to be transferred. In this case, the value (in the same time interval) dropped from around $1.5 \times 10^7$to 7×10^7,a difference of a magnitude order.

In 2013, Decker and Wattenhofer [23] implemented the bitcoin network protocol and linked a large sample of network nodes to evaluate delay in the propagation of messages. Their implementation acts precisely like a standard node with one caveat: it does not relay *inv* messages, transactions or blocks

They measured the impact of message size in terms of delay cost on the propagation delay, as shown in Figure 2.6.

Definition 5. The *delay cost* is defined as the time delay each KB causes to the diffusion of message, i.e. a block or transaction.

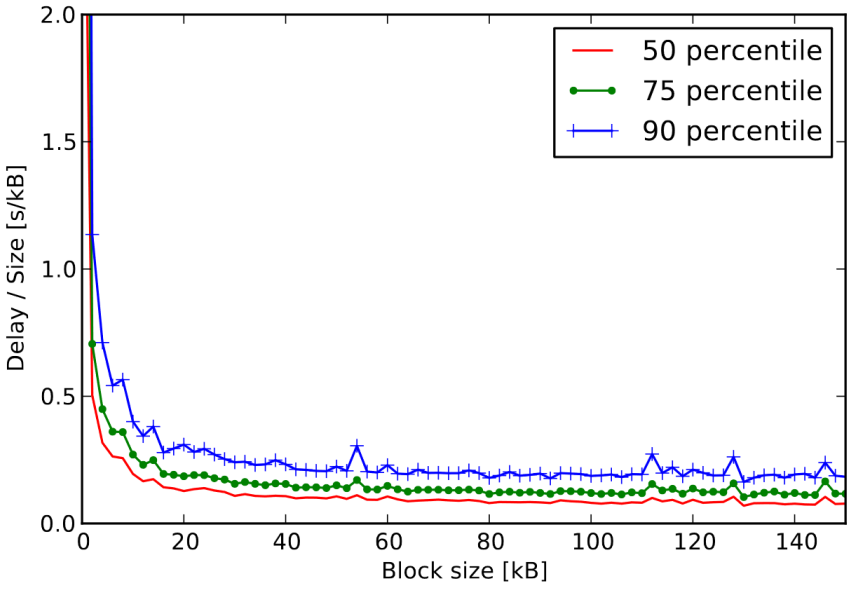

Figure 2.6: Plot of the message size versus the delay cost

This plot illustrates delay price for 50, 75 and 90 percentile for different message size. The clear result is that for messages with a size of less than 20 KB, the significant overhead is due to the round trip delay time or RTT (the time it takes to send a message plus the length of time it takes to obtain

the acknowledgement signal) caused in this situation by announcing fresh data via the *inv* message and replying to the *getdata* message. Since 96% of total transactions (when the paper was written in 2013) were smaller than 20 KB, the dominant factor for this message was RTT. For sizes greater than 20KB the cost is constant and each KB of data added supplementary 80ms of delay to reach the mass of the peers.

In their research, they propose a direct connection between propagation delay and blockchain forks, and in order to decrease them, they have proposed three modifications to the Bitcoin protocol: minimizing verification (that as already stated involves disk access), pipelining block propagation and enhancing network connectivity. They study the effect of any of these three suggestions, and the combination of them all, using the application adopted before measuring the resulting delay time. As a consequence, they were able to decrease the amount of forks by more than 50%.

Nonetheless they came to the conclusion that [23]:

> *[...]The root cause of the problem however is intrinsic to the way information is propagated in the network. The changes may mitigate the problem in the short term, until a scalable long term solution is found.[...]*

2.3 Blockchain analysis

2.3.1 Methodology

The starting point was to download the entire blockchain to evaluate some fascinating elements of the bitcoin system. The first issue we faced was the size of the ledger, which being actually over 200 GB could take a long time to download due to the restriction of our home internet connection and most importantly hard to evaluate and parse given the computing power and resource availability of our home computer.

To fix this issue, we were able to use one of the University servers: *daffyduck*. A machine with 32 cores CPU with 96 GB of memory and a large storage capability. To download the block files we used the classic Bitcoin core launcher which automatically downloads the entire blockchain from the network. It took about a day for the whole process to finish. The download started with the older and lighter blocks and continued to download the newer and heavier blocks to

keep the chain in sync with the network. Since the blockchain is a continually growing structure, we choose to stop downloading at 10:45:29 AM on October 11, 2018.

Unfortunately, considering the nature of the chain, the files are not readable immediately, in fact all the information are stored in raw files to enhance the space occupancy of the entire structure. Indeed, the solution was to use a parser.

Therefore, the next step was to parse the chain to retrieve the pivotal block and transaction information. For the purpose of the study conducted in this dissertation, we considered data from January 1, 2013 which coincide with block 214554 to October 11, 2018, block no. 545286, date when we stopped downloading.

The effective parse was done using a simple python script using the parser library *"python-bitcoin-blockchain-parser"* [25] available on Github. (For more information on the library and the script used to parse the chain see *Appendix A*).

We organized the outcome of block files parsing in a *csv* file to enhance human readability.

2.3.2 Block size analysis

Once we get block size data by parsing the blockchain, we count the occurrences for each block size and group them into classes. We plot the data classes along with their occurrence probability to get a glimpse of the data distribution.

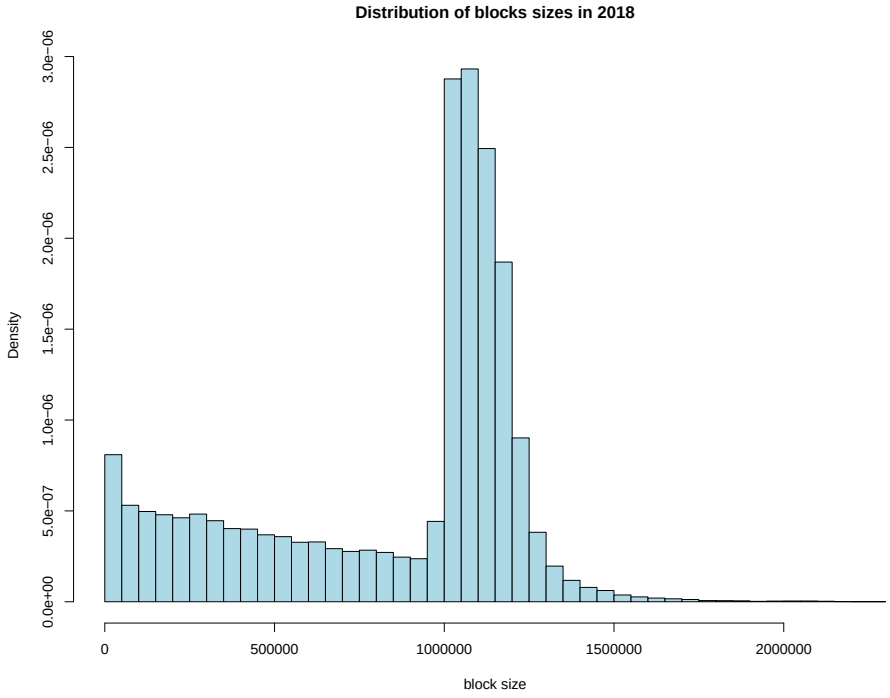

Figure 2.7: Plot of blocks size

It is evident from the plots that the vast majority of blocks are around one million bytes large. The distribution mean is 550700.8 while the standard deviation is 409576.7.

2.3.3 Empty block gaps distribution analysis

In this section, we focus on evaluating the time gaps between empty blocks, identified as the amount of non-empty blocks between two empty blocks.

First, we plotted the trend of empty block gaps over time using the R library *Plottly*, given the period between January 2013 and October 2018.

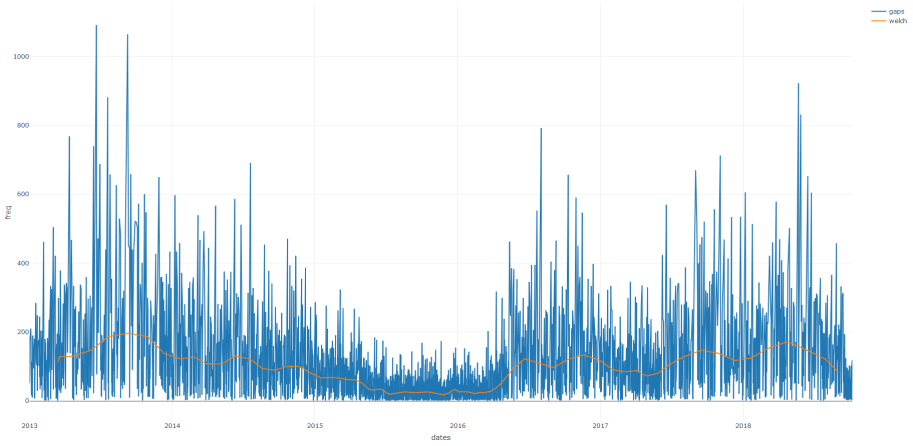

Figure 2.8: Empty blocks gaps over time

To look at data distribution, we plotted the dataset histogram.

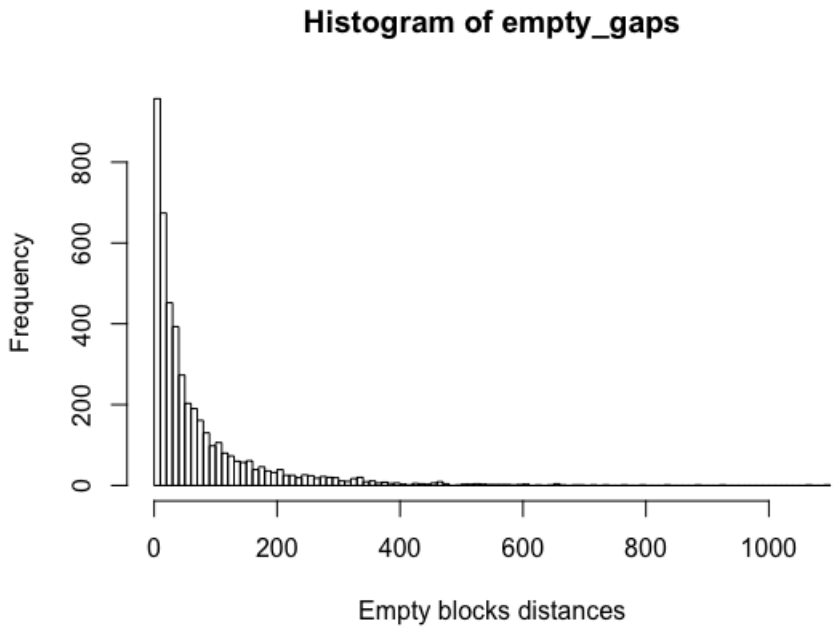

Figure 2.9: Plot of blocks size distribution

As we saw in previous section (2.2), availability of new blocks is announced via the *inv* message and the real data transmission takes place after receiving the *getdata* message. We assume the empty blocks to be generated in the time period between the announcement of the *inv* message and the actual blocks transmission (as highlighted in Figure 2.10). This could suggest that empty blocks are geometrically distributed.

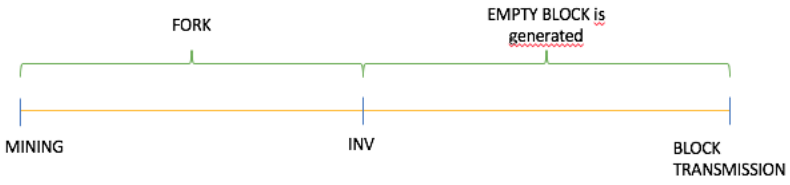

Figure 2.10: Schema of the generation of empty blocks

At a first glance Figure 2.9 seems to strengthen our guess. The best way to demonstrate it is to use a hypothesis test to evaluate the fit's goodness. There are plenty of hypothesis test available in the literature, our decision fell back on the Chi square test as it's one of the simplest and easiest to achieve.

2.4 Hypothesis test

2.4.1 Theoretical background

The chi square test for goodness of fit is a commonly used hypothesis test to measure how an observed pool of data/data set is consistent with a given fitted distribution.

As every hypothesis test it requires to state the null hypothesis H_0 and the alternative hypothesis H_1, in a way that are mutually exclusive.

Since the test is used to measure the goodness of the fit the two hypotheses it use the following structure:

- H_0: data are consistent with a certain distribution
- H_1: data are **not** consistent with a certain distribution

The significance level, denoted as alpha or α, define the probability of rejecting the null hypothesis considering it true. Usually, this value is set as 0.05 or 0.01.

With n observations classified into k mutually exclusive classes with x_i ($i = 1, 2, ..., k$) observed values and the probability p_i that an observation is contained in the *i-th* class, given by the null hypothesis.

The value:

$$X^2 = \sum_{i=1}^{k} \frac{(x_i - m_i)^2}{m_i}$$

Where $m_i = np_i$ and $\sum_{i=1}^{k} p_i = 1$

and $\sum_{i=0}^{k} m_i = n \sum_{i=0}^{k} p_i = \sum_{i=0}^{k} x_i$

When n becomes large, X^2 follows the χ^2 distribution with $k - 1$ degrees of freedom. Given the χ^2 value above and the degree of freedom is it possible to calculate the *p-value*.

p-value represents the probability of observing a sample statistic equal or more extreme than the test statistic , asserting the null hypothesis as true. The *p-value* is then compared with the *significance level*, rejecting the null hypothesis when the *p-value* is less than the *significance level*.

2.4.2 Geometric distribution

The geometric distribution is informally defined as the distribution of the number of heads to be flipped before the first using a weighted coin.

There are two distinct formal ways to define a geometric distribution:

Definition 6. The geometric distribution represents the number of Bernoulli trials before getting a success in a series of Bernoulli trials.

Definition 7. The geometric distribution represents the probability distribution of the number of failures before the first success.

Clearly, these two definition are equivalent and the choice between one or the other definition is a matter of convenience, context and convention.

Definition 8. A Bernoulli trial, named after Swiss mathematician Jacob Bernoulli who introduced it in 1713 [26], is defined as an experiment with two possible outcome: "success" and "failure". Moreover, in a Bernoulli trial the probability of success is the same for every repeated experiment.

Concerning the first definition, given a success probability p and x independent Bernoulli trial until the first success, the probability mass function (PMF) is given by :

$$Pr(X = x) = (1-p)^{x-1}p \quad \text{for } x = 0, 1, 2, 3, ... \tag{2.1}$$

The Cumulative density function (CDF) is:

$$1 - (1-p)^{x} \quad \text{for } x = 0, 1, 2, 3, ... \tag{2.2}$$

The expected value and the variance are represented by:

$$E(X) = \frac{1}{p} \tag{2.3}$$

$$Var(X) = \frac{1-p}{p^2} \tag{2.4}$$

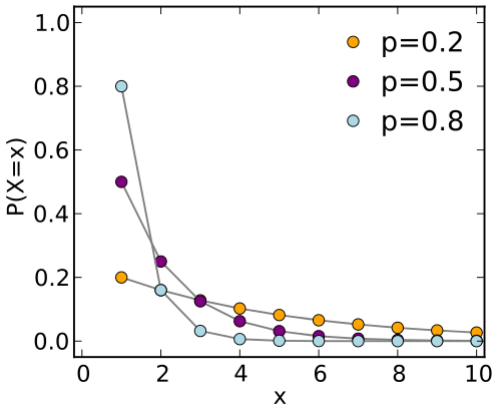

Figure 2.11: Probability mass function for first type geometric distribution

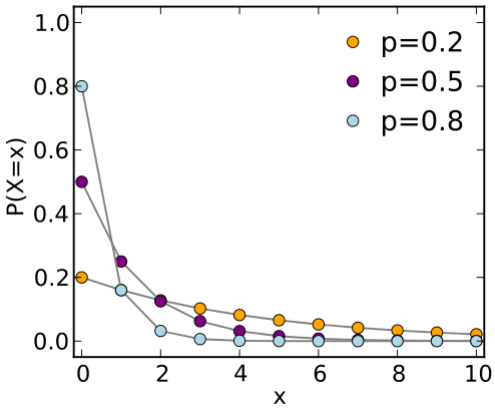

Figure 2.12: Cumulative density function for first type geometric distribution

Considering the second definition, the probability mass function is :

$$Pr(X = x) = (1-p)^x p \quad \text{for } x = 0, 1, 2, 3, ... \tag{2.5}$$

with p as the success probability and x number of failures until the first success.

The Cumulative density function (CDF) is:

$$1 - (1-p)^{x+1} \quad \text{for } x = 0, 1, 2, 3, ... \tag{2.6}$$

The expected value and the variance are given by:

$$E(X) = \frac{1-1}{p} \tag{2.7}$$

$$Var(X) = \frac{1-p}{p^2} \tag{2.8}$$

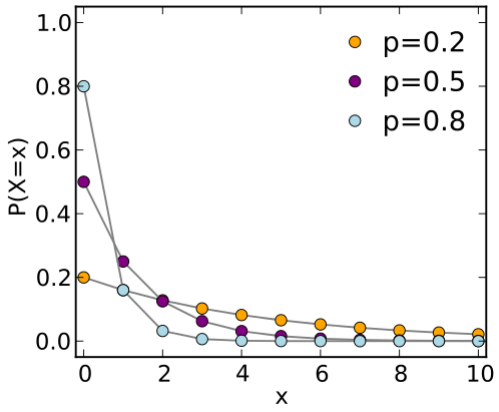

Figure 2.13: Probability mass function for second type geometric distribution

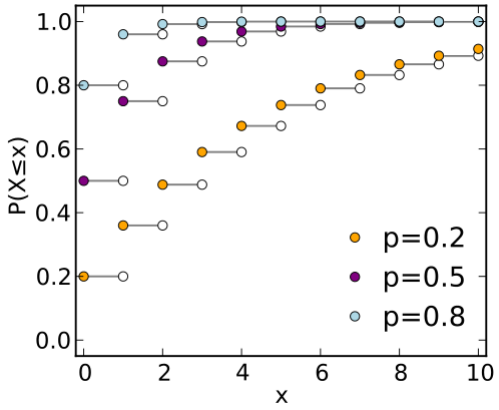

Figure 2.14: Cumulative density function for second type geometric distribution

Note that the geometric distribution is the only discrete distribution (like its continuous counterpart the exponential distribution) that satisfies the *memoryless property*.

This property states that the probability distribution of the number of additional failures does not depend on the number of failures already observed if

a success has not yet occurred at a given point.

Definition 9. A discrete random variable X is *memoryless* w.r.t a variable a if:

$$P(X > a + b | x > a) = P(x > b) \quad \text{for } a, b \in \mathbb{Z}^+ \tag{2.9}$$

Definition 10. A continuous random variable X holds the *memoryless* property if:

$$P(X > z + k | x > z) = P(x > k) \quad \text{for } z, k \in \mathbb{R} \tag{2.10}$$

For instance, let us suppose to toss a fair coin until head comes up. If we know that the coin had already been tossed ten times without observing a head, the probability distribution of the number of further toss until a head is the same as it would be without knowing about the ten failure attempts.

2.4.3 Experimental result

Given the problem settings, for this test the hypothesis are:

- H_0: data are consistent with a geometric distribution
- H_1: data are **not** consistent with a geometric distribution

The *sensibility level*, denoted as **alpha** is set at **0.05**.

To perform the test we used R software which thanks to functions $fitdist$ and $gofstat$ included in the library $fitdistrplus$ allows us to achieve the test results in a fast and easy way. The $fitdist$ function returns, by applying the *Maximum Likelihood Estimation* [27] to the data, the parameters of the selected distribution i.e. in this case the geometric distribution.

The following plot shows the results of the fitting of our data with the geometric distribution compared with the theoretical distribution and CDF.

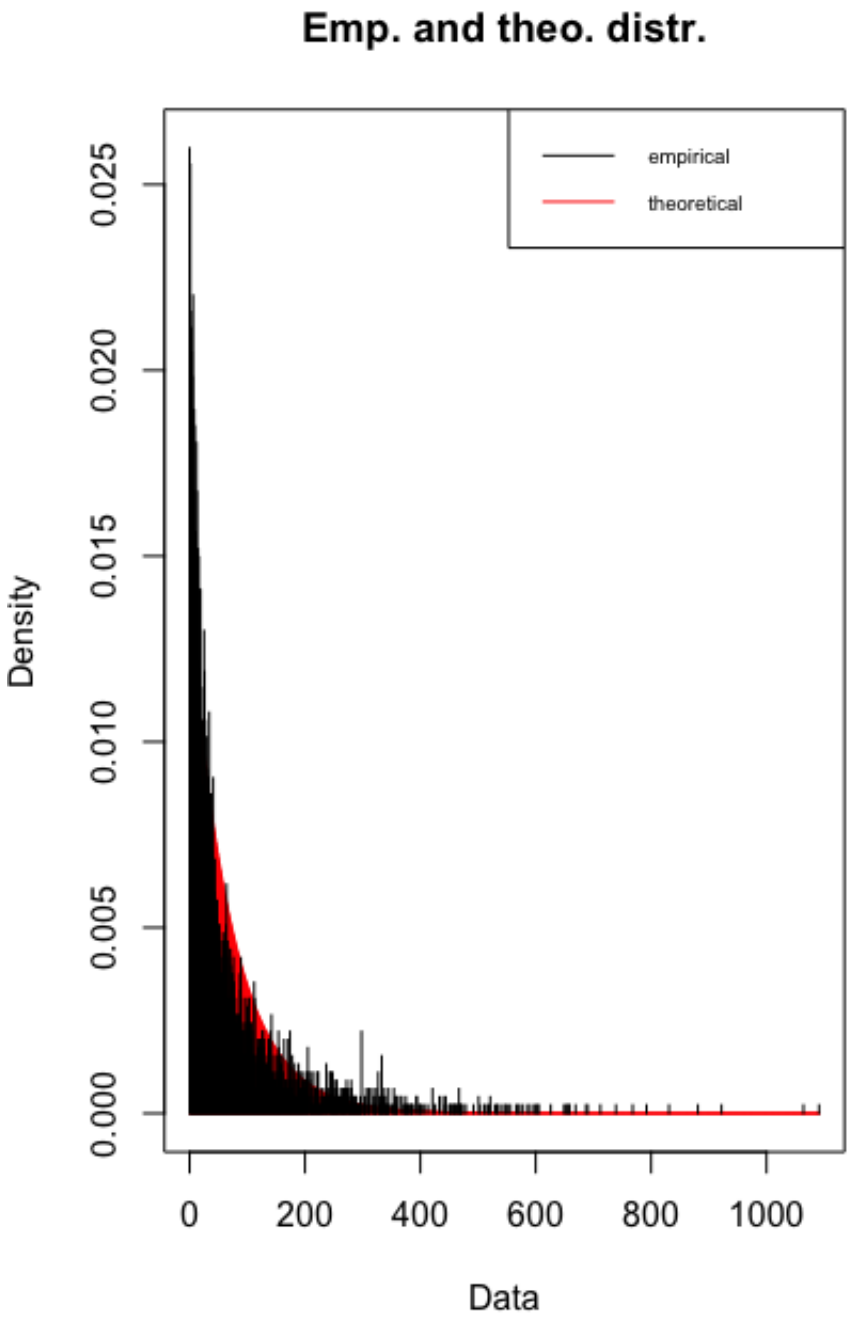

Figure 2.15: Plot of observed distribution vs theoretical geometric distribution

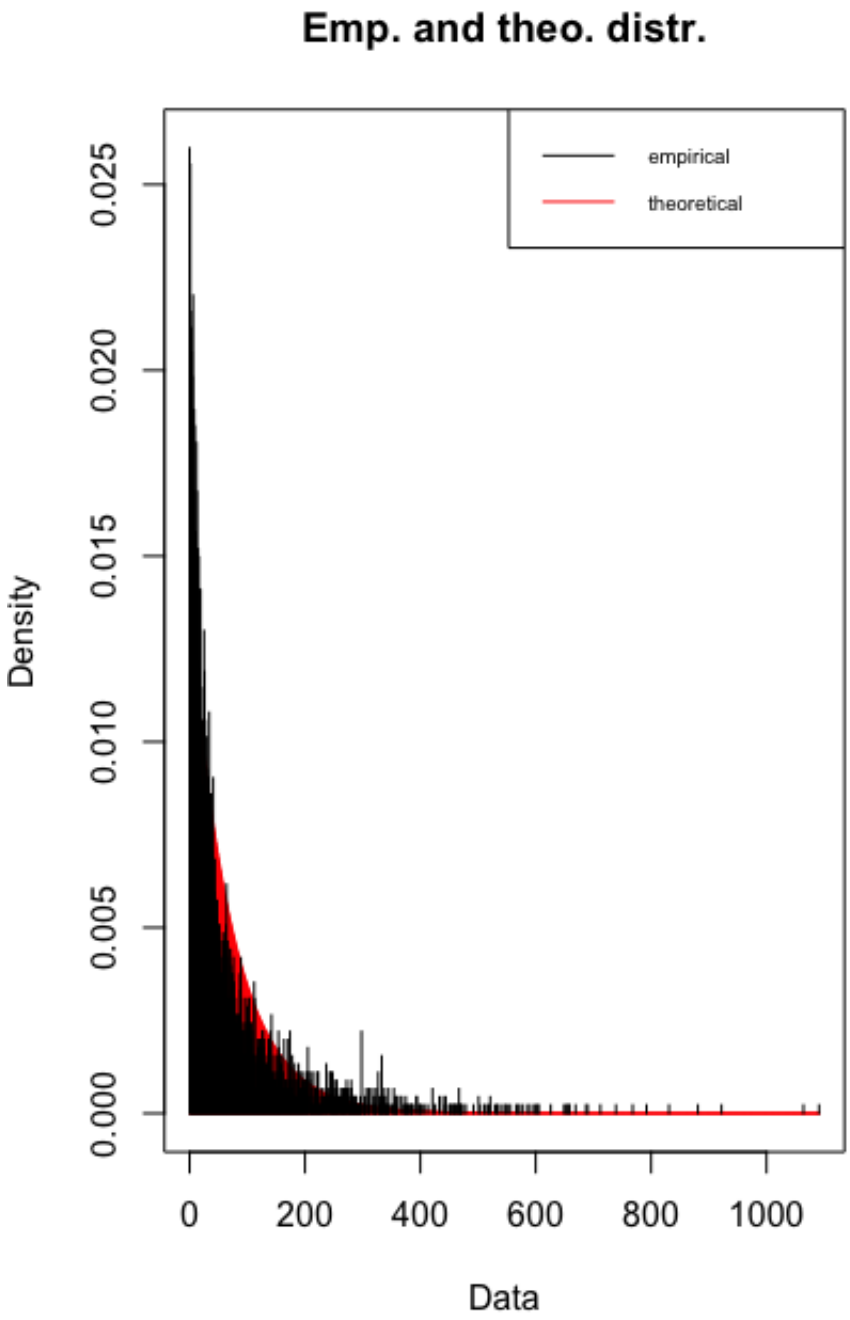

Figure 2.16: Plot of observed CDF vs theoretical geometric CDF

The real hypotesis test is performed by the $gofstat$ function which takes as input the outcomes of the previous function applies the test and returns some fundamental results like the p-value, the Chi square value along with its degrees of freedom.

The results are the following :

```
Chi-squared statistic:  929.1133
Degree of freedom of the Chi-squared distribution:  41
Chi-squared p-value:  3.455772e-168
Chi-squared table:
        obscounts theocounts
<= 0   118.000000  62.474680
<= 1   116.000000  61.615348
<= 2    98.000000  60.767836
<= 3    96.000000  59.931982
<= 4    93.000000  59.107624
<= 5    90.000000  58.294606
<= 6   100.000000  57.492770
<= 8   162.000000 112.623999
<= 10  155.000000 109.547047
<= 12  153.000000 106.554159
<= 14  131.000000 103.643039
<= 16  137.000000 100.811452
<= 18  118.000000  98.057226
<= 20  116.000000  95.378246
<= 23  127.000000 138.203841
<= 25   97.000000  88.996651
<= 27  102.000000  86.565211
<= 30  120.000000 125.433683
<= 33  124.000000 120.328575
<= 36  119.000000 115.431242
<= 39  104.000000 110.733230
<= 42   99.000000 106.226425
<= 46  104.000000 134.940598
<= 50   93.000000 127.668011
<= 55   99.000000 149.953029
<= 60   98.000000 139.919928
```

```
<= 65  107.000000 130.558124
<= 71  100.000000 145.191132
<= 77  102.000000 133.613175
<= 85   92.000000 161.725784
<= 92   93.000000 127.532758
<= 102 102.000000 162.021283
<= 112 102.000000 141.065472
<= 125  91.000000 156.469413
<= 139  92.000000 139.798817
<= 155  94.000000 129.860347
<= 174  92.000000 121.118215
<= 201  91.000000 125.529780
<= 233  90.000000  99.107770
<= 272  90.000000  74.165741
<= 330  90.000000  57.171684
<= 465  90.000000  39.221902
> 465   55.000000   7.148166

Goodness-of-fit criteria
                                1-mle-geom
Akaike's Information Criterion   47960.54
Bayesian Information Criterion   47966.96
```

The outcome *p-value* is $3.455772e^{-}16$. Having set the sensitivity level at 0.05, since it's lower than our alpha we have to **reject** the Null Hypothesis. This means that the data **does not distribute geometrically**.

Hence, we decided to change our interval since, as the features of the bitcoin blockchain have changed drastically during its lifetime, the empty block generation process is non-homogeneous. For instance, if we think about the early days when the system started, most mined blocks were empty only because the pending transactions list was so. For this reason we reformulate the test by selecting data from a narrowed period of time, considering only data from January to October 2018.

We re-applied the $fitdist$ R function to the new data-set having the following findings.

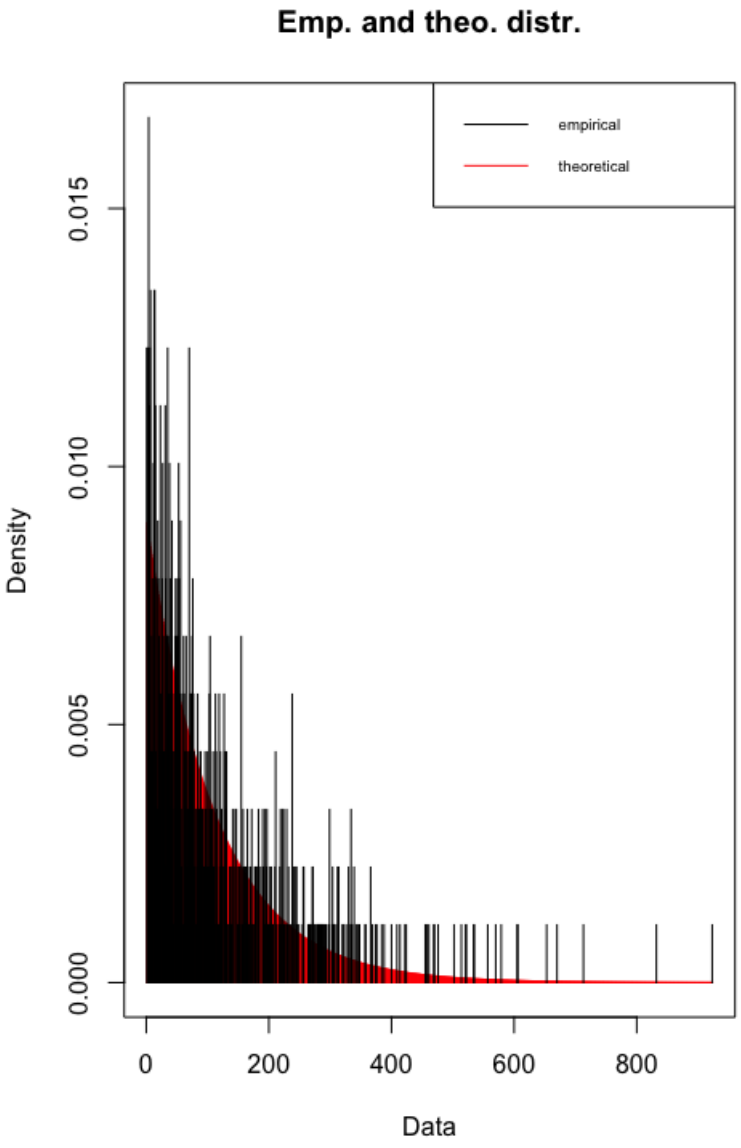

Figure 2.17: Plot of observed distribution vs theoretical geometric distribution of new data

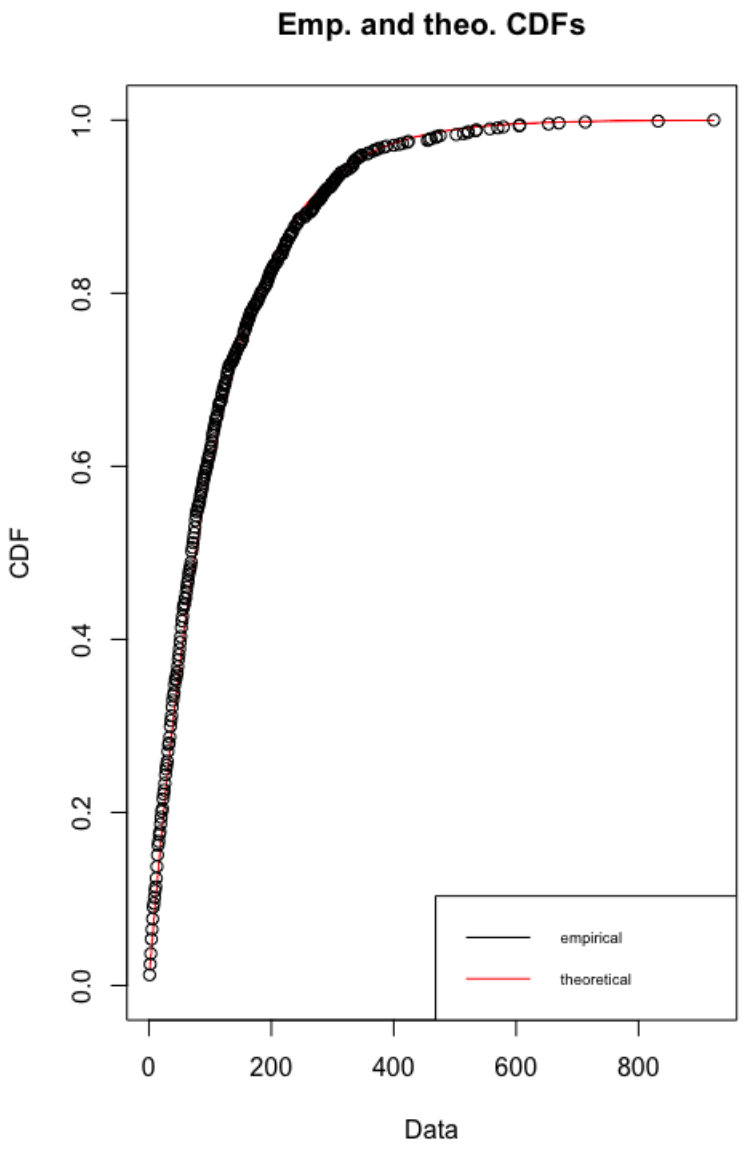

Figure 2.18: Plot of observed CDF vs theoretical geometric CDF of new data

By applying *gofstat* function we achieved:

```
Chi-squared statistic: 23.45346
Degree of freedom of the Chi-squared distribution: 23
Chi-squared p-value: 0.4345676
```

Finally given the test outcome we can say that, since our alpha is 0,05, the p-value is greater that the significance level. Which leads us to the conclusion that data considered in the shrank interval **does distributes geometrically**.

Since data are geometrically distributed their probability distribution satisfies the *memoryless* property meaning that the probability of observing an empty block, knowing that the previous n blocks were not empty, is the same as it would be if we didn't know anything about past blocks.

2.5 Chapter conclusion

By concluding this chapter, we have seen how the issue of the rise of the block size limit is still full of discussion within the Bictoin community and how this seems inevitable for the future. We then deepened the way messages are propagated in the Bitcoin network. We have seen how bitcoin forks are generated and from the literature we have seen how this issue has been resolved. With data from the University of Karlsruhe, we were able to notice that something that happened in early 2018 caused block propagation delay and, to a lesser extent, transactions to drop dramatically.

A further analysis on the drawbacks of enlarging block size limit is made in Chapter 4.

We finally evaluated the distribution of empty blocks gaps. Using the Chi Square hypothesis test, we attempted to fit data with a geometric distribution. After an initial failure with data considered in a long span between January 2013 and October 2018, we were only able to fit data from 2018 by narrowing the interval since the process of generating blocks is not homogeneous as the characteristics of the ledger have changed over time.

The large accomplishment we get from demonstrating that data are geometrically distributed is provided by the validity of the memoryless property.

Chapter 3

Queuing model for the blockchain

Chapter introduction

This chapter is centralized on the proposal of a queuing model for the Bitcoin system in a case of an infinite transaction queue.

We will initially suggest a basic introduction to the queuing theory: how queuing systems are denoted, what is a Poisson process and which are the key fundamental queuing models. We then suggest our model by providing it a thorough assessment of the outcomes.

3.1 Queuing theory background

3.1.1 Kendall's notation

To formally describe a queuing model *David G. Kendall* in 1952 [28] proposes a set of notation that takes the name of *Kendall's Notation*. In the initial script his notation used three factors generally indicated with $A/S/c$, namely: A is the arrival process, S the service time distribution and c the number of servers.

After being assumed as a standard in the literature, his notation was expanded taking the actual form consisting of six factors:

$$A/S/c/K/N/D \tag{3.1}$$

where, along with A,S and c, have been added: K denoting the maximum capacity of the system, N as the size of the population and D referring to the queuing discipline.

Each of the factor can be expressed by various letters according to the significance of the factor. The arrival process A can assume letters:

- M : Markovian, for Poisson arrival process
- G : General, for general distribution
- D : Degenerate, for deterministic inter-arrival time

Service time S can be indicated with:

- M : Markovian, for Exponential service time process
- G : General, for general service time distribution
- D : Degenerate, for deterministic service time

Servers granularity c, maximum system capacity K and the magnitude N of the population of which the customer is a part are expressed with a number. S can indeed assume the value infinity. When K and N are not indicated they are intended as infinite.

Finally the queuing discipline D can be marked as:

- $FIFO/FCFS$: First In First Out/ First Come First Served
- $LIFO/LCFS$: Last In First Out/ Last Come First Served
- PS : Processor Sharing

If not specified the queuing discipline is implied to be FIFO/FCFS.

To sum up, for instance a M/M/3/20/1500/FCFS system has: Poisson arrival process, exponential service time, 3 servers, a maximum number of jobs in the system of 20, the population size is 1500 and the queuing discipline is FCFS.

3.1.2 Little's Law

John D.C. Little, professor at the Massachusetts Institute of Technology (MIT), published in 1952 the so-called *Little's Law*, a theorem that allows us to relate

the average number of customers in the queue to the product of the average arrival rate and the average response times of the queuing system.

However, in a second publication in 1961, he demonstrated that the relationship he developed holds in any scenario.

The most remarkable accomplishment of this theorem is that neither the service distribution nor the arrival process distribution influences the relationship, not even the by the service order. The only assumption is that the system must be stable ($\lambda < \mu$) and non-preemptive.

The law is represented by the underlying formula:

$$N = X * R \tag{3.2}$$

Let N be the average number of customer in the system.

Let X be the average effective arrival rate that in the case of a Poisson process, is equal to the parameter λ

Let R be the average response time of the system.

3.1.3 Poisson process

Poisson process is a specific type of continuous time counting process. It is widely used in the fields of computer and telecommunications system performance evaluation to model the arrival process of jobs (or customers) in a service infrastructure. It's named after French mathematician Siméon Denis Poisson.

A counting process $X(t)$ satisfies the following properties:

1. $X(t) \in \mathbb{N} \quad t \in T$

   where T denotes the time domain.

2. $X(s) \leq X(t) \quad$ if $s \leq t$

A counting process can be seen as a peculiar stochastic process that counts the number of events observed in the interval $(0, t]$. Let t_1 and t_2 be two time instants such that $t_1 < t_2$. Hence, according to the properties pointed out above, the number of events ensued in the time interval (t_1, t_2) is given by $X(t_2) - X(t_1)$.

If the number of events occurring in the disjoint time interval are independent, the counting process is said to have *independent increments*, while if the distribution of events, occurred at any time interval, depends only on the interval length, the counting process has *stationary increments*.

Hereafter, we propose two distinct definition of a Poisson process:

Definition 11. A Poisson process $X(t)$ with rate (or intensity) $\lambda \in \mathbb{R}^+$ is a continuous time counting process $t \in \mathbb{R}$ that satisfies the following:

1. $X(0) = 0$
2. for any pair of disjoint intervals $(t_1, t_2]$ and $(t_3, t_4]$ the increments $X(t_2) - X(t_1)$ and $X(t_4) - X(t_3)$ are independent random variables
3. for any $t, s \geq 0$ the increment in the disjoint interval $(t, t+s]$ has a Poisson distribution with mean λs:

$$Pr\{X(t+s) - X(t) = k\} = \frac{(\lambda s)^k e^{-\lambda s}}{k!} \tag{3.3}$$

The second proposed definition is the following:

Definition 12. A Poisson process X(t) with rate (or intensity) $\lambda \in \mathbb{R}^+$ is a continuous time counting process that satisfies the following properties:

1. X(0) = 0
2. the process is stationary and has independent increments
3. $Pr\{X(h) = 1\} = \lambda h + o(h)$
4. $Pr\{X(h) \geq 2\} = o(h)$

where we say that a function f is $o(h)$ if:

$$\lim_{h \to 0} \frac{f(h)}{h} = 0 \tag{3.4}$$

It is proved that *Definition 1* and *Definition 2* are equivalent

A Poisson process in addition satisfies some useful properties:

1. **Superposition of Poisson processes**:

   Given two independent Poisson processes $X_1(t)$ and$X_2(t)$ with rate respectively of λ_1 and λ_2. Hence the superposition of the two Poisson processes $X(t) = X_1(t) + X_2(t)$ is still a Poisson process with rate $\lambda_1 + \lambda_2$.

2. **PASTA property**:

   PASTA stands for Poisson Arrivals See Time Averages, it states that customers arriving in the system characterized by a Poisson process see the state of the system as if they had entered in any instant of time.

3. **Random selection**:

   Let $X(t)$ be a Poisson process with intensity λ. If we made a random selection from $X(t)$ in a way that each arrival is selected independently with probability p, the resulting process is still a Poisson process with intensity λp

4. **Inter-arrival times distribution**:

   Let $x(t)$ be a Poisson process with rate λ, the customer inter-arrival times are independent and exponentially distributed with rate λ distributes

3.1.4 M/M/1 queue

M/M/1 queue is one of the fundamental queuing models. By recalling chapter 3.1.1, it represents a system where customers arrives according to a Poisson arrival process with rate λ, service times are exponentially distributed with rate μ and with a single server that serves customer in FIFO/FCFS order, meaning that they are served following the order of arrival. Since the buffer size is not explicitly indicated, it is intended to be infinite.

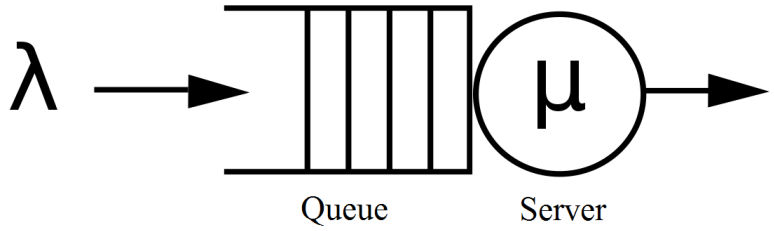

Figure 3.1: An M/M/1 queue

The underlying Continuous Time Markov Chain in which the system can be described has irreducible and infinite state space. Its space diagram is depicted in the figure below.

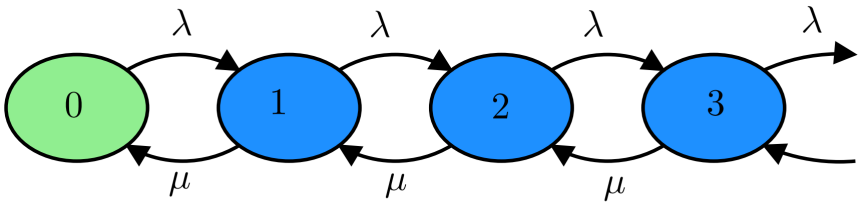

Figure 3.2: M/M/1 queue correspondent CTMC

This chain is the same as in a birth-death process.

Stability of the system is ensured when $\lambda < \mu$ otherwise if customers arrives faster than the service capacity, the queue will grow endlessly long. As a consequence to instability the system loses its stationary distribution.

Let ρ be the ratio λ/μ. This is called the *load factor* of the system. If the load factor $\rho < 1$, that is equal to say that $\lambda < \mu$, the system is stable.

The following system describes the Global Balance Equations for M/M/1 queue are:

$$\begin{cases} \pi(0)\lambda = \pi(1)\mu \\ \pi(n)(\lambda+\mu) = \pi(n-1)\lambda + \pi(n+1)\mu \quad n > 0 \end{cases} \tag{3.5}$$

By fixing $\pi(0)$ we have that :

$$\pi(n) = \pi(0)(\frac{\lambda}{\mu})^n \quad n \geq 0 \tag{3.6}$$

By imposing $\sum_{n=0}^{\infty} \pi(n) = 1$

$$\sum_{n=0}^{\infty} \pi(0)\rho^n = \pi(0) \sum_{n=0}^{\infty} \rho^n = \pi(0)\frac{1}{1-\rho} \tag{3.7}$$

The geometric series converges *iff* $\rho < 1$ so when the system is stable.

Computing $\pi(0)$:

$$\pi(0)\frac{1}{1-\rho} = 1 \Rightarrow \pi(0) = (1-\rho) \tag{3.8}$$

Various performance indices can be computed for this queuing model :

- **Utilization**

  The steady-state probability of finding the server busy is :

$$U = 1 - \pi(0) = \rho \tag{3.9}$$

- **Expected number of customers in the system** :

  We have that

$$\bar{N} = E[n] = \sum_{n=0}^{\infty} n\pi(n) = \sum_{n=0}^{\infty} n\pi(0)\rho^n \tag{3.10}$$

  So :

$$\begin{aligned} \bar{N} = \sum_{i=0}^{\infty} i\pi(i) = (1-\rho)\sum_{i=0}^{\infty} i\rho^i = \\ (1-\rho)\rho\sum_{i=0}^{\infty}(i+1)\rho^i = \\ (1-\rho)\rho\sum_{i=0}^{\infty}\frac{\partial\rho^(i+1)}{\partial\rho} = \\ (1-\rho)\rho\frac{\partial}{\partial\rho}\sum_{i=0}^{\infty}\rho^(i+1) = \\ (1-\rho)\rho\frac{\partial}{\partial\rho}\frac{\rho}{1-\rho} = \\ (1-\rho)\rho\frac{1-\rho+\rho}{(1-\rho)^2} = \frac{\rho}{1-\rho} \end{aligned} \tag{3.11}$$

  That considering $\rho = \lambda/\mu$ is correspondent to:

$$\frac{\lambda}{\mu - \lambda} \tag{3.12}$$

- **Expected number of customers in server** :

  Knowing that the expected number of customers in the service room is $E[n_s] = \sum_{i=1}^{\infty} 1\pi(i) = 1 - \pi(0) = \rho$.

  By recalling that $E[N] = E[n_q] = E[n_s]$ which straightforwardly means that the average number of customers in the whole system is given by the combination of average number of customers in the service room and average number of customers in the queue:

$$E[n_q] = \frac{\rho}{1-\rho} - \rho = \frac{\rho^2}{1-\rho} \tag{3.13}$$

- **Expected response time** :

  Since the system is in stability the throughput X = λ.

  By Little's Theorem we can compute the expected response time:

$$\bar{R} = E[r] = \frac{\bar{N}}{X} = \frac{1}{\mu - \lambda} \tag{3.14}$$

- **Expected waiting time** :

  Since the expected service time $E[s]$ is $1/\mu$

$$E[w] = E[R] - E[s] = \frac{1}{\mu - \lambda} - \frac{1}{\mu} = \frac{1}{\mu(\mu - \lambda)} \tag{3.15}$$

A real world example of an M/M/1 queuing system is a big telephone exchange. In this case the number of customers in the system is vast, all the customers are behaves independently from each other and a single user has a small impact on the system performance. So the telephone calls arrival are Poisson distributed. Assuming that there is only one exchange, our system has a single server.

Service times are also generated by a Poisson process since total number of customer served by the system is huge, performance impact on the system of a single call is negligible and users decide independently for how long they make their call last.

3.1.5 M/M/m queue

M/M/m queue can be seen as a generalization of an M/M/1 queue. The arrival process is Poisson process, service time is exponentially distributed and differently from the previous system we have m independent servers.

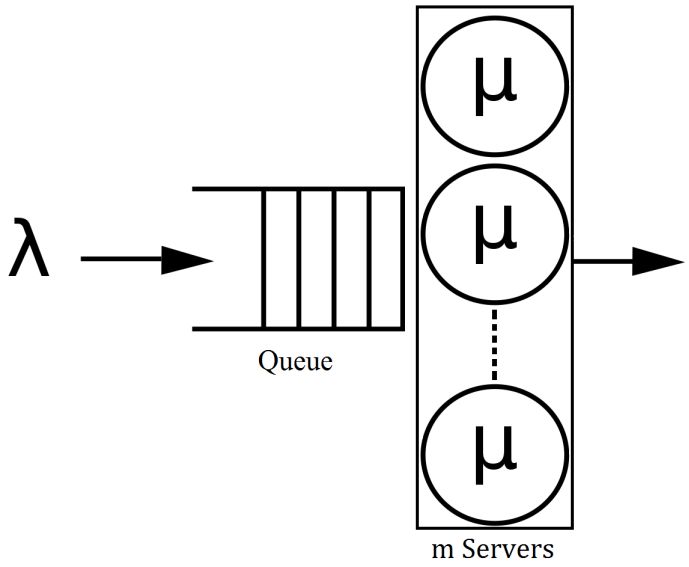

Figure 3.3: An M/M/m queue

As $\lambda(n) = \lambda$ and :

$$\mu(n) = \begin{cases} n\mu & if \;\; n \leq m \\ m\mu & otherwise \end{cases} \tag{3.16}$$

the steady-state distribution for this queuing system is:

$$\pi(n) = \begin{cases} \pi(0)\frac{1}{n!}(\frac{\lambda}{\mu})^n & if \;\; n \leq m \\ \pi(0)\frac{1}{m!}(\frac{\lambda}{\mu})^n \frac{1}{m^{n-m}} & if \;\; n > m \end{cases} \tag{3.17}$$

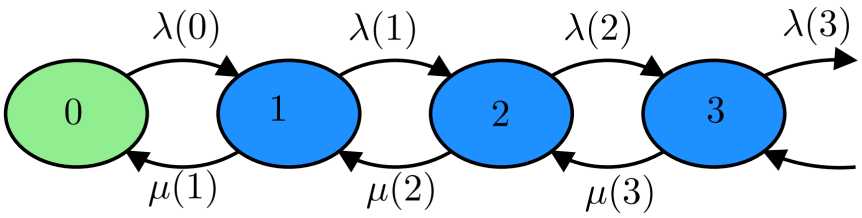

Figure 3.4: The CTMC associated to an M/M/m queue

The stability condition of the system is $\lambda < m\mu$ that is the same condition for ergodicity. $\pi(o)$ is computed as

$$\pi(0) = \left(\sum_{j=0}^{m-1} \left(\frac{\lambda}{\mu}\right)^j \frac{1}{j!} + \left(\frac{\lambda}{\mu}\right)^m \frac{1}{m!} \frac{1}{1 - \frac{\lambda}{m\mu}} \right)^{-1} \tag{3.18}$$

Like in the specific M/M/1 queue case we can express various performance indices for this queuing model :

- **Probability of finding all the servers busy**: This probability is given by the Erlang-C formula proposed by one of the father of queuing theory the Danish mathematician, statistician and engineer *Agner Krarup Erlang.*(inserire se si tova paper erlang)

$$C\Big(m, \frac{\lambda}{\mu}\Big) = \frac{1}{1 + (1-\rho)(\frac{m!}{(m\rho)^m}) \sum_{k=0}^{m-1} \frac{(m\rho)^k}{k!}} \tag{3.19}$$

  If $m = 1$ we are in the case of an M/M/1 queue and:

$$C\Big(m, \frac{\lambda}{\mu}\Big) = \frac{1}{1 + (1-\rho)\frac{1}{\rho}} = \rho \tag{3.20}$$

- **Expected number of customers in the system**

  Assuming stability ($\lambda < m\mu$), we have that the system throughput $X = \lambda$ So in steady-state, expected number of customer is given by:

$$\bar{N} = E[n] = \frac{\rho}{1-\rho} C\Big(m, \frac{\lambda}{\mu}\Big) + \frac{\lambda}{\mu} \tag{3.21}$$

- **Expected response time**

  Having the same assumptions made in the previous point, using Little's Theorem we can derive:

$$\bar{R} = E[r] = \frac{C(m, \frac{\lambda}{\mu})}{m\mu - \lambda} + \frac{1}{\mu} \tag{3.22}$$

Taking the same example made for the M/M/1 queue, a telephone exchange with m different exchange units can be modeled as an M/M/m queue having the same assumption made in the previous section's example.

For now we have always considered the case of infinite queue capacity. If this assumption falls and we impose a maximum dimension we we face another issue.

We are now using an M/M/m/m queue to model our system, and since not all customers can be served we want to know what is the probability for a customer,to arrive in the system and not being rejected.

To solve this problem we use the *Erlang-B* formula made by *Agner Krarup Erlang*.

$$E_B = \left(m, \frac{\lambda}{\mu}\right) = \frac{(\lambda/\mu)^m}{m!} \left(\sum_{i=0}^{m} \frac{(\lambda/\mu)^i}{i!} \right)^{-1} \tag{3.23}$$

Since its computation is difficult and is not numerically stable is preferable to use the following recursive scheme:

$$\begin{cases} E_B\left(0, \frac{\lambda}{\mu}\right) = 1 \\ E_B\left(m, \frac{\lambda}{\mu}\right) = \frac{(\lambda/\mu) E_B(m-1, \lambda/\mu)}{m + (\lambda/\mu) E_B(m-1, \lambda/\mu)} \quad if \;\; m > 0 \end{cases} \tag{3.24}$$

To use the same example adopted before, if we restrict our telephone lines to be at most m, our exchange is modelled by an M/M/m/m queue.

3.1.6 M/M/∞ queue

This peculiar queuing model, as the notation should suggest, has: Poisson arrival process with rate λ, exponential service time with rate and , in contrast to the other models seen before, infinite number of servers. This queuing system usually model parallel independent computation.

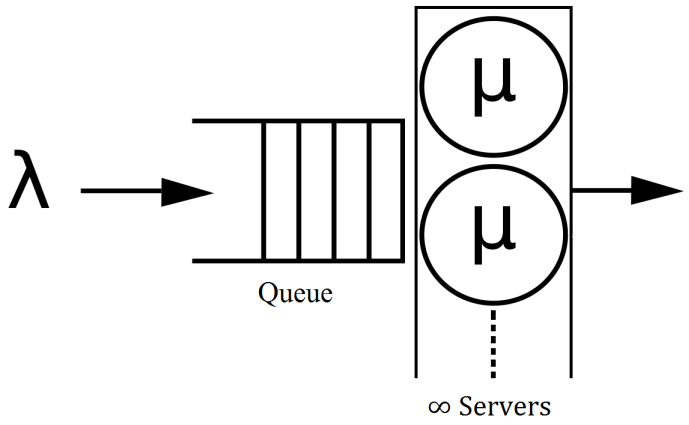

Figure 3.5: An M/M/∞ queue

For the steady state distribution of the model we can show that:

$$\pi(n) = \pi(0)\Big(\frac{\lambda}{\mu}\Big)^n \frac{1}{n!} \tag{3.25}$$

where $\pi(0)$ is:

$$\pi(0) = \left(\sum_{n=0}^{\infty}\Big(\frac{\lambda}{\mu}\Big)^n \frac{1}{n!}\right) = e^{-\lambda/\mu} \tag{3.26}$$

where the series $\sum_{n=0}^{\infty}(\lambda/\mu)^n \frac{1}{n!}$ is always convergent.

Then the steady state distribution $\pi(n)$ can be rewritten as:

$$\pi(n) = e^{-\lambda/\mu}\Big(\frac{\lambda}{\mu}\Big)^n \frac{1}{n!} \tag{3.27}$$

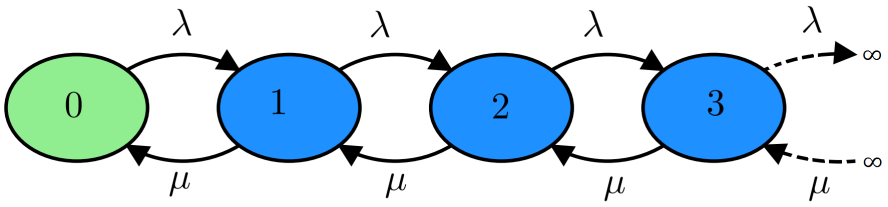

Figure 3.6: The CTMC of an M/M/∞ queue

It's performance measures are:

- **Expected response time**

  Since the response time for each customer arriving at the queue is described by an exponential distribution with rate μ, average response time is:

$$\bar{R} = \frac{1}{\mu} \tag{3.28}$$

- **Expected number of customers**

$$\bar{N} = \frac{\lambda}{\mu} \tag{3.29}$$

- **Throughput**

$$X = \lambda \tag{3.30}$$

- **Average wait time at the queue**

  Since the number of servers is infinite, customers does not have wait time, then the average wait time is zero.

3.1.7 M/G/1 queue

M/G/1 queuing system is characterized by a Poisson arrival process with rate λ, a *general* service time distribution that has mean $1/\mu$ and variance σ_s^2. Since is not explicitly written the queuing discipline is FCFS, so the first customer to be served is the first to arrive. The queue is not restricted on the buffer size that is infinite and has a single server.

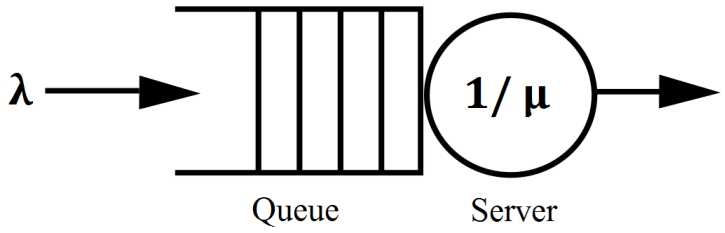

Figure 3.7: An M/G/1 queue

For this queue the stability condition is given by $\lambda < \mu$ with the load factor $\rho = \lambda/\mu < 1$ as seen in the M/M/m queue. In fact this model shares a lot of features with the model seen in chapter 3.1.5 as has the same arrival process distribution and the same number of server along with the same buffer size.

Noteworthy performance measures for this model are:

- **Average queue length**:

  This measure can be computed thanks to *P-K Formula*. P-K is the acronym of Pollaczek–Khinchine, the names of the two mathematicians who first propose it. Austrian-French Felix Pollaczek first published it in 1930[29] then two years later the Soviet Aleksandr Khinchin [30]revise the formula in probabilistic terms.

  $$E[n_q] = \frac{\rho^2 + \lambda^2\sigma_s^2}{2(1-\rho)} \tag{3.31}$$

  If we set $\sigma_s^2 = 1/\mu^2$ we have:

  $$E[n_q] = \frac{\rho^2\rho^2}{2(1-\rho)} = \frac{2\rho^2}{2(1-\rho)} = \frac{\rho^2}{1-\rho} \tag{3.32}$$

  that is the value for an M/M/1 queue.

  The core of the PK formula is provided by intuition that the queue length also depends on the variance in service time, in addition to the ratio between expected service time and arrival rate.

- **Expected number of customers in queue**:

  $$\bar{N} = E[n] = \frac{\rho^2 + \lambda^2\sigma_s^2}{2(1-\rho)} + \rho \tag{3.33}$$

- **Expected waiting time**:

$$E[w] = \frac{\rho + \lambda\mu\sigma_s^2}{2(\mu - \lambda)} \tag{3.34}$$

- **Expected response time**:

$$\bar{R} = E[r] = \frac{\rho + \lambda\mu\sigma_s^2}{2(\mu - \lambda)} + \frac{1}{\mu} \tag{3.35}$$

3.2 Queuing model for BTC blockchain

Our case study is modeled as an $M/M^{X_i}/1$ queue, the service takes places with batch of fixed dimension, modeled by a random variable.

An $M/M^{X_i}/1$ queue, by recalling chapter 3.1.1, denotes a model with Poisson arrival process, a batch of X_i exponential service times and one server with FIFO discipline.

Let X_i the r.v. that describes the number of transactions placed in the $i-th$ block.

We assume Y_i be the r.v. that set how many transactions the system would put in a block if the pending transactions queue had infinite size.

In this case Y_i, are i.i.d. random variables, meaning that all the variables have the same probability distribution and are mutually independent.

Hence,

$$Y_i \sim \bar{P} \tag{3.36}$$

where $\bar{P}$ is a discrete probability distribution:

$$\bar{P} = (P_1, ..., P_k) \tag{3.37}$$

P_1 denote the probability that block contains exactly one transaction, P_k is the probability that the block contains exactly k transactions.

Observation 1. Y_i has finite support.

It exists a maximum number of transactions that can be inserted in a block (since its dimension is finite).

Then $\bar{P} = (p_0, p_1, p_2, ..., p_T)$ has the following properties:

- $T < \infty$
- $p_k >= 0, \forall\ k = 1, ..., T$

- $\sum_{k=1}^{T} p_k = 1$

Then we can say that $X_i = min(N_i, Y_i)$ Where N_i is the number of transaction in the queue at the moment of the creation of the $i-th$ block.

3.2.1 Markovian Analysis

We model the pending transaction list in a way that state $n(t)$ represents the number of pending transactions at time $t > 0$.

We know that the time required to mine the $i-th$ block is described by a r.v. Z_i, where z_i are i.i.d. and exponentially distributed with a parameter μ.

For the law of rare event [31], we assume the arrival process to be a Poisson process with parameter λ

The *law of rare events*, also known as Poisson limit theorem (named after Siméon Denis Poisson) states that, under certain conditions, the Poisson distribution can be used as an approximation to the binomial distribution.

By recalling chapter 3.1.3, assuming the arrival process to be Poisson is a very excellent approximation for real systems where the amount of customers in the queue is very big, each single client has a tiny, negligible effect on system performance and customers independence is ensured. This fits perfectly the case with our scenario.

Hence the following propostions are valid:

Proposition 1. $n(t)$ is a Continuous Tine Markov Chain (CTMC) for any $\bar{P}$

Proposition 2. The chain $n(t)$ is irreducible.

Proposition 3. The chain $n(t)$ is ergodic then admits one unique stationary distribution iff :

$$\lambda < \mu E[Y] \tag{3.38}$$

Where $E[Y]$ is the expected value of the r.v. Y_i

3.2.2 Stationary distribution computation

Let us consider state i, $i > 0$. There are two events that make us leave state i : the arrival of a new transaction and the mining of a new block. With the total rate of $\lambda + \mu$

State i can be reached from the state $i-1$ for the arrival of a transaction with rate λ or from the state $i+j,\ j>0$, because a new block of dimension j has been mined. It follows that the balance equations are:

1.
$$\pi_i(\lambda+\mu) = \pi_{i-1}\lambda + \sum_{j=0}^{T} \pi_{i+j}\mu P_j, \quad i>0 \tag{3.39}$$

In the state 0, blocks are mined anyway, even if empty. This state is reached from the states $j,\ 0 <= j <= T$, with different probabilities.

For instance, if the chain is in the state 3 and the r.v. Y has value 5, the variable X takes the value 3.

2.
$$\pi_0(\lambda+\mu) = \sum_{j=0}^{T} \pi_j \mu \sum_{h=j}^{T} P_h \tag{3.40}$$

For the Proposition 3 the system

$$\begin{cases} \pi_i(\lambda+\mu) = \pi_{i-1}\lambda + \mu \sum_{j=1}^{T} \pi_{i+j}P_j, \quad i>0 \\ \pi_0(\lambda+\mu) = \mu \sum_{j=0}^{T} \pi_j \sum_{h=j}^{T} P_h \end{cases} \tag{3.41}$$

with the normalizing condition:

$$\sum_{i=0}^{\infty} \pi_i = 1 \tag{3.42}$$

has an unique solution when the system is stable.

Considering equation (3.39) and assuming that:

3.
$$\pi_i \propto \rho^i \tag{3.43}$$

We have:

$$\rho^i(\lambda+\mu) = \rho^{i-1}\lambda + \mu \sum_{j=1}^{T} \rho^{i+j} p_{\mathrm{j}} \tag{3.44}$$

By dividing the latter for ρ^{i-1} we obtain:

$$\rho(\lambda+\mu) = \lambda + \mu \sum_{j=1}^{T} \rho^{j+1} p_j \tag{3.45}$$

We observe that the equation is independent of i.

Hence, by rewriting the latter we have

4.

$$\mu \sum_{j=1}^{T} \rho^{j+1} P_j - \rho(\lambda + \mu) + \lambda = 0 \tag{3.46}$$

For the Fundamental Theorem of Algebra we know that this has $j+1$ total solutions and only one of them can be real and such that $0 < \rho < 1$ by the ergodicity of the CTMC. Thus this confirms that the assumption on the geometric distribution of the stationary probabilities is correct.

Assuming (3.43), thanks to the Proposition 3 we know that the series (3.42) must converge.

Since the series is geometric then $0 < \rho < 1$. There cannot be any other real solution in $(0, 1)$ because the ergodicity of the CTMC implies the uniqueness of the solution.

Let $\hat{\rho}$ be the solution of (11) with $0 < \hat{\rho} < 1$. In this settings the stationary distribution, following the proof of an $M/M/1$, becomes :

$$\pi_{\mathrm{i}} = (1 - \hat{\rho})\hat{\rho}^i, \quad i > 0 \tag{3.47}$$

3.2.3 Sanity check

Let suppose $\bar{p} = (0, 1)$ with $T = 1$. Then the chain is the same of an $M/M/1$ queue.

By rewriting equation (3.46) as :

$$\mu\rho^2 - \rho(\lambda + \mu) + \lambda = 0 \tag{3.48}$$

$$\Delta = (\lambda + \mu)^2 - 4\lambda\mu = \lambda^2 + \mu^2 + 2\lambda\mu - 4\lambda\mu = (\mu - \lambda)^2 \tag{3.49}$$

$$\rho_{1,2} = \frac{(\lambda + \mu) \pm (\mu - \lambda)}{2\mu} = \begin{cases} \frac{\lambda}{\mu} \\ 1 \end{cases} \tag{3.50}$$

The condition of the proposition 3 implies $\lambda < \mu$, then the unique solution in the interval $(0, 1)$ is:

$$\hat{\rho} = \frac{\lambda}{\mu} \tag{3.51}$$

This reproduces exactly the result of an M/M/1 queue.

$$\pi_{\mathrm{i}} = \Big(1 - \frac{\lambda}{\mu}\Big)\Big(\frac{\lambda}{\mu}\Big)^i, \quad i > 0 \tag{3.52}$$

3.3 Model parameterization

Let's recall the equation :

$$\mu \sum_{j=1}^{T} \rho^{j+1} P_j - \rho(\lambda + \mu)\lambda = 0 \tag{3.53}$$

Let T be the maximum number of transaction per block that in our setting is 4109.

Let P_j be the probability that a block will contains j transactions.

Let μ be the parameter of the r.v that model the time required to mine a block. μ was set as $1/600$ as the Bitcoin system ensures that a block is mined every 10 minutes in average.

Let λ be the rate of the Poisson arrival process. Since $\lambda < \mu E[T]$ with $E[T]$ the average number of transaction per block, that is 1385.51, we have set this parameter at three different levels to fit the cases of a system with low, middle and high load.

- $\lambda_1 = 0.5 * \mu * E[T]$
- $\lambda_2 = 0.7 * \mu * E[T]$
- $\lambda_3 = 0.9 * \mu * E[T]$

We solve equation 3.53 with the latter levels of workload with a simple homotopy implemented in Wolfram Mathematica. The results are the following for the respective three levels of load:

- $\rho_1 = 0.999108$
- $\rho_2 = 0.99959$
- $\rho_3 = 0.999945$

With an average number of transaction in the queue (given by $\rho/1 - \rho$) respectively of 1120.22, 2438 and 18300.

Since we model our pending transaction list as an M/M/1 queue, the average response time i.e. the time interval between the moment a transaction arrives in the queue and the instant when it is packed in a block, as we saw in chapter (3.1.4),is given by:

$$\bar{R} = \frac{1}{\mu - \lambda} \tag{3.54}$$

Then the response times for the latter levels of load with the service rate μ set as $1/600$, are :

- $\bar{R}_1 = 0.8661071s$
- $\bar{R}_2 = 1.443512s$
- $\bar{R}_3 = 4.330535s$

The figure below depicts the response times for each workload level (green for low load, yellow for middle load and red for high load).

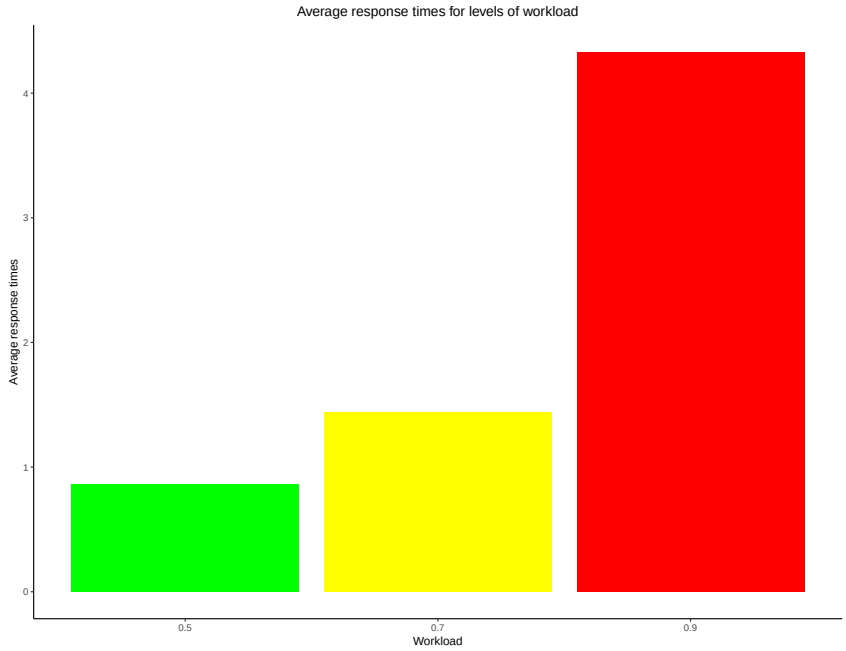

Figure 3.8: Plot of the response times for the three levels of workload

3.4 Parameterized model analysis

By investigating the blockchain average transaction rate, i.e. the average number of Bitcoin transactions added to the mempool per second, turned out to be 2.479515 in the last period.

Its distribution is plotted in the figure below.

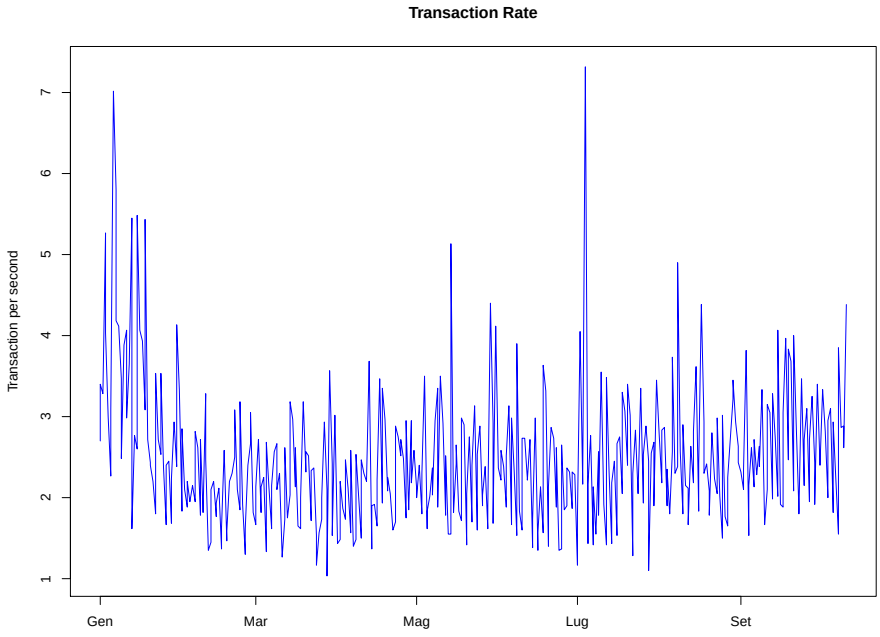

Figure 3.9: Plot of transaction rate

Having this average coincide with having 1488 transactions added to the mempool every 10 minutes, which, as said before, is the average time to mine a new block.

The distribution of the number of transactions per block is shown in the following plot.

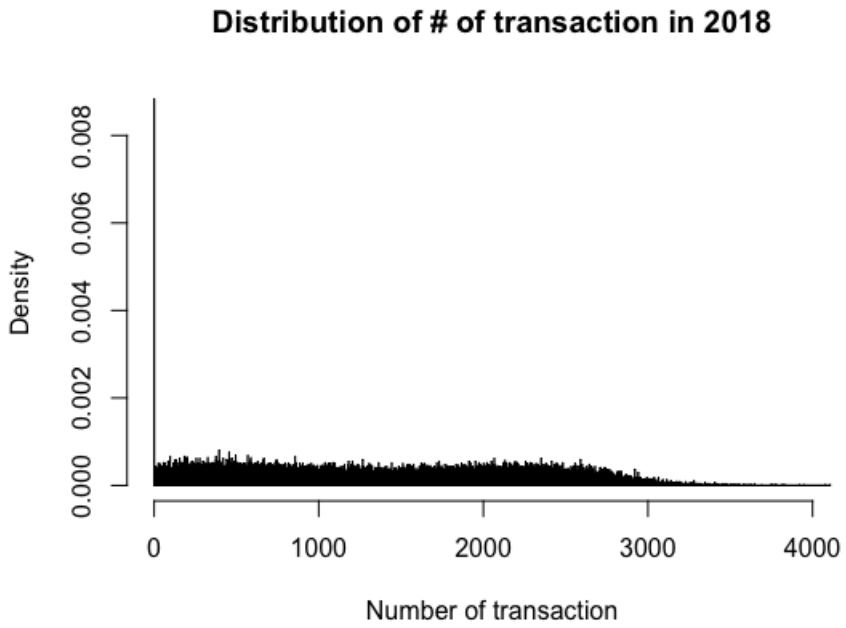

Figure 3.10: Distribution of number of transactions per block

As we can obviously see, most blocks are empty. The mean of this distribution is 1385.51 transactions per block.

Considering this average value and the average service rate retrieved before, we see that the real Bitcoin system can not bundle all transactions in the pending list, which as a result develops continually. When the list is saturated, old transactions are dropped in FIFO order to ensure the stability of the system.

Since in our model the pending transaction queue has infinite size, there is no dropping mechanism. The continuous development of the queue leads the old transaction to persist in the system for a long time resulting with an average number of transactions in the system rising with the workload.

3.5 Chapter conclusion

In this chapter we saw an introduction to the queuing theory: beginning with the notation of the Kendall which gave a formal and uniform way of immediately

describing a system, through the main type of queuing models M/M/1, M/M/m, M/M/∞ and M/G/1.

The core of the chapter was the formal definition of a queuing model that attempts to predict the system mempool saturation accurately, with the only restriction of having an infinite pending transaction list. Results are pretty compelling as the system could not catch the rate at which transaction is generated in the real case and a transaction drop is required to keep the system stable. In the suggested model, we have that the forecast are really precise highlighting how the pending transactions list grows to infinite causing to old "unlucky" transaction to remain unresolved for a long time.

For this reason in the real Bitcoin system customers rely on transaction fees, which act like a priority level for the single transactions in the pending list. Miners, when selecting the transactions to add in a new block, behaves with the only purpose of maximizing their income. Hence, a transaction with the highest fee is likely to be the next one included in the following new block successfully added to the chain.

Chapter 4

Block size impact on performance

Chapter introduction

As we deepened in chapter 2.1, the bitcoin community has been wondering for years whether raising the actual block size limit of 1 MB could be the solution to the problems affecting the network nowadays, which puts it at a disadvantage compared to other large payment systems. The focus of this chapter is an analysis of two key aspect, energy waste and goodput, and how they should be affected by the increased block size. First of all we provide some vital formulas and definition mandatory for the subsequent analysis of the behavior of the two indices with variation of block size.

4.1 Definitions

To offer a mathematical formal definition of the energy waste and goodput indices of our system, we must first point out some preliminary formulas.

As we have already figured out the system throughput in terms of block that define the rate at which information are produced in our system, indicated with T_{block}, is:

$$T_{block} = \frac{1}{600} \ [blocks/s] \tag{4.1}$$

As the bitcoin system adjusts the mining difficulty to maintain an average time of 10 minutes to mine a block.

With k we indicate the value of delay cost (in terms of delay/size) of a block which, as we saw in chapter 2.2 about information propagation in the bitcoin network, measure the time delay each kilobytes causes to the spread of a block or a transaction in the network.

A significance of this metric can be discovered in Article [23] , but since the paper is from 2013 we use the most updated value of 12,828 from the BitcoinStat website [32]

We assume the mining time to be defined by an exponential distribution X with parameter $\mu = 600^{-1}$ and the propagation time also exponentially distributed, expressed as Y, with parameter $T_D^{-1} = kB^{-1}$

$$X \sim exp(\mu) \tag{4.2}$$

$$Y \sim exp(T_D^{-1}) \tag{4.3}$$

If we specify the average block size with B, the probability of finding an empty block mined by the miners pool Pe is provided by:

$$P_e = \frac{T_D}{T_D + \frac{1}{\mu}} \tag{4.4}$$

which is given by:

$$\begin{aligned} P_e &= P_r\{X < Y\} \\ &= \frac{\mu}{\mu + T_D^{-1}} \\ &= \frac{\mu}{\frac{T_D\mu+1}{T_D}} \\ &= \frac{T_D\mu}{T_D\mu + 1} \\ &= \frac{T_D}{T_D + \frac{1}{\mu}} \end{aligned}$$

Given the latter concept and the network's average energy consumption to mine a block M, we can now lastly describe the energy waste W_W as:

$$W_W = P_e M \tag{4.5}$$

We consider as energy waste the amount of energy used to mine an empty block.

Hence given (5.4):

$$W_W = P_e M = \frac{\mu}{\mu + T_D^{-1}} M \tag{4.6}$$

For clarity purpose only do not indicate the value of M and leave it as unit of measurement.

We can now straightforwardly define the system's throughput in terms of transactions $T_{transac}$ as:

$$T_{transac} = \frac{B}{kB + \frac{1}{\mu}} \tag{4.7}$$

which is obtained by:

$$\begin{aligned}
T_{transac} &= \mu B(1 - P_e) \\
&= \mu B(1 - \frac{T_D \mu}{T_D - \frac{1}{\mu}}) \\
&= \mu B(1 - \frac{T_D \mu}{\mu T_D + 1}) \\
&= \mu B(\frac{\mu T_D + 1 - \mu T_D}{\mu T_D + 1}) \\
&= B(\frac{\mu}{\mu T_D + 1}) \\
&= \frac{\mu B}{\mu k B + 1} \\
&= \frac{\mu B}{\mu(kB + \frac{1}{\mu})} \\
&= \frac{B}{kB + \frac{1}{\mu}}
\end{aligned}$$

This measure the goodput of our system.

Definition 13. The *goodput* is defined as the number of useful bits of information delivered by the network to a specific destination per time unit.

In our case scenario, since we targeted the absence of transaction (or otherwise the presence of empty blocks) as useless information, the transaction throughput represents the goodput of the system.

4.2 Block size impact analysis

The first element we evaluate is the variation in the waste of energy in the bitcoin network with the rise of the block size. Let us recall the formula we derived before:

$$W_W = \frac{T_D}{T_D + \frac{1}{\mu}}[MB/s * M]$$

For the latter formula, we first study the limit cases:

$$\lim_{T_D \to \infty} W_W = \lim_{T_D \to \infty} \frac{T_D}{T_D + \frac{1}{\mu}} = 1 \tag{4.8}$$

$$\lim_{T_D \to 0} W_W = \lim_{T_D \to 0} \frac{T_D}{T_D + \frac{1}{\mu}} = 0 \tag{4.9}$$

This implies that the value of waste varies in a range from 0 to 1.

Then we issued the following plot

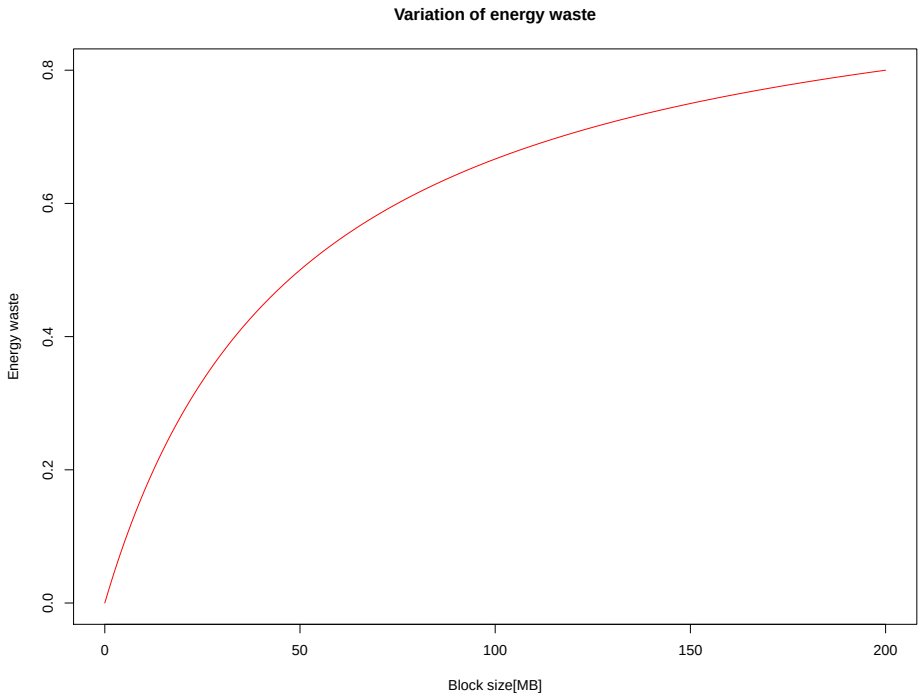

Figure 4.1: Plot of energy versus block size

Moving on to the goodput, the number of useful information (blocks and transactions) flooded in the network for unit of time, we recall the definition given before:

$$Goodput = \frac{B}{kB + \frac{1}{\mu}}$$

As for the energy waste we test the limit cases:

$$\lim_{B\to\infty} Goodput = \lim_{T_D\to\infty} \frac{B}{kB + \frac{1}{\mu}} = 1/12 = 0.08\overline{3}[MB/s] \tag{4.10}$$

$$\lim_{B\to 0} Goodput = \lim_{T_D\to 0} \frac{B}{kB + \frac{1}{\mu}} = 0 \tag{4.11}$$

From this analysis emerge the optimal value for the goodput that is $0.08\overline{3}$ that will be an asymptote for the plot of this index.

Hence, we can also observe that the goodput is severely limited by the network latency.

To study how the value of the goodput changes along with the expanded block size, we compile a plot for this ratio.

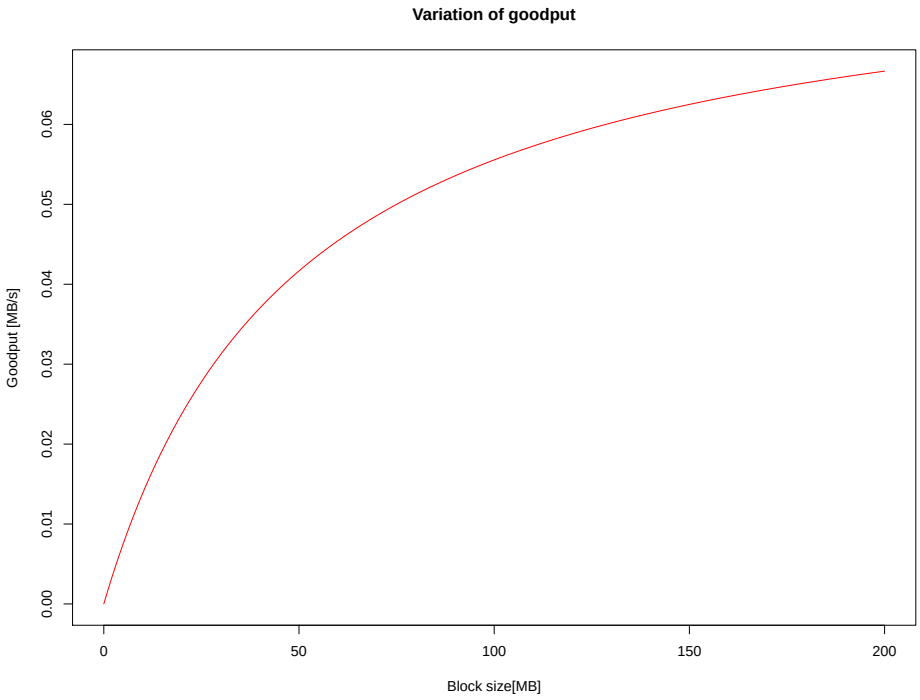

Figure 4.2: Plot of system goodput versus block size

As we can see both plots 4.1 and 4.2 shows a comparable trend even if the energy waste has an asymptote at $y = 1$ while goodput has it at $0.08\overline{3}$ as

emerged from the limit analysis.

4.3 Chapter conclusion

In this section, we focused on two elements that are immediately affected from the overshoot of the actual size threshold for a 1 MB bitcoin block.

From what arose from the study of the two index' graphs and their limit case tests, both indicators rise alongside the block size growth. The energy wasted in empty blocks mining for the same value of block size takes on larger value, reaching almost the totality of the energy consumed for blocks larger than 200 MB, than the system goodput that never exceeds $0.08\overline{3}$ that result strictly limited by the network latency.

From an initial analysis, this could suggest that an increase in the block size limit is not worth the money as the elevated energy wasted in the process, and the resulting cash loss, not supported by a comparable increase in the goodput, will prevent a standard miner from starting to create fresh blocks (since it bases iis earnings mainly on transaction fees).

However, bigger blocks could store more transactions and even if the blocks throughput is restricted by the system regulations (one block mined every 10 minutes averagely as we saw before), by increasing the amount of transactions per second, miners will earn much more profit from transaction fees while wasting almost the totality of the energy used in the mining process in creating empty blocks.

A profitable situation is reached only if we assume that the average transaction fee value does not reduce due to any rise in block size and if we assume that the average transaction size remains stable. In this context an increase in block size brings a direct rise in the average amount of transactions and consequently in the average number of transaction fees. Since the value of transaction fees do not reduce, owing to the increase in the amount of transactions, the miner has more profit. Mining is cost-effective only if the earnings, summed up with the award provided by the block solution, are greater than the price of the energy wasted in the process of mining empty blocks.

On the other hand, if an increase of the block size limit is followed by a decrease on the average value of the transactions fee, we will reach a point, for large sized blocks, where the profit gained by the sum of the transaction fees and the block reward will not sustain the the expenditure energy wasted from

mining empty blocks. This causes the miner to lose cash and any interest in joining the network.

Chapter 5

Conclusion

The drastic evolution of cryptocurrencies, including Bitcoins, has raised several issues and questions. In this dissertation, we have developed some of them, first of all the issue of poor throughput, compared to main payment processors, that restrict Bitcoin's potential to become a currency and payment method of everyday use. Related to this, the problem of block size, which is one of the variables that leads to lower throughput, that since is limited to 1 MB per block restricts the amount of transactions that can be loaded into it, given the increase in the system's transaction rate.

Both of this problem are still open and have been present since the dawn of the bitcoin era even if they were not determining factors in the early days due to the cryptocurrency's low utilization. The community is still wondering whether the limit should be exceeded in the future, even if the change seems imperative.

The most intriguing aspect deepened in this dissertation was the topic of the presence in the ledger of empty blocks, which have risen in quantities in recent times. It is interesting to see that from the analysis made with last year data of the distance of normal blocks from empty blocks they turns out to be gemoetrically distributed. This implies that due the memoryless property, held by the geometric distrubution, we can state that information about how many empty blocks were present in the past does not contribute to the probability that the next mined block will be devoid of any worthily transactions.

We then suggested a queuing model to attempt to forecast the mempool's saturation, assuming there is an infinite size list of pending transactions. The model forecast conclusions are precise, but only in the event of low workload,

as otherwise the system result unstable.

The purpose behind generating empty blocks instead of creating normal transaction-full blocks behavior is still not fully resolved. One of the reasons for this is undoubtedly the aim of miners to retain at least the portion of the complete revenue consisting of the proof of work prize (12.5 BTC currently) by sacrificing the fraction made of the transaction fees. A faster block creation, i.e. building a block without wrapping in any kind of transaction, gives nodes a higher security of income due to network latency.

The final study addressed in this dissertation is created by analyzing the drawbacks of raising the limit at the size of blocks by studying how the waste of energy and the goodput of the system change alongside the variation of the size limit.

The result of the examination is quite clear: even if the goodput is limited, the bigger block size threshold will provide more profit for the miners as there will be more transaction to fit in a larger block that will reward numerically more fees. As a drawback miners will waste the vast majority of the energy.

It is news of these days that the blockchain ledger has registered the presence of a new orphan blocks[33]. Orhpan blocks are blocks that have no identified parent in the current longest blockchain.

In parallel researchers and blockchain experts noticed a decline in the number of these particular blocks in the chain[34].

An interesting future works should be a correlation analysis between the decrease of orphan blocks with the variation of empty blocks

Appendices

Appendix A

Libraries

A.1 Python: Blockchain parser

Since the downloaded blockchain files are in a proprietary compressed format to minimize spatial impact on Bitcoin node drives to interpreting the blockchain we used the *python-bitcoin-blockchain-parser* from Antoine Le Calvez. This Python parser available on Github[25] allows us to read and operate with data contained in the raw *.blk* blockchain files.

This library allows us to parse the ledger blocks and transaction in two ways:

- Unordered : with the function $get_unordered_blocks()$
- Ordered: with the function $get_ordered_blocks(< indexpath >)$

The second one is indicated in cases where it is useful to maintain blocks temporal order, but it requires substantially more time since it has to build the whole block index based on the indicated path, a priori.

Once chosen the parsing method the library makes available Blocks and Transaction data modeled in distinct objects.

A.2 R: distribution fitting and hypothesis test

To compute the Maximum Likelihood Estimation (MLE) fit and then to test the result with the Chi Square Hypothesis Test for Goodness of Fit we used R language. There are several libraries available for this type of statistics, like the *chisq.test* from the default R library *stat*. Indeed one of the most complete

and simple to use is the *fitdistrplus* made by Aurélie Siberchicot. This library provides two functions *fitdist* and *gofstat* that suits perfectly our case.

fitdist by taking as inputs a numeric vector of data and a reference distribution, like the geometric in our case, perform a fit of univariate distribution to the given data. The default method, and the one we used, is the already mentioned MLE, but are also available Moment Matching Estimation, Quantile Matching Estimation and Maximum Goodness-of-Fit Estimation. When computed it returns an object containing two useful plots like the ones on Figure 2.11 and 2.12 in addition to the fitted distribution's parameter.

To perform the Chi Square Test for Goodness of Fit with the *gofstat* function we just gave as input the result of the previous fit. The test returns the measure of the Chi-squared statistic, the degrees of freedom as well at the pivotal output, the *p-value*.

Appendix B

Script

The following is the python script we used to parse the Blockchain:

```
# Instantiate the Blockchain by giving the path to
    the directory
# containing the .blk files created by bitcoind
path = '/Users/andreasina/Documents/original blocks
    copia/'
blockchain = Blockchain(os.path.expanduser(path))


block_file = open("blocks.txt","w")
max_buf_size = 1024
date_threshold = datetime(2018,01,01)#selecting
    blocks from 2018

start = 0
step = 12983 # 545283 / 42
end = step

blocks = blockchain.get_ordered_blocks(path+"index",
    start= start, end = end)
date = next(blocks).header.timestamp
blocks_buffer = ""
```

```
while start < end:
    for blk in blocks:
        if len(blocks_buffer.encode('utf-8')) >
           max_buf_size:
            block_file.write(blocks_buffer) #write
               buffer on file
            blocks_buffer = None #delete the buffer
            blocks_buffer = ""
        if blk.header.timestamp > date_threshold:
            blocks_buffer +=(str(blk.height)+";"+
            str(blk.n_transactions-1)+
            ";"+str(blk.header.timestamp -
               date)+";"+str(blk.size)+"\n")
        date = blk.header.timestamp
    start = end
    end += step
    blocks = None
    blocks = blockchain.get_ordered_blocks(path
       +"index", start = start,
    end = end)
'''
write the remaining blocks on file
'''
block_file.write(blocks_buffer)
blocks_buffer = None
block_file.close()
```

The results have been saved in a separate text file.

Bibliography

[1] Bitcoin developer guide. Bitcoin community.

[2] T. Dahlberg. K.Korpela, J.Hallikas. Digital supply chain transformation toward blockchain integration. *Hawaii International Conference on System Sciences*, 2017.

[3] A.Ekblaw et al. Case study for blockchain in healthcare:"medrec" prototype for electronic healthrecords and medical research data. *2nd International Conference on Open & Big Data*, 2016.

[4] Ralph Merkle. A digital signature based on a conventional encryption function. *Advances in Cryptology — CRYPTO '87. Lecture Notes in Computer Science*, 293:369–378, 1988.

[5] Ralph Merkle. Method of providing digital signatures. *US patent 4309569, assigned to The Board Of Trustees Of The Leland Stanford Junior University*, 1979.

[6] Satoshi Nakamoto. Bitcoin: A peer-to-peer electronic cash system., 2009.

[7] V.Oksman N.King and C.Bry. Updating and distributing encryption keys. *US20100042841A1*, 2008.

[8] Bitcointalk User: slush. Slush pool (slushpool.com); overt asicboost; world first mining pool. Forum Post on Bitcointalk.org, November 2010.

[9] Charles Arthur. Slush pool (slushpool.com); overt asicboost; world first mining pool, November 2011.

[10] Tim Worstall. Bitcoin: the first $500,000 theft. Forbes, June 2011.

[11] Federal Bureau of Investigation. Bitcoin virtual currency: unique features present distinct challenges for deterring illegal activities. FBI Unclassified file, May 2012.

[12] Jose Pagliery. Fbi shuts down online drug market silk road. CNN, October 2013.

[13] Sophie Knight Yoshifumi Takemoto. Mt. gox files for bankruptcy, hit with lawsuit. Reuters, February 2014.

[14] Bitcoin pubblication trend by year. Dimension.ai.

[15] Aaron van Wirdum. How bitcoin's second halving came and went, and not much happened. Blog post on bitcoinmagazine.com, July 2016.

[16] Laura Shin. A second version of bitcoin has launched. could it threaten the original? Forbes website, August 2017.

[17] Bitcoinblockhalf : Bitcoin block reward halving countdown. Website.

[18] K. Croman et al. On scaling decentralized blockchains, 2016.

[19] Why was the target block time chosen to be 10 minutes? Stackexchange post.

[20] Reddit user: u/LiThiuMElectro. Why is increasing the block size bad? Thread on Reddit.

[21] Quora user. What are the primary arguments against increasing bitcoin's block size? Thread on Quora.

[22] The blocksize limit controversy. Bitcoin wiki page.

[23] Christian Decker and Roger Wattenhofer. Information propagation in the bitcoin network. *13-th IEEE International Conference on Peer-to-Peer Computing.*, 2013.

[24] Karlsruher Institut für Technologie. Kit blockchain data website.

[25] Python-bitcoin-blockchain-parser, a parser for the bitcoin blockchain.

[26] J.Bernoulli. Ars conjectandi, 1713.

[27] Russell B. Millar. Maximum likelihood estimation and inference: With examples in r, sas. *Wiley. 978-0-470-09482-2*, 2011.

[28] D. G. Kendall. Stochastic processes occurring in the theory of queues and their analysis by the method of the imbedded markov chain. *The Annals of Mathematical Statistics*, 24(3):338, 1953.

[29] F. Pollaczek. Über eine aufgabe der wahrscheinlichkeitstheorie. *Mathematische Zeitschrift*, 32:64–100, 1930.

[30] A. Y Khintchine. Mathematical theory of a stationary queue. *Matematicheskii Sbornik*, 39(4):73–84, 1932.

[31] R.Reiss M.Falk, J.Hüsler. *Laws of Small Numbers: Extremes and Rare Events*. Birkhäuser, 2011.

[32] Bitcoinstat : Propagation time chart. Website.

[33] Emanuele Pagliari. An orphan block on the bitcoin (btc) blockchain. Blog Post on Cryptonomist, May 2019.

[34] Larry Cermak. Orphaned blocks in bitcoin are becoming less frequent as miners improve direct communication. Blog Post on Theblockcrypto.