NATIONAL AND KAPODISTRIAN UNIVERSITY OF ATHENS

SCHOOL OF SCIENCE
DEPARTMENT OF INFORMATICS AND TELECOMMUNICATIONS

GRADUATE PROGRAM IN COMPUTER SCIENCE
COMPUTING SYSTEMS

MASTER THESIS

Reduce Blockchain Nodes Storage Requirements by using Distributed Hash Table.

Evangelos K. Kolyvas

Supervisor: **Mema Roussopoulos,** Associate Professor N.K.U.A.

ATHENS

FEBRUARY 2019

ΕΘΝΙΚΟ ΚΑΙ ΚΑΠΟΔΙΣΤΡΙΑΚΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ

ΣΧΟΛΗ ΘΕΤΙΚΩΝ ΕΠΙΣΤΗΜΩΝ
ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΤΗΛΕΠΙΚΟΙΝΩΝΙΩΝ

ΠΡΟΓΡΑΜΜΑ ΜΕΤΑΠΤΥΧΙΑΚΩΝ ΣΠΟΥΔΩΝ ΣΤΗΝ ΠΛΗΡΟΦΟΡΙΚΗ
ΥΠΟΛΟΓΙΣΤΙΚΑ ΣΥΣΤΗΜΑΤΑ

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Μείωση Αποθηκευτικών Απαιτήσεων Blockchain Κόμβων με χρήση Distributed Hash Table.

Ευάγγελος Κ. Κολυβάς

Επιβλέπων: **Μέμα Ρουσσοπούλου,** Αναπληρώτρια Καθηγήτρια Ε.Κ.Π.Α.

ΑΘΗΝΑ

ΦΕΒΡΟΥΑΡΙΟΣ 2019

MASTER THESIS

Reduce Blockchain Nodes Storage Requirements by using Distributed Hash Table.

Evangelos K. Kolyvas
Student ID: M1595

SUPERVISOR: **Mema Roussopoulos,** Associate Professor N.K.U.A.

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Μείωση Αποθηκευτικών Απαιτήσεων Blockchain Κόμβων με χρήση Distributed Hash Table.

Ευάγγελος Κ. Κολυβάς
Α.Μ.: Μ1595

ΕΠΙΒΛΕΠΩΝ: **Μέμα Ρουσσοπούλου,** Αναπληρώτρια Καθηγήτρια Ε.Κ.Π.Α.

ABSTRACT

A distributed ledger is a database that is spread across several nodes on a peer-to-peer network. The main characteristic of such system is the lack of central authority: Each node replicates and saves an identical copy of the ledger. When a node hears a client's transaction request, it propagates the request to its peers. Every node on the network processes every transaction, coming to its own conclusions. When a ledger update happens, each node constructs a new state of the ledger based on the transactions it has heard, and then the nodes vote by consensus algorithm on which copy is correct. Once a consensus has been determined, all nodes update themselves with the new, correct, copy of the ledger. Distributed ledgers use blockchain protocols as one main means of implementation.

Nowadays, blockchains and distributed ledgers attract massive attention and trigger multiple projects in different industries. However, the financial industry is seen as a primary user of the blockchain concept. This is due to the fact that the technology is primarily used to verify transactions, within digital currencies. The most well-known application of this technology is the accounting method for the virtual currency, Bitcoin. Bitcoin is a form of money that no government or bank can control. Instead, transactions are verified by its users, eliminating the need for a third party to process or store payments.

However, apart from its benefits, blockchain technology has some significant drawbacks: low throughput, big latency, security and privacy issues, high energy consumption and dealing with the ever-growing ledger's data volume. On the whole, this basically means that blockchains face scalability problems. Some notable and well-known solutions for the scalability problem include:

- Usage of different consensus algorithms for the agreement among the nodes (proof-of-work, proof-of-stake, byzantine agreement).
- Replacement of the underlying chain-structured ledger with a directed acyclic graph.
- Build a payment network over the chain (the off-chain approach) where applications don't have to put every single transaction in the blockchain, instead to use the blockchain as a court system if there is a dispute between the parties.

The purpose of this master thesis is the study and the analysis of the novel blockchain technology, as well as a proposal of a new approach on its implementation. In particular, it considers the idea of organizing the blockchain peers into a DHT in order to store the blockchain data: As blockchains get older, their size gets larger. Currently, Bitcoin blockchain is over than 190GB. In addition, as long as some of the adove ideas about the increase of the transaction rate put into practice, blockchain size will grow even faster. The ever-growing size of the blockchain it is a factor of making the system more centralized, since too few of the nodes will be able to keep a full replica of the data. After all, archival nodes with open ports, have little or no incentives to waste their storage and their bandwidth, in order to keep all the data and serve it to a bootstrapping node. Thus, it is reasonable to mitigate their load (especially that we call "cold data", some historical blocks) by splitting it into a DHT. The evaluation of this proposal is achieved through simulations of the first and most popular blockchain network, the Bitcoin: We build a simplified DHT that reminds the Chord. We tune our system and we make a system whose characteristics is similar to Bitcoin. We measure several aspects of our system such as capacity savings per node compared to the amount of data that a full node stores today, load balance, query latency, bandwidth usage. We present our results, we make comments on them, and we draw a conclusion.

SUBJECT AREA: Blockchains, Distributed Systems, Peer-to-peer networks

KEYWORDS: Blockchain, Bitcoin, Distributed Hash Table, Chord, Scalability, Storage, Simulation, Markov Chain Monte Carlo

ΠΕΡΙΛΗΨΗ

Ένας κατανεμημένος κατάλογος είναι μια βάση δεδομένων διεσπαρμένη σε διάφορους κόμβους σε ένα δίκτυο ομότιμων χρηστών. Το κύριο χαρακτηριστικό αυτού του συστήματος είναι η έλλειψη κεντρικής εξουσίας: Κάθε κόμβος αναπαράγει και αποθηκεύει ένα πανομοιότυπο αντίγραφο του καταλόγου. Όταν ένας κόμβος ακούει ένα αίτημα συναλλαγής ενός πελάτη, μεταδίδει το αίτημα στους ομότιμους του. Κάθε κόμβος στο δίκτυο επεξεργάζεται κάθε συναλλαγή, καταλήγοντας στα δικά του συμπεράσματα. Όταν συμβαίνει μια επικαιροποίηση του καταλόγου, κάθε κόμβος δημιουργεί μια νέα κατάσταση του καταλόγου με βάση τις συναλλαγές που έχει ακούσει και στη συνέχεια οι κόμβοι ψηφίζουν με συναινετικό αλγόριθμο για το ποιο αντίγραφο είναι σωστό. Μόλις επέλθει ομοφωνία, όλοι οι άλλοι κόμβοι ενημερώνονται με το νέο, σωστό, αντίγραφο του καταλόγου. Οι κατανεμημένοι κατάλογοι χρησιμοποιούν πρωτόκολλα blockchain ως ένα κύριο μέσο υλοποίησής τους.

Στις μέρες μας, τα blockchain και οι κατανεμημένοι κατάλογοι προσελκύουν έντονο ενδιαφέρον και έχουν αποτελέσει έναυσμα για πληθώρα project σε διαφορετικούς τομείς της βιομηχανίας. Εντούτοις, η οικονομική βιομηχανία είναι ο βασικός χρήστης της ιδέας του blockchain. Αυτό συμβαίνει λόγω του ότι η συγκεκριμένη τεχνολογία αρχικά χρησιμοποιήθηκε για να επαληθεύσει συναλλαγές ψηφιακών νομισμάτων. Η πιο διάσημη εφαρμογή αυτής της τεχνολογίας είναι η λογιστική μέθοδος για το εικονικό νόμισμα, το Bitcoin. Το Bitcoin είναι ένα είδος νομίσματος το οποίο δεν ελέγχεται από καμία κυβέρνηση ή τράπεζα. Αντ' αυτού οι συναλλαγές επαληθεύονται από τους χρήστες του, εξαλείφοντας την ανάγκη για κάποια τρίτη οντότητα η οποία θα επεξεργάζεται ή θα αποθηκεύει τις πληρωμές.

Ωστόσο, εκτός από τα οφέλη της, η τεχνολογία blockchain έχει μερικά σημαντικά μειονεκτήματα: χαμηλή διεκπεραιωτική ικανότητα, μεγάλη καθυστέρηση, θέματα ασφάλειας και ιδιωτικότητας, υψηλή κατανάλωση ενέργειας και αντιμετώπιση του συνεχώς αυξανόμενου όγκου δεδομένων του καταλόγου. Αυτό ουσιαστικά σημαίνει ότι τα blockchains αντιμετωπίζουν προβλήματα κλιμάκωσης. Ορισμένες αξιοσημείωτες λύσεις για το πρόβλημα της κλιμάκωσης περιλαμβάνουν:

- Η χρήση διαφορετικών αλγορίθμων οι οποίοι καταλήγουν στην ομοφωνία μεταξύ των κόμβων (proof-of-work, proof-of-stake, byzantine agreement).
- Η αντικατάσταση της υποκείμενης αλυσιδωτής δομής των μπλοκ με κατευθυνόμενο άκυκλο γράφο.
- Η χρήση ενός δικτύου πληρωμών δεύτερου επιπέδου όπου οι εφαρμογές δεν θα χρειάζονται να τοποθετούν κάθε συναλλαγή στο blockchain. Αντ' αυτού, το blockchain θα χρησιμοποιείται σαν ένα σύστημα επίλυσης των διενέξεων μεταξύ των συμβαλλόμενων μελών.

Σκοπός της παρούσας διπλωματικής εργασίας είναι η μελέτη και η ανάλυση της επαναστατικής τεχνολογίας blockchain και η πρόταση μίας νέας προσέγγισης στην υλοποίησή της. Συγκεκριμένα, εξετάζεται η ιδέα της οργάνωσης των ομότιμων του blockchain σε DHT με σκοπό την αποθηκεύση των δεδομένων του blockchain: Καθώς τα blockchain γερνάνε, το μέγεθός τους γίνεται ολοένα και μεγαλύτερο. Αυτή τη στιγμή, το Bitcoin blockchain υπερβαίνει τα 190GB. Επιπλέον, εφόσον εφαρμοστούν ορισμένες από τις παραπάνω ιδέες που αφορούν την αύξηση του ρυθμού των συναλλαγών, το μέγεθος του blockchain θα αυξάνεται ακόμα πιο γρήγορα. Το συνεχώς αυξανόμενο μέγεθος του blockchain είναι ένας παράγοντας που καθιστά το σύστημα περισσότερο συγκεντρωτικό, αφού πολύ λίγοι από τους κόμβους θα είναι σε θέση να διατηρήσουν ένα πλήρες αντίγραφο των δεδομένων. Εξάλλου, οι αρχειακοί κόμβοι με ανοιχτές θύρες έχουν ελάχιστα ή καθόλου κίνητρα για να σπαταλήσουν αποθηκευτικό χώρο και εύρος ζώνης, προκειμένου να διατηρήσουν όλα τα δεδομένα και να εξυπηρετήσουν έναν νέο κόμβο κατά την εκκίνησή του. Συνεπώς, είναι λογικό να θέλουμε να μετριάζεται το φορτίο τους (ειδικά αυτό που ονομάζουμε ”κρύα δεδομένα”, μερικά ιστορικά μπλοκ), μοιράζοντάς το σε ένα DHT. Η αξιολόγηση αυτής της πρότασης επιτυγχάνεται μέσω προσομοιώσεων του πρώτου και πιο δημοφιλούς δικτύου blockchain, του Bitcoin: Δημιουργούμε ένα απλοποιημένο DHT που θυμίζει το Chord. Ρυθμίζουμε το σύστημά μας και δημιουργούμε ένα σύστημα του οποίου τα χαρακτηριστικά είναι παρόμοια με το Bitcoin. Μετράμε διάφορες πτυχές του συστήματος μας όπως εξοικονόμηση χωρητικότητας των κόμβων συγκριτικά με την κλασική υλοποίηση, load balance, query latency, bandwidth usage. Παρουσιάζουμε τα αποτελέσματά μας, τα σχολιάζουμε, και καταλήγουμε σε συμπεράσματα.

ΘΕΜΑΤΙΚΗ ΠΕΡΙΟΧΗ: Blockchains, Κατανεμημένα Συστήματα, Δίκτυα ομότιμων χρηστών

ΛΕΞΕΙΣ ΚΛΕΙΔΙΑ: Blockchain, Bitcoin, Κατανεμημένος Πίνακας Κατακερματισμού, Chord, Κλιμακωσιμότητα, Αποθήκευση, Προσομοίωση, Μαρκοβιανή Αλυσίδα Μόντε Κάρλο

This master thesis is dedicated to my mother

ACKNOWLEDGEMENTS

I would like to thank my supervisor, Prof. Mema Roussopoulos, for the chance she gave me to work on this research project, and for the cooperation and the insightful discussions we had during my studies in this graduate program.

Also, I would like to thank my parents for all this patience and the support they saw to me all these years.

Evangelos Kolyvas
February 2019

CONTENTS

1 INTRODUCTION **1**

2 BITCOIN **3**
2.1 Bitcoin Transactions 3
2.2 Bitcoin Blocks 5
2.3 Bitcoin Blockchain 6
2.4 Blockchain Forks 7
2.5 Bitcoin Clients 8
2.6 Decentralization 9
2.7 Bitcoin Network 11

3 BITCOIN SCALABILITY **13**
3.1 Scalability Problem 13
3.2 Scaling Suggestions 16
3.2.1 Sharding 17
3.2.2 Private Channels 18
3.2.3 Consensus Algorithms 19
3.2.4 Directed Acyclic Graphs 22
3.3 What's the novelty of this master thesis 22

4 PEER-TO-PEER NETWORKS **23**
4.1 Peer-to-peer Architecture 23
4.1.1 Unstructured Peer-to-peer Networks 23
4.1.2 Structured Peer-to-peer Networks 24
4.2 Distributed Hash Tables 24
4.2.1 Properties 24
4.2.2 Structure 25
4.3 Chord 25
4.3.1 Query Key 26
4.3.2 Node Join 26

5 BITCOIN BLOCKCHAIN IN CHORD DHT **28**
5.1 DHT-Distributed-Blockchain Base Protocol 29

5.2 Simulation Parameters 33
5.3 Simulations 39
5.4 Evaluation of the DHT Blockchain 44

6 CONCLUSION 45
6.1 Concluding Remarks 45
6.2 Future Work 45

REFERENCES 46

LIST OF FIGURES

Figure 1: Transactions inputs and outputs 4
Figure 2: Block interval since 2010 6
Figure 3: An example of blockchain which consists of a continuous sequence of blocks 7
Figure 4: Blockchain forks 8
Figure 5: The number of reachable nodes during the last 730 days 12
Figure 6: Concentration of reachable Bitcoin nodes around the world 12
Figure 7: Daily Transactions since 2017 14
Figure 8: Blockchain Size 15
Figure 9: UTXO Set Size 16
Figure 10: Global mining energy consumption during the last 2 years 19
Figure 11: Bitcoin mining pools 20
Figure 12: Unstructured P2P network 24
Figure 13: Structured P2P network 24
Figure 14: The ”fingers” of one of the nodes 27
Figure 15: Routing path between nodes A and B 27
Figure 16: A 32-node-DHT split in 4 segments 31
Figure 17: Red’s node block list 31
Figure 18: DHT pointers before and after a node’s departure 32
Figure 19: Bitcoin network latency the last 30 days 35
Figure 20: Ping messages per second received by a full node the last 30 days . 35
Figure 21: Incoming & Outgoing bandwidth usage per second the last 30 days . 36
Figure 22: Average time to check and accept a block the last 30 days 36
Figure 23: Capacity savings per node 39
Figure 24: Blocks per node 40
Figure 25: Minimum departure rate for data loss 41
Figure 26: Query latency 42
Figure 27: Bandwidth usage per node on behalf of the DHT protocol 43

LIST OF TABLES

Table 1: Finger table of the predecessor of a new joining node 33
Table 2: Finger table of the new joining node 33
Table 3: Top 10 countries with their respective number of reachable nodes . . 34
Table 4: Simulation Parameters 38

1. INTRODUCTION

In the information age, we are experiencing a rapid digitization of data and services. The new trend is the digitization of money. This is an idea that started in late 2008 when an unknown person, using the name Satoshi Nakamoto, has published a white paper titled *"Bitcoin: A peer-to-peer electronic cash system"* [13]. This system is based on the pioneering combination of basic cryptography, proof-of-work [7], and peer-to-peer technology [18]. Bitcoin's innovation is the proposition of a fully distributed server that can execute commands on a global payment system. This new technology later became known as "blockchain". Since its implementation in 2009 and its use by some enthusiastic users, the system has grown into a global payment system and one of the most talked-about technological achievements with billions of euros of investment to support it and an entire industry built on its bases.

The features that make Bitcoin attractive as a currency are its decentralized design and the fact that there is no need for trust between the parties. It is an open system where everyone can participate both in making transactions as well as in verification of transactions, and to be paid for his contribution. Bitcoin ledger is not maintained by a trusted central server, but by a distributed network of collaborating volunteers. Transactions are irreversible and can be easily and directly executed between any users, eliminating middle men, without geographical and, for the time being, without legal restrictions.

After Bitcoin, it began a significant technological breakthrough. Blockchain technology was expanded in many applications, from digital coins, to smart contracts [3] and decentralized voting applications.

However, like any new technology, blockchain is called upon to face some challenges, related to its practical application, that will determine its future. In particular, the key bottlenecks are the throughput (transactions per second), the delay for adding new data to the blockchain, and the equipment requirements (computing power and data storage). These parameters are limiting to blockchain technology because their scaling has not been achieved so far.

The purpose of this work is to analyze the blockchain technology, to identify and explain its weaknesses and to present the most interesting proposals studied for the blockchain escalation. Additionally, there is the goal of contributing to the blockchain community with a proposal based on DHT usage, which stores the blockchain data, and looking forward to even more decentralized and affordable blockchains. The assessment of viability and the evaluation of the proposal will be done through simulations.

In this master thesis, there is an extensive reference to Bitcoin, as the first, largest blockchain application with most participating users. Chapter 2 analyzes the Bitcoin design and protocol, features and statistics of its network, and explains the blockchain technology.

Chapter 3 examines the performance and scalability of Bitcoin, the first digital currency to serve global transaction demand. It defines the weaknesses of the blockchain technology and refers to the most important proposals that have been discussed and aimed at its escalation. In addition, we present the proposal of this master thesis, which considers the idea of organizing the blockchain peers into a DHT in order to store the blockchain data, aiming to reduce the required storage per node.

The content of Chapter 4 is about peer-to-peer networks. After a short definition of these networks, reference is made to DHTs, to their properties and structure. At the end of this chapter we talk about Chord, the DHT system that we try to modify in order to store the blockchain data.

Finally, Chapter 5 describes how we build a simplified DHT that reminds Chord. We tune our system and we make a system whose characteristics is similar to Bitcoin. We present our results, we make comments on them, and we draw a conclusion.

2. BITCOIN

Bitcoin is a cryptocurrency and a digital payment system proposed by a developer, or a group of developers, under the name Satoshi Nakamoto. It released as open source software in 2009. Bitcoin is implemented through a peer-to-peer network, and transactions take place directly between users without an intermediary. In this chapter we will give a general description of Bitcoin, as well as some basic technical details. We will focus on the system and its protocol, describing how the blockchain is built and the technology on which it is based.

In Bitcoin, information is propagated through two types: transactions and blocks. Transactions are the basic information units and contain all the necessary information to complete a money transfer, while blocks are groups of transactions and their existence serves to achieve synchronization of status between the peer-to-peer network nodes.

Unlike traditional currencies such as the Euro, Bitcoin is not based on a central authority that controls its supply, the distribution of money and confirms the validity of the transactions. Bitcoin relies on a network of ”volunteers” who maintain faithful copies of the ledger. The ledger contains all the necessary information to infer the balance of each holder of bitcoins. It is crucial to have consistency between the ledgers of the nodes throughout the network, since the validity of a transaction is confirmed by them.

2.1 Bitcoin Transactions

As a concept, a transaction is a transfer of bitcoins from one or more source accounts to one or more destination accounts. In fact, an account is a public/private key pair. The public key is the account address and serves as an identifier. In order to transfer an amount of bitcoins to an account, a transaction is created. The destination address of the transaction is recipient’s public key. To send bitcoins from an account, the transaction should be signed with the private key of the sender’s account. This method is not the only one to transfer bitcoins to an account, but it is the most common. As its description is sufficient for the development of the topic, we will stick with it.

Instead of the classic add-on method for calculating the balance of an account, tracing and summing transactions that transfer bitcoins (outputs) into the account is used. The outputs are actually tuples of a numeric value in bitcoins and a condition. Anyone who can satisfy the cryptographic condition can claim and spend the bitcoins of the output. The balance of an account is derived from the sum of the arithmetic values of all the unspent transaction outputs (UTXO) of the account.

Transactions are determined based on the hash value of their serialized representation. Transactions are composed of inputs and outputs; unspent outputs are the actual bitcoins. A transaction spends outputs by providing a proof of ownership of these. References to claimed outputs along with proofs of ownership compose what is called input in a transaction. In Figure 1 *User A* spends 0.0020 BTC: 0.0005 BTC to *User C*, 0.0013 BTC to himself and 0.0002 BTC as a miner fee. *User B* spends 0.0010 BTC: 0.0003 BTC to *User C*, 0.0006 BTC to himself and 0.0001 BTC as a miner fee. *User C* spends 0.0008 BTC.

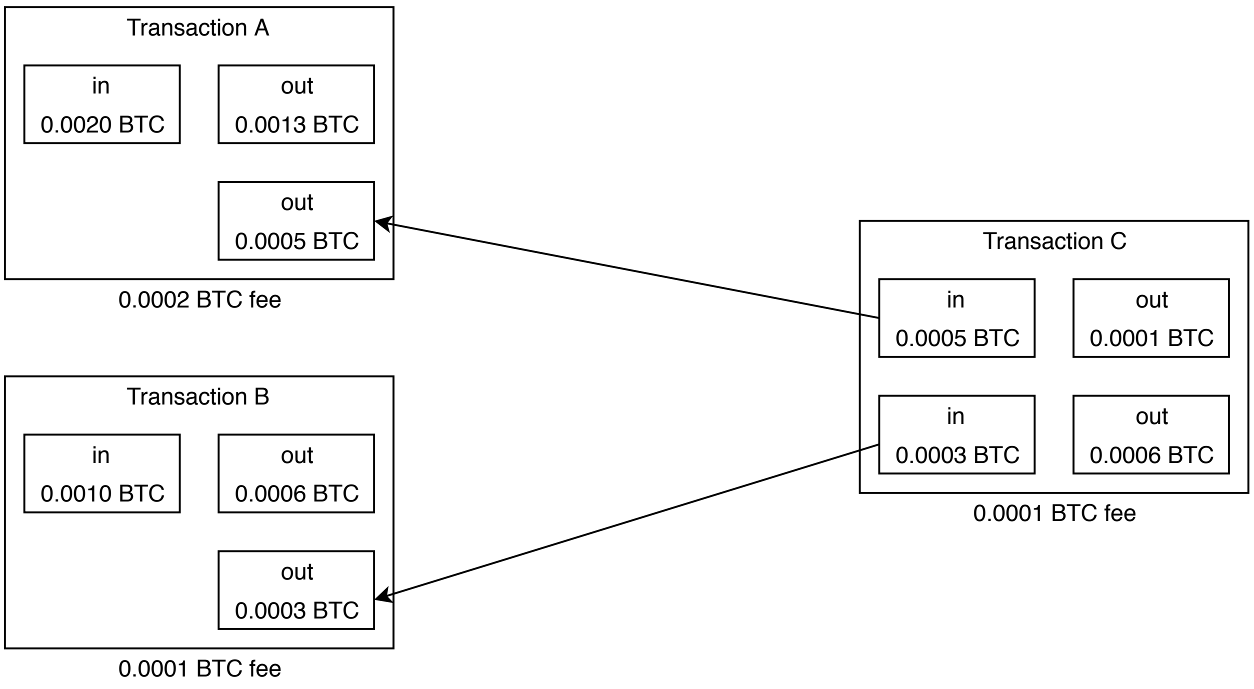

Figure 1: Transactions inputs and outputs

The outputs are the key information units of the ledger and their status should be consistent across all copies. For a transaction to be valid, the following restrictions must be met with respect to the outputs they spend and create:

- An output may be spent at most once.
- New outputs are created only as transaction results.
- The sum of the numerical values of inputs should be equal to or greater than the sum of output values generated.

As new transactions are transmitted to the network, the status of the ledger of each node changes. When a node receives a new transaction, it confirms its validity and includes it in its local copy. It is possible that some instances of inconsistencies occur between the copies of the ledger on different nodes:

- It is possible that a node takes a transaction that transfers an amount from an account without having received the transaction that makes this amount available to the account.
- Two or more transactions can claim the same outputs, trying to spend the respective amount more than once. This is called a double spending attack.

Double spending attacks have a direct effect on the consistency of the ledger copies. When a double spending occurs, deliberately or not, two or more transactions attempt to spend the same output at the same time. A node that receives the first transaction will confirm it and include it in its ledger. Later on, when it receives the rest, their confirmation will fail as the output has already been spent. As there is no guarantee that all nodes will receive these transactions in the same order, the nodes will disagree on the validity of these conflicting transactions and any other transaction that will be based on them by spending their outputs. This mismatch is solved through blocks, as will be discussed in the following section.

2.2 Bitcoin Blocks

In order to maintain consistency between the copies of the ledger, there must be an agreement between the nodes on the order of transactions. Achieving such an agreement is not a trivial issue. Bitcoin solves this problem by firstly testing acceptance of the transactions and then synchronizing at regular intervals by transmitting the blocks created by the nodes. A block b contains a set of transactions T_b, which are the transactions accepted by the authoring node after the immediately preceding block. This block is distributed across all network nodes. Every node that receives the bock, reverses the test transactions that they have accepted and applies those that contain the new block b.

At this point all nodes have agreed on the validity of all transactions within b. b transactions that had already been included in the ledger do not need to be reapplied. The reversed transactions will be revalidated and will be tested again in the new ledger status. Transactions that are no longer valid due to collusion with transactions deposited as part of b will be rejected.

The node that created the block b somehow dictates the change in the status of the ledger from the previous block based on its own perspective. However, the decisions that the authoring node can take is limited. It can not counterfeit a transaction since the public/private key cryptosystem is secure. The creator node can only decide on the order in which the transactions were made and whether and which transactions to include in its block.

To make the selection of the node that will create the next block and dictate the new status of the ledger, the nodes try to find a solution to a **proof-of-work** (PoW) [7] cryptographic problem that has a given degree of difficulty. The proof-of-work is, in fact, the challenge of finding a binary string called nonce, which, in combination with the block header gives a hash value H_b such that it has a required number of zero bits at the beginning of it (target). Because the hash functions are one-way functions, finding a target is done only by exhaustively searching and testing possible nonces, until one with the desired property is found. Thus, finding a nonce that provides a solution to the proof-of-work problem is difficult, however its validation is easily done by computing a result of the hash function. Nonce is part of the block, so the recipients nodes confirm that the creator really solved the proof-of-work problem. The H_b value is also used as block identifier. The target is determined so that the average time of creating a new block from the network in total is 10 minutes and re-adjusted after every 2016 blocks. Figure 2[1] shows block interval since 2010. The average value of block interval for all these years is 553.18 sec.

Nodes that try to find a solution to the proof-of-work are called miners. For miners to have an incentive, the node that creates a block gets a reward in the form of newly formed bitcoins. For example, it can include in the block a transaction that it has no inputs. This reward is only valid if it occurs within the block and is the only exception to the rule that the sum of the inputs should be equal to or greater than the sum of the outputs.

[1]https://charts.bitcoin.com/btc/chart/block-interval

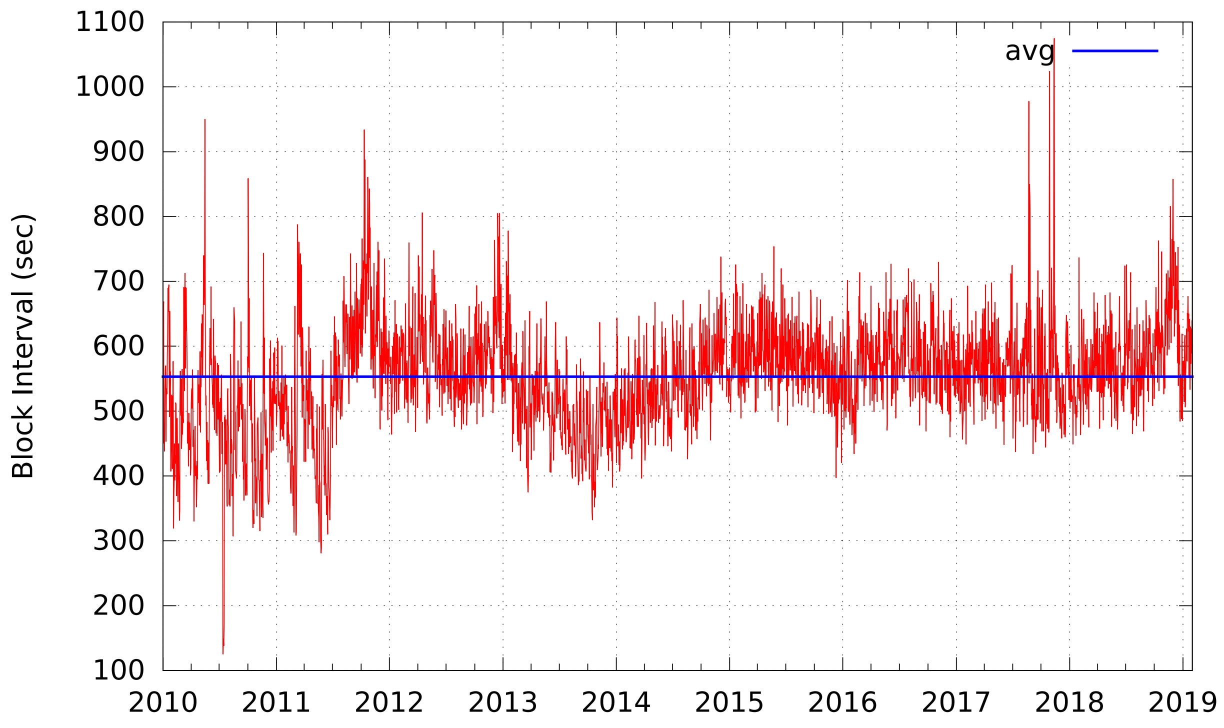

Figure 2: Block interval since 2010

2.3 Bitcoin Blockchain

One way to set the blockchain is as a distributed database that solves the Byzantine Generals Problem and the Sybil Attack Problem. In the Byzantine Generals Problem [11], nodes are asked to agree on the value of a distributed database entry, with the restriction that nodes are likely to fail randomly (including malicious behavior). The Sybil Attack Problem [6] arises when one or more nodes find a way to unfairly disproportionately influence the process of agreeing at the price of an entry. It is the "clone attack" - a number of seemingly independent voters who actually work together to trick the system.

Referring to the previous sections on Bitcoin, there is no block element that offers additional synchronization and serialization in transactions. However, this is achieved when blocks are linked to chain formation, defining a chronological sequence between them and hence between transactions.

Blocks are organized into a directed acyclic graph (DAG). Each block contains a reference to the previous block. When a block b is referred by a block b' as its predecessor, then block b is called parent of b'. The root of this DAG is called genesis block and it is hardcoded to all Bitcoin clients. The genesis block is the ancestor of all blocks. A typical example for a blockchain is illustrated in Figure 3.

Blockchain is defined as the longer path from any block to the genesis block. The distance between block b and the genesis block is reported as the height of block h_b. The genesis block has zero height. The block with the highest height is referred to as the blockchain head and it has height h_{head}.

To make a reference to a block as a parent, the block's ID (its hash value) should be known. So, the child block should be created after the parent block was created. This

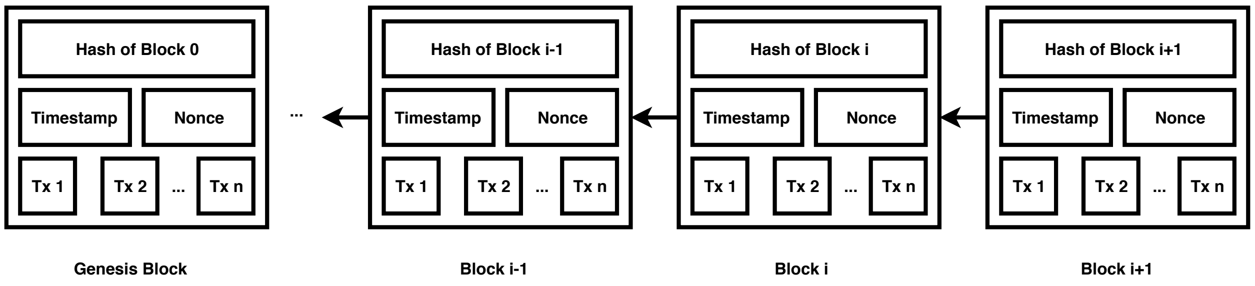

Figure 3: An example of blockchain which consists of a continuous sequence of blocks

chain formation is used to define a chronological order between transactions: transactions in blocks of lower height have been confirmed before these blocks of a higher height.

Only the blocks that are included in the longest branch are rewarded with newly formed coins that are user-acceptable. So miners will attempt to create blocks on the blockchain head. Creating a block in other place than the head, would require this path to get a longer length than h_{head} in order to be rewarded.

Having explained the function of the blockchain, the answer about the usefulness of proof-of-work is obvious: Proof-of-work serves the safety of the blockchain. In order for a block to be considered valid, it must meet the requirements of the target, indicating that time, computational power and energy have been spent on its creation. As blocks are organized into chain formation, falsification of a block implies rebuilding of all its subsequent. In this way, blocks are confirmed: as many blocks follow a certain block in the chain, it is more unlikely for that particular block to get out of the blockchain.

2.4 Blockchain Forks

From the blockchain definition it's possible to have multiple blockchains heads at any time. This is called blockchain fork. In the case of a blockchain fork there is a disagreement between the nodes of the network as to which block is the blockchain head.

When a node, whose blockchain head b_h is at a height h, receives a block $b_{h'}$ with a height h' > h, it defines it as a blockchain head. The new block $b_{h'}$ may belong either to the same tree branch, i.e. $b_{h'}$ to be descendant of b_h, or to belong to a different branch.

If block $b_{h'}$ belongs to the same branch as b_h, then the node will find all the intermediate blocks of the branch and will gradually apply all the changes that they define. Otherwise, where $b_{h'}$ belongs to a different branch, so b_h is not his ancestor and a common ancestor is searched. Once detected, the node will reverse all the necessary changes from block b_h until the common ancestor and it will apply those which defined in the branch of $b_{h'}$.

A blockchain fork can be extended to longer than one block length as a continuation of conflicting blocks. In the end, however, only one branch will prevail by acquiring a longer length than the rest. The nodes supporting different branches will adopt it. At this point the blockchain fork is resolved and all the copies of the ledger get coherent accepting the same blockchain head. The blocks finally discarded by the blockchain are called orphan blocks. Figure 4 illustrates an example: The main chain (blue) consists of the longest branch of blocks from the genesis block (green) to the current block. Orphan blocks (red) exist outside of the main chain.

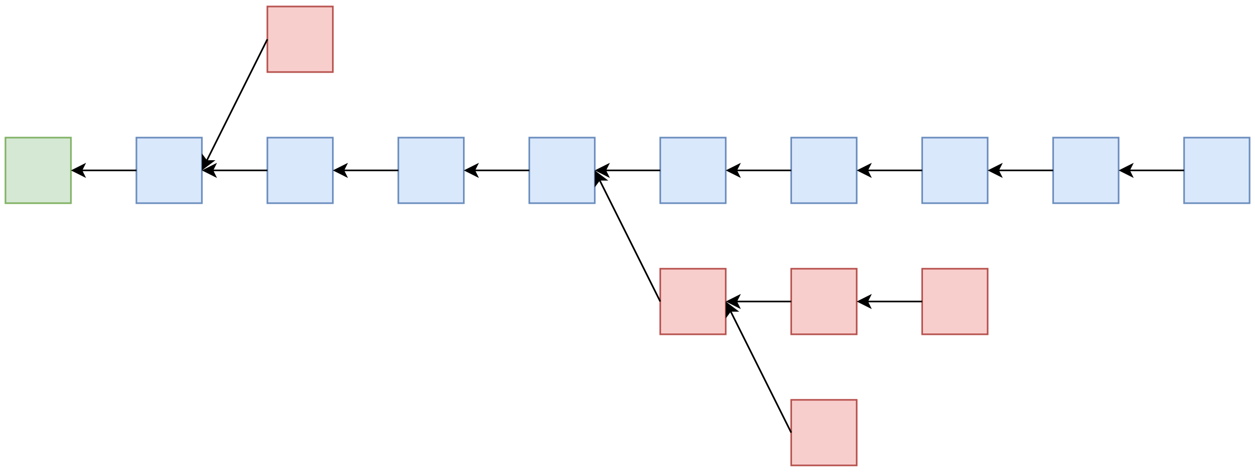

Figure 4: Blockchain forks

For this reason it is obvious that Bitcoin never records a transaction permanently. Every blockchain transaction can be canceled if a longer path is created that does not include it, i.e. it starts from a point in the tree that is earlier than the one containing the transaction. If an attacker wants to reverse a transaction included in block b_h, he should create a new transaction that would counter the original and include it in a block $b_{h'}$, where $h' < h$. Then, the attacker should create blocks by continuing the branch of $b_{h'}$ until it overtakes the original blockchain length. This would be feasible for someone who would control the majority of the hashing power, since it would create blocks at a higher rate than the rest of the network, and could therefore reverse any transaction.

The close relationship between the blocks and the validity of a transaction not only slows down the confirmation time of a transaction, but also limits its confirmation to a probabilistic method.

2.5 Bitcoin Clients

A Bitcoin client stores all the necessary information for a user to deal with bitcoins. Although Bitcoin clients are often referred to as ”bitcoin storage”, they are in fact inseparably linked to the blockchain ledger. A more accurate description of a client would be a means of storing digital credentials for bithoins owned by someone and making it accessible. As Bitcoin uses public-private key cryptography, the content of a client is basically such a key pair.

In addition to storing the necessary credentials to confirm the ownership of their bitcoins, Bitcoin clients are connected to the network and allow for transactions. Bitcoin clients are divided into two main categories:

- **Full nodes**, which are nodes that confirm transactions by maintaining a local copy of the blockchain. Full nodes maintain a complete and up-to-date copy of the Bitcoin blockchain with all the transactions, which they independently build and verify, starting with the very first block (genesis block) and building up to the latest known block in the network. A full node can independently and authoritatively verify any transaction without recourse or reliance on any other node or source of information. Due to the size of the blockchain (190GB, February 2019) and its processing requirements, most of the nodes can not support such a function, so they retain a subset of the blockchain. These nodes are synchronized to Bitcoin blockchain as they download the whole blockchain at the begining, in order to be sure of the blockchain's correctness. Once the validity of a block is confirmed, its content is unnecessary.

- Not all nodes have the ability to store the full blockchain. Many bitcoin clients are designed to run on space- and power-constrained devices, such as smartphones, tablets, or embedded systems. For such devices, a simplified payment verification (SPV) method is used to allow them to operate without storing the full blockchain. These types of clients are called **Lightweight clients** or **Simple Payment Verification clients** (SPV clients). SPV nodes download only the block headers and do not download the transactions included in each block. The resulting chain of blocks, without transactions, is 1,000 times smaller than the full blockchain. SPV nodes cannot construct a full picture of all the UTXOs that are available for spending because they do not know about all the transactions on the network. SPV nodes verify transactions using a slightly different methodology that relies on peers to provide partial views of relevant parts of the blockchain on demand. SPV nodes consult full nodes to confirm transactions as they do not keep a copy of the blockchain. Thus, by making use of a lightweight client, the user should trust his provider, who can not steal bitcoins, but may provide false indications / affirmations. This user can have strong evidence for his transactions, but he can never be sure of them, as the only way for absolute certainty is to keep a local copy of the blockchain (full node).

2.6 Decentralization

The most important parameters of Bitcoin at this time are trust, security and privacy. Full nodes are able to check that all of Bitcoin's rules are being followed. Rules such as the inflation schedule, no spending the same coin twice, no spending of coins that don't belong to the holder of the private key and all the other rules required to make Bitcoin to work. Full nodes are the ones that give Bitcoin users the ability to be sure about the proper operation of the system, without trusting anyone else (trustless). There is no need to trust in any financial institution (bank, paypal). Everyone knows the proper functioning and integrity of the blockchain he maintains on his personal computer and that is enough. Whether each Bitcoin user will retain his own full node to use it as a wallet, is at his discretion.

Trust: Running a full node and using it as a wallet is the only way to know with certainty that no Bitcoin rule has been violated. Any other type of wallet requires trust in an intermediary server. Thus, through personal full node is the only way to verify the bitcoins that each person holds, while any third party information contains a dose of uncertainty.

Security: All validity checks by full nodes increase the security. There are many attacks against lightweight client wallets that do not affect full node wallets.

Privacy: Full node wallets are so far the best way to keep one's privacy, as no one has the information to link the Bitcoin public key to the person who owns the private key. Each use of lightweight wallet leaks information about which addresses are assigned to the individual user as they send requests to intermediate servers. Maintaining a full node is the only way to keep your privacy.

Governments are good at cutting off the heads of a centrally controlled networks like Napster, but pure P2P networks like Gnutella and Tor seem to be holding their own.

Satoshi Nakamoto [12]

In order for a Bitcoin user to obtain information about the blockchain, e.g. for a transaction that concerns him, he either has to check himself in the blockchain, or trust a third party. The fact of trust in a third person reduces locality, i.e. cancels the essence of a peer-to-peer existence. In order to maintain a peer-to-peer network, users should participate as peers, which for Bitcoin means to maintain a full node.

The cost of maintaining a full node is high. However, in a system where the counterfeit risk is always measurable, a full node is the only means of confirming that the bitcoins are correctly distributed, strictly following all the rules defined by the Bitcoin protocol.

The only node that matters is the one you use.

Peter Todd [21]

The significance of the above-mentioned phrase, that one can use his own information (a blockchain copy) to prove the validity of a transaction, leads to a critical conclusion: Bitcoin’s degree of decentralization is directly determined by the number of full nodes, so in essence, by the cost required to maintain a full node.

If we think about it, one extreme is that this cost would be zero. Then anyone would be able to confirm whether or not a transaction took place and, by extension, the whole Bitcoin blockchain about its correctness, without relying on a third party. At the other end, if the cost of a full node was so high that there was only one node, then the administrator of that node would control the entire network: complete centralization.

2.7 Bitcoin Network

The precise number of Bitcoin's full nodes is unknown. All measurements are referred to the available full nodes, i.e. those with open ports that are accessible. However, many nodes are behind firewalls or configured to prevent connection requests. The main reason for such a decision is probably due to the use of the network, since network bandwidth costs. Thus, nobody knows the exact number of full nodes and it is very likely that nodes with closed ports are several thousands. In other words, just because open-port-nodes can only be measured and closed-port-nodes cannot, some members of the Bitcoin community have been mistaken into believing that open-port-nodes represent the full count.

According to the measurements [2], there are 10,500 available full nodes. Assuming they are all using default Bitcoin Core settings, they will each provide 117 TCP/IP connection slots to the network (125 available minus 8 for their own use). SPV nodes typically use 4 connection slots and full nodes typically use 8. So the network can support up to 1,228,500 connections, translated to a maximum of around 153,562 non-listening full nodes, or 307,125 SPV nodes at one time. This is roughly the upper limit for the number of wallets that are online and connected to the Bitcoin network at any one time. (If there were more people online at once than that, people would start seeing various issues.) This doesn't include wallets that don't actually connect to the Bitcoin network, of course.

According to the metrics [19] on a long-running listening full node, the TCP/IP incoming connections are averaged: 110 with other full nodes and 15 with SPV nodes. Given the 110 TCP/IP connections for uploading to full nodes, and the 15 TCP/IP connections with the SPV nodes, we can rough guess that there are at this moment (10,500*110)/8 = 144,375 full nodes (10,500 listening) and (10,500*15)/4 = 39,375 SPV nodes connected to the network.

Nodes with open ports are useful to the Bitcoin network as they help bootstrap new nodes by uploading historical blocks - they are a measure of the number of redundant copies of the blockchain available for synchronizing with. For the time being, there has been no shortage of bandwidth capacity for simply syncing wallets from the available nodes. Bitcoin community thinks that only trust, security, and privacy are what matters right now. Therefore, if there were shortage of bandwidth, they think that bandwidth might be added by renting cloud servers, but that would lead to a even more centralized system. Figure 5[2] shows the number of reachable nodes during the last 730 days (2 years), from Feb 11, 2017 (5873 nodes), to Feb 11, 2019 (10510 nodes). Figure 6[3] shows the concentration of reachable Bitcoin nodes found in countries around the world.

[2]https://bitnodes.earn.com/dashboard/?days=730

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

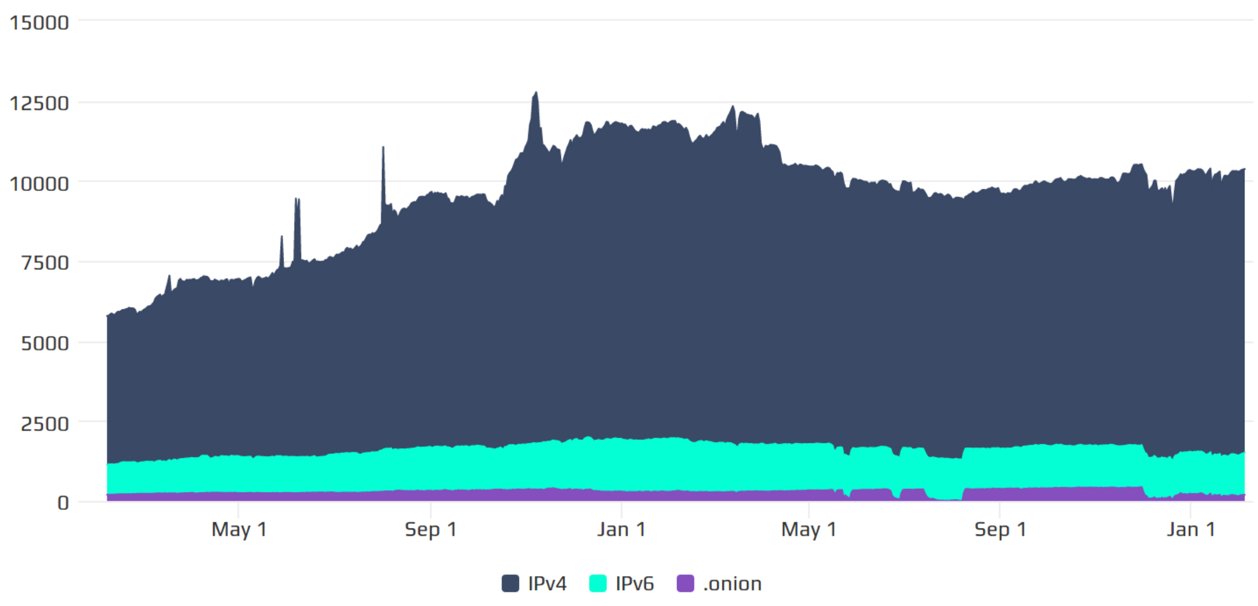

Figure 5: The number of reachable nodes during the last 730 days

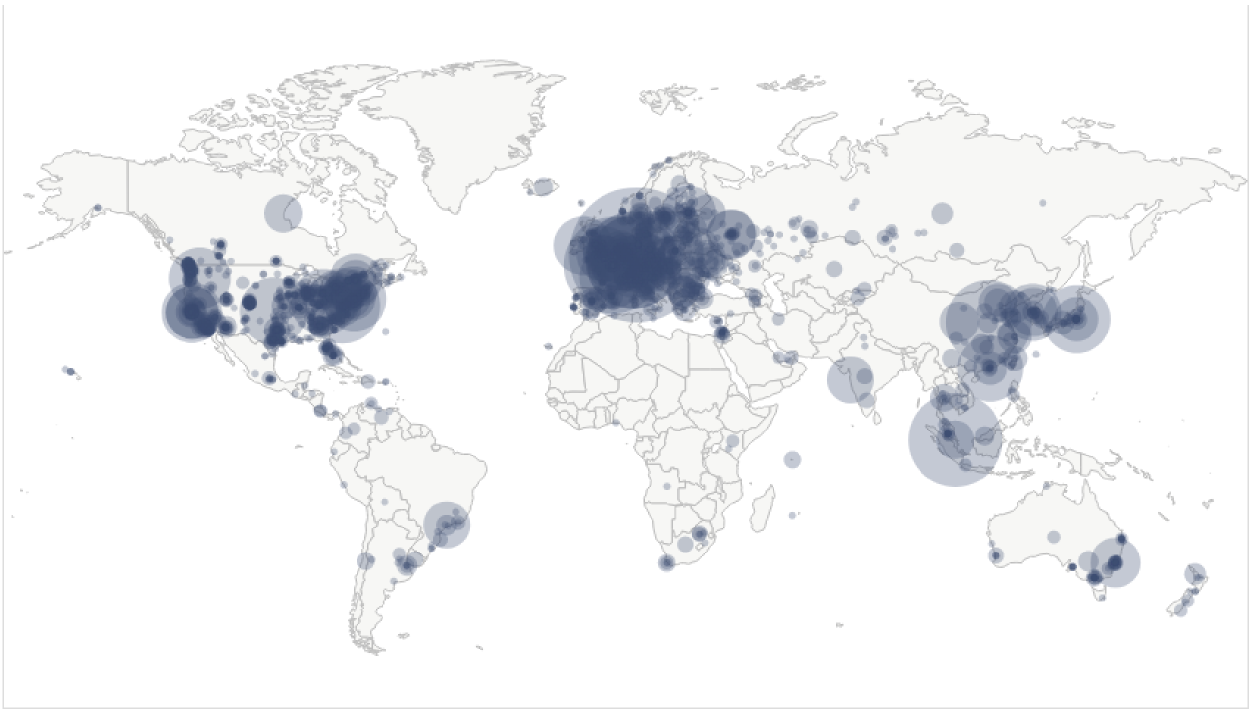

Figure 6: Concentration of reachable Bitcoin nodes around the world

3. BITCOIN SCALABILITY

3.1 Scalability Problem

The ever-growing adoption of some cryptocurrencies as a means of payment has begun to raise concerns about the ability to scale their technology. Since Bitcoin is a self-managed system which creates blocks at approximately constant time intervals, the maximum transaction rate is limited and is equal to the ratio of the maximum number of transactions that a block may contain divided by the block interval.

Cryptocurrency community continuously discuss how to improve blockchain scalability, with many interesting suggestions, but no proposal seems to be capable of solving the problem in its entirety. For example, in Bitcoin, **Segregated Witness** (SegWit) [23] which was activated in August 2017. SegWit suggests to segregate the digital signature from the transactions data. Digital signature accounts for 65% of the space in a given transaction. SegWit attempts to ignore the data attached to a signature by stripping off the signature from within the input and moving it to a structure towards the end of a transaction. This would increase the 1 MB limit for block sizes to a little under 4 MB.

Of course, SegWit, as well as supporters, has also faced some critics who rejected the update decided by the vast majority of the Bitcoin community. Although the protocol upgrade was adopted by the 96.49% of the miners (by hash power), a small group of mostly China-based Bitcoin miners - programmers refused to follow and set up their own new cryptocurrency, called Bitcoin Cash [9]. Today (Feb 5, 2019), 99.10% [5] of the Bitcoin nodes, are Bitcoin Core nodes. This example of partitioning between the community is indicative of the inability to define a clear strategy to tackle the problem of blockchain scalability. Note that SegWit may allow Bitcoin to increase its transaction rate, but it is not a long-term or medium-term solution to Bitcoin scalability problem.

In order to have an overview of Bitcoin's current capabilities, let's calculate the number of transactions per second: Figure 7[4] shows the number of daily transactions since 2017 (the last ~2 years). The average value for this time period is 255820 transactions per day. That means 255820 (txs/day) / 86400 (sec/day) = 2.96 transactions per sec (in practice). In theory this number is a little bit higher (~7 transactions per second [4]), while a transaction typically takes several minutes or even hours to be confirmed. In comparison, a mainstream payment processor such as Visa credit card confirms a transaction within seconds, handles on average around 2,000 transactions per second (tps) and it has a peak capacity of around 56,000 transactions per second [22]. Clearly, a large gap exists between where Bitcoin is today, and the scalability of a mainstream payment processor.

Throughput (tps) can theoretically be parameterized, either by increasing the block size limit, or by reducing the block interval. But both of these solutions pose risks to blockchain consensus and transaction security. In particular, block interval reduction means easier / faster creation of new blocks, and impairment of proof-of-work security, as explained in Sections 2.3 and 2.4. In addition, increasing the block size will lead to delay in the spread of new blocks, i.e. lower levels of consensus, as the effective throughput decreases. Effective throughput refers to the percentage of nodes that are synchronized

[4]https://charts.bitcoin.com/btc/chart/daily-transactions

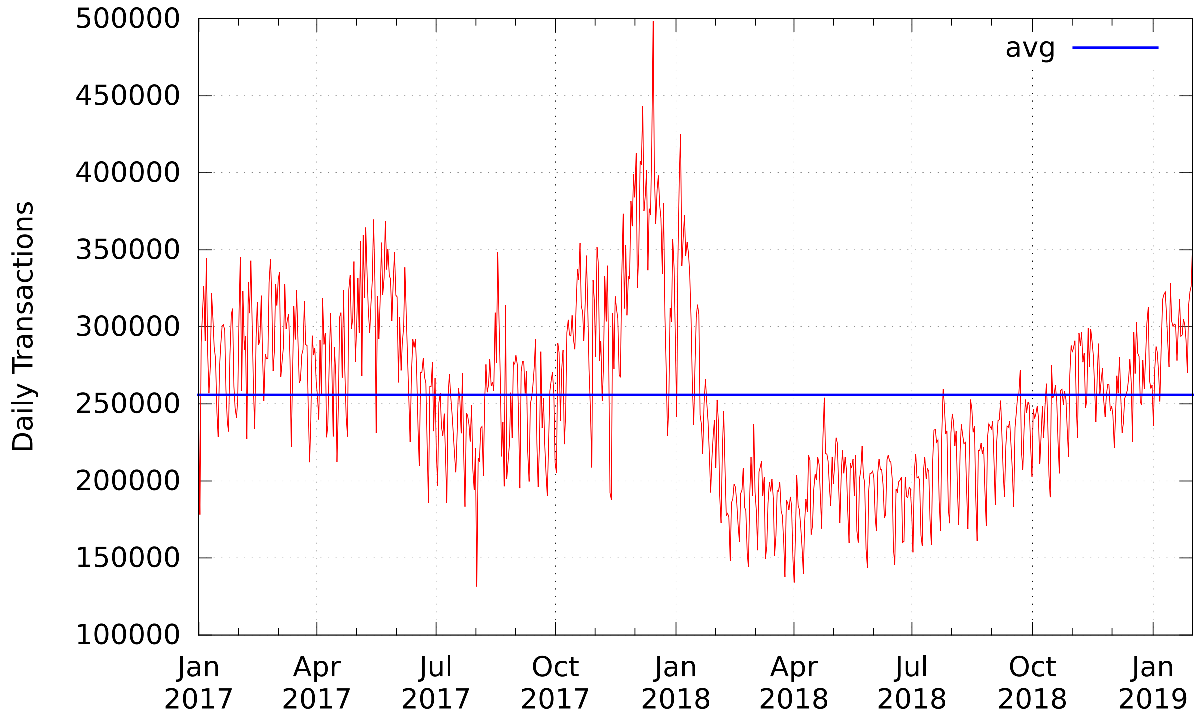

Figure 7: Daily Transactions since 2017

with the blockchain, despite the network delays and the speeds supported by their connections. Therefore, the transaction rate is a parameter that is difficult to customize based on Bitcoin's current state.

A second point in which Bitcoin is inferior to other payment methods is **latency** (confirmation time of a single transaction). A transaction that has been included in the blockchain, theoretically, is never 100% assured (permanent). This is explained in detail in Section 2.4 on blockchain forks. However, the "deeper" a block is, in terms of blockchain's height, the less likely is to reject it. By convention, a block is considered finalized in the blockchain after it has succeeded by at least 6 blocks. Thus, a transaction requires at least one hour to be considered finalized, without counting the time it takes to deposit it into a blockchain block.

An additional constraint parameter is that of **storage capacity**. Currently (February 2019), the size of the Bitcoin blockchain is 190GB and exhibits an exponential increase in time (Figure 8[5]). This is due to the increase in the use of Bitcoin as a means of payment; stabilizing transactions rate in the unit of time will also entail stabilizing the growth rate of the blockchain.

To clear things up, it is not necessary for a Bitcoin node to keep the whole blockchain file on its disk. Satoshi's white paper [13] describes the process of "pruning", in which unnecessary data on fully spent transactions is deleted. This reduces the volume of required data for a node that performs a transaction verification. All the unspent transactions are called Unspent Transaction Output set (UTXO-set). In case of Bitcoin, they are about 64MB (February 2019, Figure 9[6]) and is all the information that one verifier needs. Keep-

[5]https://charts.bitcoin.com/btc/chart/blockchain-size

[6]https://charts.bitcoin.com/btc/chart/utxo-set-size

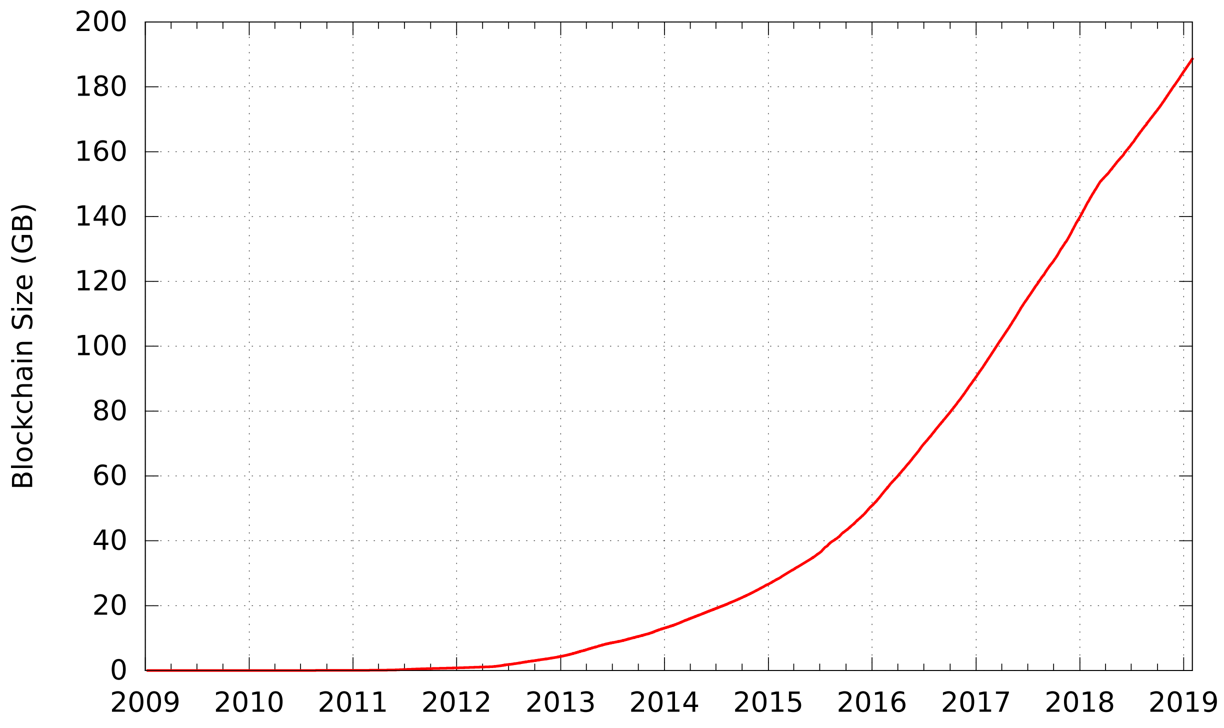

Figure 8: Blockchain Size

ing the whole blockchain may not be required for the mining or verifying operations, but, in general, its preservation is necessary for the operation of a blockchain. One reason is that in a blockchain, by definition, it should be possible to prove the validity of any information by providing its entire history since the genesis block. In addition, the blockchain file, as a whole, needs for bootstrapping new nodes, a process that repeats the whole blockchain history in order for the new node to verify the correctness of the current state of the blockchain, i.e. the UTXO-set.

To sum up, although the entire blockchain is not necessary for every validating node, its maintenance is essential. A number of archival nodes need to store the full chain going back to the genesis block. These nodes can be used to bootstrap new validating nodes from scratch. On the other hand, while the size of the UTXO-set is less than 100MB, which is small enough to easily fit in RAM for even quite old computers, is not a problem.

Today (February 2019), the minimum requirements [1] to run a Bitcoin full node are:

- Desktop or laptop hardware running recent versions of Windows, Mac OS X, or Linux.
- 200 gigabytes of free disk space, accessible at a minimum read/write speed of 100 MB/s.
- 2 gigabytes of memory (RAM)
- A broadband Internet connection with upload speeds of at least 400 kilobits (50 kilobytes) per second
- An unmetered connection, a connection with high upload limits, or a connection you regularly monitor to ensure it doesn't exceed its upload limits. It's common for full

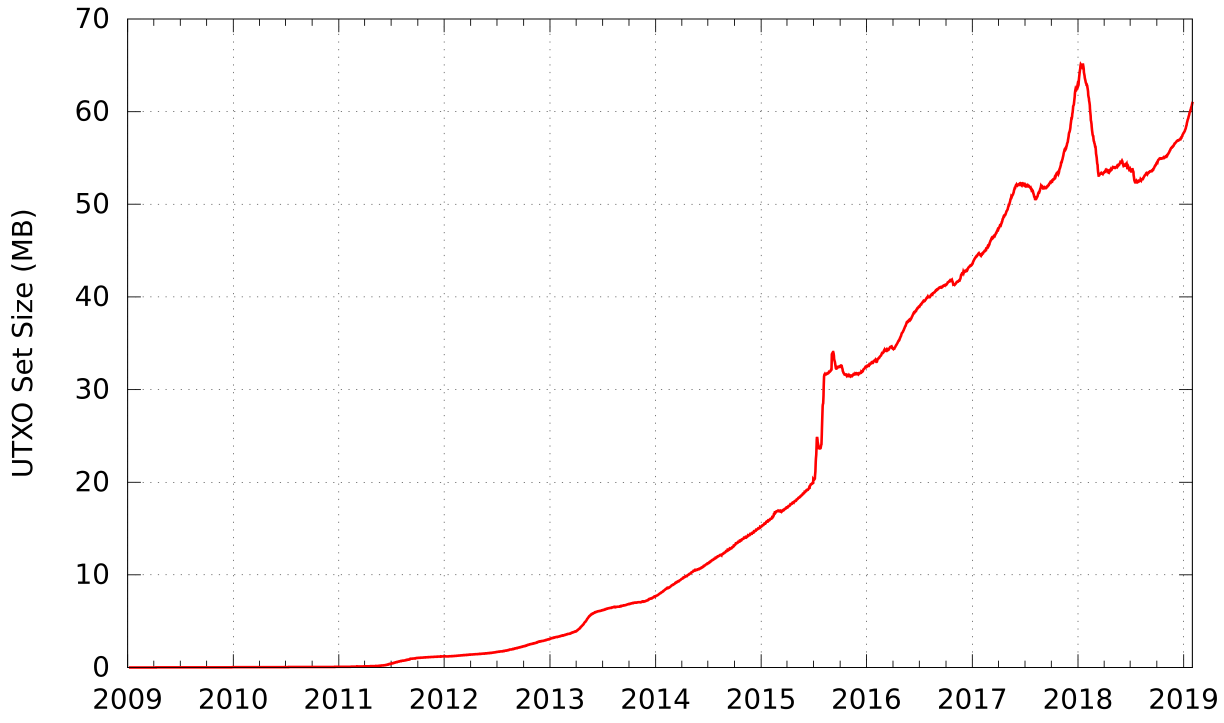

Figure 9: UTXO Set Size

nodes on high-speed connections to use 200 gigabytes upload or more a month. Download usage is around 20 gigabytes a month, plus around an additional 195 gigabytes the first time you start your node.

- 6 hours a day that your full node can be left running. (You can do other things with your computer while running a full node.) More hours would be better, and best of all would be if you can run your node continuously.

3.2 Scaling Suggestions

Before delving into the scaling solutions proposed by the community, it is important to understand the tradeoffs (or Scalability Trilemma) that comes along when we talk about scalability.

There are generally 3 main attributes that must be considered when developing block-chains:

1. **Security**: A distributed network should be resistant to a wide variety of attacks or hacks such as 51% attacks, Distributed Denial-of-Service (DDoS) attacks and Sybil attacks. More importantly, blockchains should be fault-tolerant, meaning that the system will continue to operate even if a component of the network fails.
2. **Decentralization**: Perhaps the core tenant of Blockchain technology, open source decentralization allows for a censorship-resistant, inclusive network that enables anyone to participate without prejudice.

3. **Scalability**: This refers to the capacity of blockchains in processing transactions in the network. A scalable system is able to cater to more transaction and activity in the network without suffering from network stress.

A blockchain can only choose 2 attributes and perhaps sacrifice an attribute. For instance, Bitcoin and Ethereum [3] were designed to focus on decentralization and network security: In both blockchain protocols each node stores the entire state and processes all transactions. This provides a large amount of security, but greatly limits scalability: a blockchain cannot process more transactions than a single node can. In large part because of this, Bitcoin is limited to 3–7 transactions per second and Ethereum to 7–15.

However, this poses a question: Are there ways to create a new mechanism, where only a small subset of nodes verifies each transaction? As long as there are sufficiently many nodes verifying each transaction that the system is still highly secure, but a sufficiently small percentage of the total validator set that the system can process many transactions in parallel, could we not split up transaction processing between smaller groups of nodes to greatly increase a blockchain's total throughput?

3.2.1 Sharding

The basic idea behind **sharding** is to split the state and history of a blockchain up into $K = O(N/c)$ partitions that we call "shards". For example, a sharding scheme might put all addresses starting with 0x00 into one shard, all addresses starting with 0x01 into another shard, etc. In the simplest form of sharding, each shard also has its own transaction history, and the effect of transactions in some shard k are limited to the state of shard k. One simple example would be a multi-asset blockchain, where there are K shards and each shard stores the balances and processes the transactions associated with one particular asset. In more advanced forms of sharding, some form of cross-shard communication capability, where transactions on one shard can trigger events on other shards, is also included.

Sharding is proposed to be implemented once Ethereum moves to a Proof-of-Stake model. A basic design of a sharded blockchain is as follows:

There exists a set of validator nodes, who randomly get assigned the right to create shard blocks. During each time slot, for each shard a random validator gets selected, and given the right to create a block on that shard. Also, for each shard, a set of validators get selected as attesters. The header of a block together with at least 2/3 of the attesting signatures can be published as an object that gets included in the ”main chain” (also called a beacon chain).

3.2.2 Private Channels

Bitcoin transactions can get slow and expensive. Transactions are confirmed only once every 10 minutes on average and if you didn't attach a high enough fee, it could take even days to get confirmed. Unfortunately, this becomes a real issue with day-to-day micro transactions. One of the main arguments of why Bitcoin can't be used as a medium of exchange is due to how slowly it works and how expensive it is to send small payments through the network.

So, wouldn't be great if we could have instant and "feeless" transactions? This is where the **Lightning Network** [15] comes in. The key idea behind the Lightning Network is that small, everyday transactions don't have to be stored on the main blockchain. This avoids the 7 transactions per second limit and is also called the off-chain approach. Let's take a look at an example to understand how does it work:

Every morning Bob buys a cup of coffee on his way to work. Creating a transaction on the blockchain for a simple coffee is really overkill. He might end up paying more fees than the actual price of his coffee. However, with the Lightning Network Bob can setup a payment channel with the coffee shop. To do that, both the coffee shop and Bob deposit a certain amount of Bitcoin in what is called a multi-signature address. Let's assume that Bob deposits 0.05 BTC and the coffee shop deposits nothing, because they don't offer refunds. This multi-signature address is basically like a safe that can only be opened when both parties agree. When we open the payment channel we also make a balance sheet that says how the funds in the address should be distributed. So right now it says: *"Bob will get 0.05 BTC and the coffee shop will get 0 BTC"*, the same as they deposited. Opening the payment channel happens on the main blockchain so that there is full transparency. The coffee shop owner can see that Bob has deposited 0.05 BTC and they can rest assured that they will get their money once the channel closes.

Now that the channel is open, Bob can order his morning coffee. Let's say that a coffee costs 0.001 BTC. To pay for it, Bob simply changes the balance sheet. He subtracts the cost of the coffee from his balance, and adds it to the coffee shop's balance. So now it says: *"Bob will get 0.049 BTC and the coffee shop will get 0.001 BTC"*. Bob and the coffee shop now sign the updated balance sheet with their private keys, they then each keep a copy of it, but they don't do anything else with it. Bob can keep ordering coffees for as long as he has a balance in the payment channel. Both of them can make hundreds of thousands of transactions between them. There is really no limit because this happens away from the main blockchain.

The payment channel can be closed at anytime by either Bob or the coffee shop. All they have to do is take the latest balance sheet, which was signed by both parties and broadcast it to the Bitcoin network. Miners will then validate the signatures on the balance sheet and - if everything checks out - release the funds according to the balance sheet. This will create a single transaction on the Bitcoin blockchain. So the Lightning Network can significantly reduce the load on the main blockchain. It only requires two transactions on the blockchain: one to open the payment channel and another one to close it.

Moreover, in the Lightning Network you don't need to open a direct payment channel with everyone you want to send bitcoins to. You can simply use the network to pass your coins around. For example, suppose Alice has exchanged money on the Lightning Network with Bob before, so they have an active payment channel. Now let's say that Alice wants to buy a coffee. Instead of opening a direct channel to the coffee shop, she can transfer the money to Bob, who will then transfer it to the coffee shop. So there is no need to create a payment channel with everyone. In the Lightning Network, your payment tries to find a route from person A to person B and it tries to do this with the least amount of intermediates and the least amount of fees. This further reduces the strain on the blockchain but it requires the intermediates to have enough money in the payment channels.

Ethereum's equivalent to Bitcoin's Lightning Network is called **Raiden Network** [17].

3.2.3 Consensus Algorithms

One of the main drawbacks of cryptocurrencies, like Bitcoin, is that they use enormous amounts of energy to secure their networks. This is because mining new coins takes a lot of computing power due to the proof-of-work algorithm. As described in Section 2.2, the proof-of-work algorithm works by having all nodes solve a cryptographic puzzle. The only way to solve this puzzle is by exhaustively searching and testing all possible solutions, consuming huge amount of electric power. Today (February 2019), Bitcoin miners alone use about 47.100TWh of electricity per year, enough to power the entire country of Greece for about 11 months. While, in its peak period (August - November 2018), Bitcoin miners were using 73.121TWh per year, enough to power Greece for about 17 months [8]. Figure 10[7] shows the global mining energy consumption during the last 2 years.

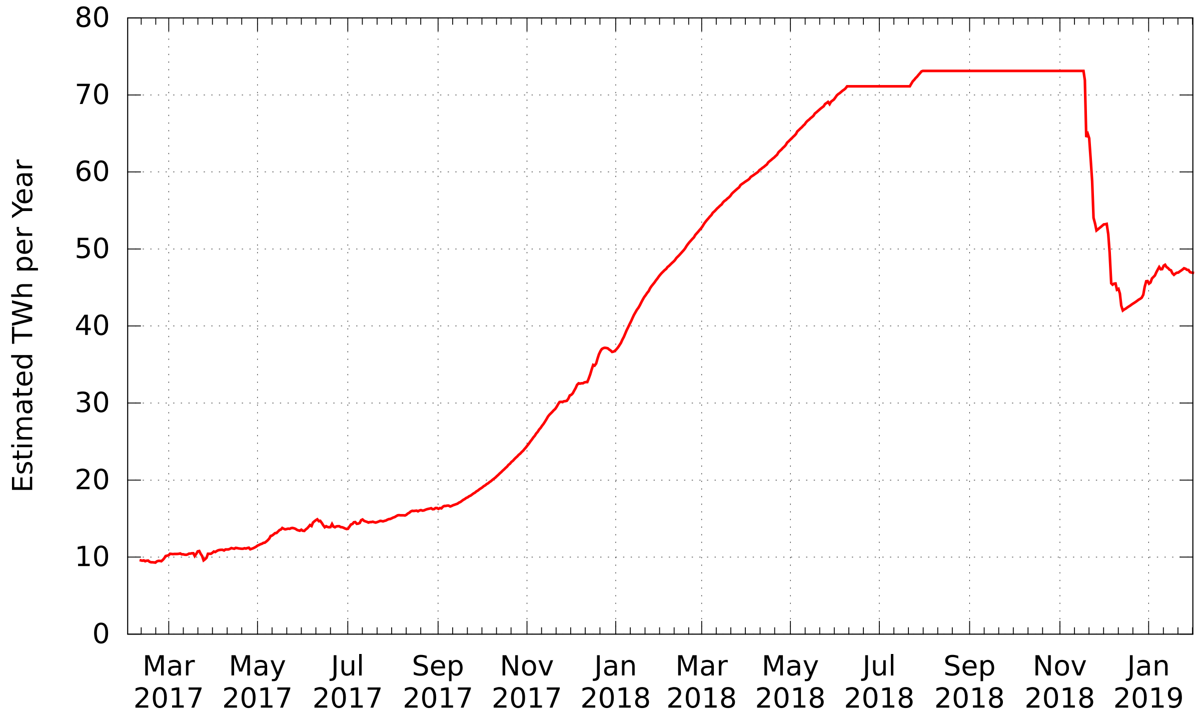

Figure 10: Global mining energy consumption during the last 2 years

[7]https://digiconomist.net/bitcoin-energy-consumption

But it doesn't stop there. Proof-of-work gives more rewards to people with better and more equipment. The higher your hash rate is, the higher the chance that you'll get to create the next block, and receive the mining reward, is. To increase chances even further, miners have come together in what's called "mining pools". They combine their hashing power and distribute the reward evenly across everyone in the pool. Figure 11[8] shows the current Bitcoin mining pool distribution. So to sum it up: proof-of-work is causing miners to use massive amounts of energy and it encourages the use of mining pools, which makes the blockchain more centralized as opposed to decentralized. So, to solve these issues, we have to find a new consensus algorithm.

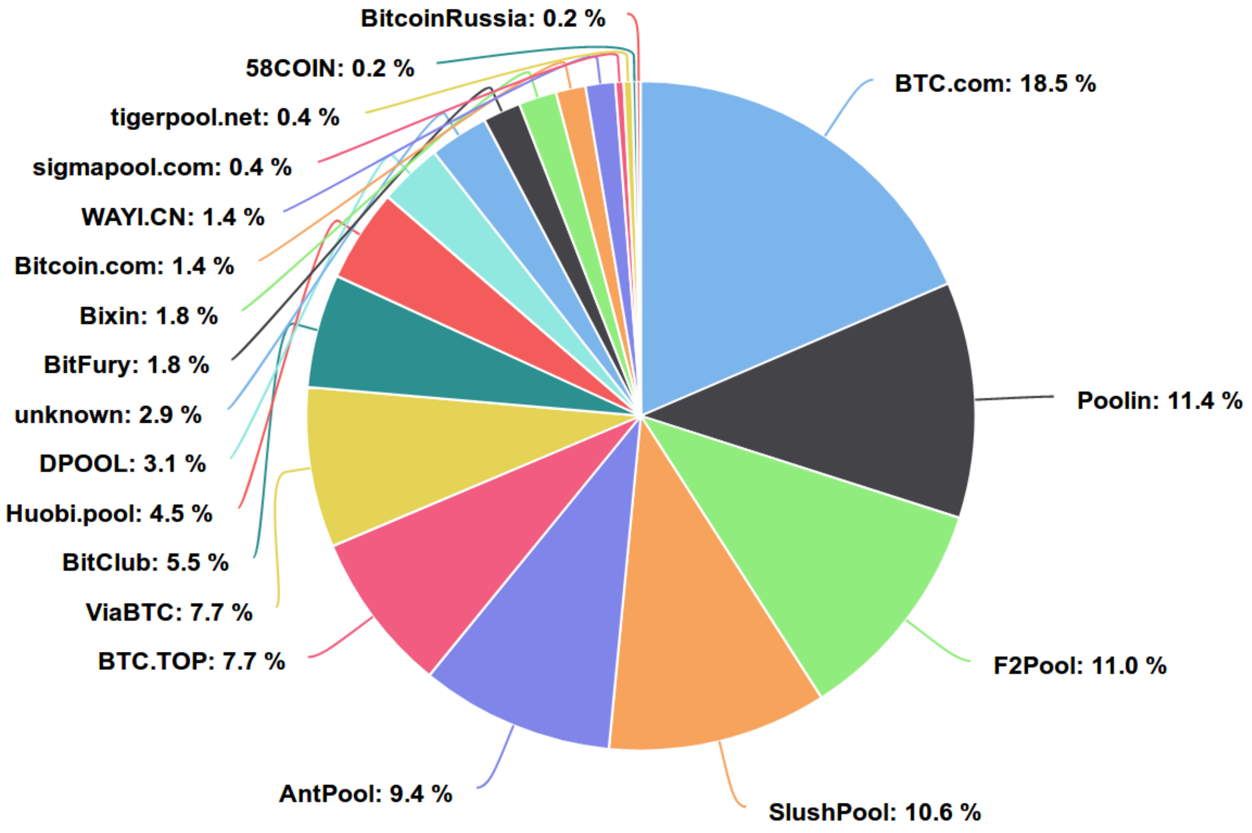

Figure 11: Bitcoin mining pools

In 2011 a new technique was proposed, called "**proof-of-stake**" (PoS) [16]. The basic idea is that letting everyone compete against each other with mining is wasteful. So instead proof-of-stake uses an election process in which one node is randomly chosen to validate the next block. A small difference in terminology here: PoS has no miners, but instead has "validators" and it doesn't let people mine new blocks but instead "mint" or "forge" new blocks. Validators aren't chosen completely randomly. To become a validator, a node has to deposit a certain amount of coins into the network as stake. You can think of this as a security deposit. The size of the stake determines the chances of a validator to be chosen to forge the next block. It's a linear correlation: if someone, for example, holds 1% of the coins of the network, he will end up forge 1% of the blocks.

If a node is chosen to validate the next block, he'll check if all the transactions within it are indeed valid. If everything checks out, the node signs off on the block and adds it to the blockchain. In case validators approve fraudulent transactions, they will lose a part of

[8]https://btc.com/stats/pool

their stake. As long as the stake is higher than what the validator gets from the transaction fees, we can trust them to correctly do their job. Because if not, they lose more money than they gain, it's basically a financial motivator. If a node stops being a validator, his stake plus all the transaction fees that he got will be released after a certain period of time. Not straight away because the network still needs to be able to punish you, should they discover that some of your blocks where fraudulent.

So the differences between PoW and PoS are quite significant. PoS doesn't let everyone mine for new blocks and therefore uses considerably less energy. It's also more decentralized. In PoW we have these mining pools which now control large portions of the Bitcoin blockchain. They centralize the mining process and that's dangerous. As Figure 11 indicates, if the four biggest mining pools would merge together, they would have a majority stake in the network and could start approving fraudulent transactions.

Another important advantage is that setting up a node for a PoS based blockchain is a lot less expensive compared to a PoW based one. You don't need expensive mining equipment and thus PoS encourages more people to set up a node, making the network more decentralized and also more secure.

Moreover, PoS offers higher security guarantee against the 51% attack than the PoW. In PoW, if a single miner or group of miners can obtain 51% of the hasing power, they can effectively control the blockchain. PoS on the other hand makes this attack very impractical, depending on the value of a cryptocurrency. If Bitcoin would be converted to PoS, acquiring 51% of all the coins would set you back a whopping 32.80 billion dollars[9]. So the 51% attack is actually less likely to happen with PoS.

The main challenge in a PoS algorithm is how it selects the next validator. It can't be completely random because the size of the stake has to be factored in. But at the same time the stake alone isn't enough because that will favor rich people, who will get chosen more frequently, will collect more transaction fees, become even richer, and thus increase their chances of being chosen as validator even further. There are a number of proposals to deal with this; so, in fact, there are many "flavors" of proof of stake.

So, PoS is not just a single algorithm, is a category of consensus algorithms that depend on a validator's economic stake in the network. From an algorithmic perspective, there are two major categories: chain-based and Byzantine Fault Tolerant (BFT). In **chain-based PoS**, the algorithm pseudo-randomly selects a validator during each time slot, and assigns that validator the right to create a single block. In **BFT-style PoS**, the act of suggesting the next block and creating the next block are seperated. So validators that suggest the next block are selected randomly. Then a multi-round voting process takes place. At the end of the process all (honest and online) validators permanently agree on whether or not any given block is part of the chain. Unlike chain-based PoS, BFT-style PoS consensus on a block can come within one block, and does not depend on the length or size of the chain after it.

[9] 1BTC = \$3,667.76, Market Cap = \$64.31B, 51% $\times$ Market Cap = \$32.80B, https://markets.bitcoin.com/

3.2.4 Directed Acyclic Graphs

Directed Acyclic Graphs (DAGs) is an exciting new development of distributed ledger technology that do not use the data structure of traditional blockchains. BlockDAG protocols increasing throughput by replacing the underlying chain-structured ledger with DAG structures. Transactions in the DAG run asynchronously, which means that transactions operate independently and do not conform to a particular process. Forks are acceptable in these protocols, so a blockDAG distributed ledger can have multiple heads in the same time. Moreover, there is a algorithm which traverses the DAG and resolves the conflicts of these forks. In case of a conflict transaction in two or more blocks, the block that the algorithm visited first is the valid one and the rest of the blocks are invalid. In other words, topological ordering between the blocks of a DAG, leads to event ordering between the transactions. A big advantage of DAGs is that its data structure theoretically allows infinite number of transactions to be processed. On the other hand, the main drawback of DAGs is that it is hard to be used in a general purpose decentralized platform, like Ethereum, where total ordering of events matters. Therefore, these protocols are focusing on simpler applications, like decentralized money e.g. Bitcoin.

3.3 What's the novelty of this master thesis

In this master thesis we consider the idea of organizing the blockchain peers into a DHT in order to store the blockchain data. As we discussed in Section 3.1, storage capacity is the primary limiting factor to maintain an archival node. As blockchains get older, their size gets larger. Currently, Bitcoin blockchain is over than 190GB. In addition, as long as some of the adove ideas about scalability put into practice, blockchain size will grow even faster. The ever-growing size of the blockchain it might be a factor of making the system more centralized, since too few of the nodes will be able to keep a full replica of the data.

In addition, archival nodes with open ports, have little or no incentives to waste their storage and their bandwidth, in order to keep all the data and serve it to a bootstrapping node. Besides, archival nodes not only serve bootstrapping nodes, but also they serve historical full blocks to every other node that has been offline for a while. So, their usefulness is huge. The more archival nodes there are, the healthier the system is. Thus, it is reasonable to mitigate their load (especially that we call “cold data”, some historical blocks) by splitting it into a DHT.

DHT is a structured peer-to-peer network that provides a lookup service by sharing the data across all participating nodes. Peer-to-peer networks, in particular DHTs, will be discussed in Chapter 4.

4. PEER-TO-PEER NETWORKS

4.1 Peer-to-peer Architecture

Peer-to-peer (P2P) computing or networking is a distributed application architecture that partitions tasks or workloads between peers. Peers make a portion of their resources, such as processing power, disk storage or network bandwidth, directly available to other network participants, without the need for central coordination by servers. A peer-to-peer network is designed around the notion of equal peer nodes simultaneously functioning as both ”clients” and ”servers” to the other nodes on the network. This model of network arrangement differs from the client-server model where communication is usually to and from a central server.

Peer-to-peer networks generally implement some form of virtual overlay network on top of the physical network topology, where the nodes in the overlay form a subset of the nodes in the physical network. Data is exchanged indirectly over the underlying TCP/IP network, but at the application layer peers are able to communicate with each other directly, via the logical overlay links. Overlays are used for indexing and peer discovery, and make the P2P system independent from the physical network topology. Based on how the nodes are linked to each other within the overlay network, and how resources are indexed and located, we can classify networks as unstructured or structured.

4.1.1 Unstructured Peer-to-peer Networks

Unstructured peer-to-peer networks (Figure 12) do not impose a particular structure on the overlay network by design, but rather are formed by nodes that randomly form connections to each other. Because there is no structure globally imposed upon them, unstructured networks are easy to build and allow for localized optimizations to different regions of the overlay. Also, because the role of all peers in the network is the same, unstructured networks are highly robust in the face of high rates of ”churn”-that is, when large numbers of peers are frequently joining and leaving the network.

However, the primary limitations of unstructured networks also arise from this lack of structure. In particular, when a peer wants to find a desired piece of data in the network, the search query must be flooded through the network to find as many peers as possible that share the data. Flooding causes a very high amount of signaling traffic in the network and uses more CPU / memory (by requiring every peer to process all search queries). Furthermore, since there is no correlation between a peer and the content managed by it, there is no guarantee that flooding will find a peer that has the desired data. Popular content is likely to be available at several peers and any peer searching for it is likely to find the same thing. But if a peer is looking for rare data shared by only a few other peers, then it is highly unlikely that search will be successful.

4.1.2 Structured Peer-to-peer Networks

In structured peer-to-peer networks (Figure 13) the overlay is organized into a specific topology, and the protocol ensures that any node can efficiently search the network for a file / resource, even if the resource is extremely rare. The most common type of structured P2P networks implement a distributed hash table (DHT), in which a variant of consistent hashing [10] is used to assign ownership of each file to a particular peer. This enables peers to search for resources on the network using a hash table: that is, (key, value) pairs are stored in the DHT, and any participating node can efficiently retrieve the value associated with a given key.

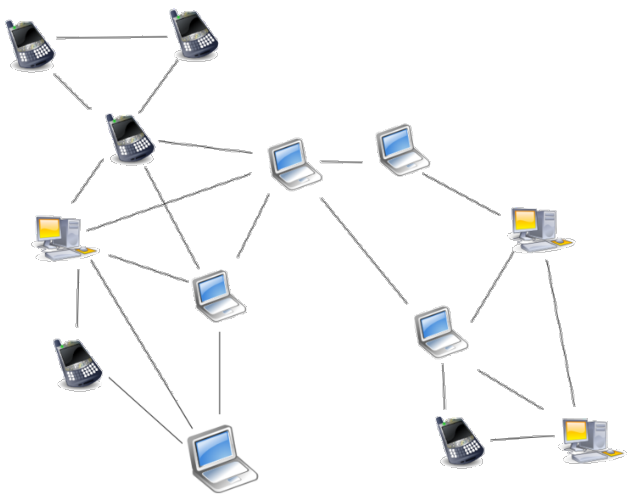
Figure 12: Unstructured P2P network

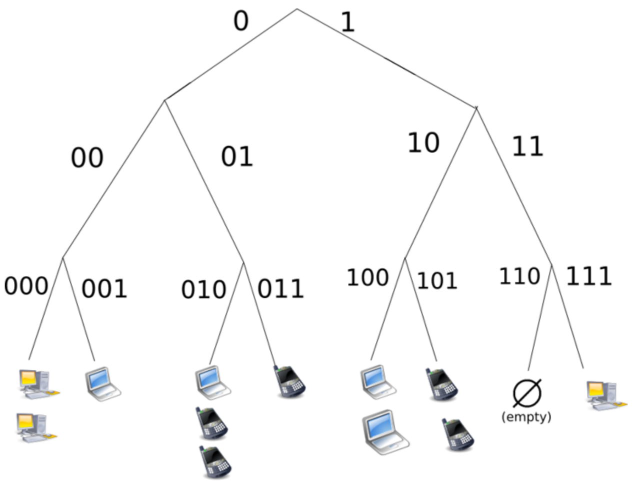

Figure 13: Structured P2P network

4.2 Distributed Hash Tables

A distributed hash table (DHT) is a class of a decentralized distributed system that provides a lookup service similar to a hash table: (key, value) pairs are stored in a DHT, and any participating node can efficiently retrieve the value associated with a given key. Keys are unique identifiers which map to particular values, which in turn can be anything from addresses, to documents, to arbitrary data. Responsibility for maintaining the mapping from keys to values is distributed among the nodes, in such a way that a change in the set of participants causes a minimal amount of disruption. This allows a DHT to scale to extremely large numbers of nodes and to handle continual node arrivals, departures, and failures.

4.2.1 Properties

DHTs characteristically emphasize the following properties:

- Autonomy and decentralization: the nodes collectively form the system without any central coordination.
- Fault tolerance: the system should be reliable even with nodes continuously joining, leaving, and failing.

- Scalability: the system should function efficiently even with thousands or millions of nodes.

A key technique used to achieve these goals is that any one node needs to coordinate with only a few other nodes in the system - most commonly, $O(log\ N)$ of the N participants - so that only a limited amount of work needs to be done for each change in membership. DHTs must deal with more traditional distributed systems issues such as load balancing, data integrity, and performance (in particular, ensuring that operations such as routing and data storage or retrieval complete quickly).

4.2.2 Structure

The structure of a DHT can be decomposed into several main components. The foundation is an abstract keyspace, such as the set of 160-bit strings. A keyspace partitioning scheme splits ownership of this keyspace among the participating nodes. An overlay network then connects the nodes, allowing them to find the owner of any given key in the keyspace.

Once these components are in place, a typical use of the DHT for storage and retrieval might proceed as follows. Suppose the keyspace is the set of 160-bit strings. To index a file with given $filename$ and $data$ in the DHT, the SHA-1 [14] hash of $filename$ is generated, producing a 160-bit key k, and a message $put(k, data)$ is sent to any node participating in the DHT. The message is forwarded from node to node through the overlay network until it reaches the single node responsible for key k as specified by the keyspace partitioning. That node then stores the key and the data. Any other client can then retrieve the contents of the file by again hashing $filename$ to produce k and asking any DHT node to find the data associated with k with a message $get(k)$. The message will again be routed through the overlay to the node responsible for k, which will reply with the stored $data$.

4.3 Chord

Chord [20] is a protocol and algorithm for a peer-to-peer distributed hash table. Like every DHT, it stores key-value pairs by assigning keys to different nodes; a node will store the values for all the keys for which it is responsible. Chord specifies how keys are assigned to nodes, and how a node can discover the value for a given key by first locating the node responsible for that key.

Nodes and keys are assigned an m-bit identifier using consistent hashing. Consistent hashing is integral to the robustness and performance of Chord because both keys and nodes are uniformly distributed in the same identifier space with a negligible possibility of collision. Thus, it also allows nodes to join and leave the network without disruption.

Using the Chord lookup protocol, nodes and keys are arranged in an identifier circle that has at most 2^m nodes, ranging from 0 to $2^m - 1$. Some of these nodes will map to machines or keys while others will be empty.

Each node has a successor and a predecessor. The successor to a node is the next node in the identifier circle in a clockwise direction. The predecessor is counter-clockwise. If there is a node for each possible ID, the successor of node 0 is node 1, and the predecessor of node 0 is node 2^m-1; however, normally there are ”holes” in the sequence. For example, the successor of node 48 may be node 61 (nodes from 49 to 60 do not exist); in this case, the predecessor of node 61 will be node 48.

The concept of successor can be used for keys as well. The successor node of a key k is the first node whose ID equals to k or follows k in the identifier circle, denoted by $successor(k)$. Every key is assigned to (stored at) its successor node, so looking up a key k is to query $successor(k)$.

Since the successor (or predecessor) of a node may disappear from the network (because of failure or departure), each node records a whole segment of the circle adjacent to it, i.e., the r nodes preceding it and the r nodes following it. This list results in a high probability that a node is able to correctly locate its successor or predecessor, even if the network in question suffers from a high failure rate.

4.3.1 Query Key

The core usage of the Chord protocol is to query a key from a client, i.e. to find $successor(k)$. The basic approach is to pass the query to a node's successor, if it cannot find the key locally. This will lead to a $O(N)$ query time where N is the number of machines in the ring.

To avoid the linear search, Chord implements a faster search method by requiring each node to keep a $finger\ table$ (Figure 14) containing up to m entries, recall that m is the number of bits in the hash key. The i^{th} entry of node n will contain $successor((n+2^{i-1})\mod 2^m$. The first entry of finger table is actually the node's immediate successor. Every time a node wants to look up a key k, it will pass the query to the closest successor of k in its finger table (the ”largest” one on the circle whose ID is smaller than k), until a node finds out the key is stored in its immediate successor (Figure 15). With such a finger table, the number of nodes that must be contacted to find a successor in an N-node network is $O(log\ N)$.

4.3.2 Node Join

Whenever a new node joins, three invariants should be maintained (the first two ensure correctness and the last one keeps querying fast):

1. Each node's successor points to its immediate successor correctly.
2. Each key is stored in $successor(k)$.
3. Each node's finger table should be correct.

To satisfy these invariants, a predecessor field is maintained for each node. The following tasks should be done for a newly joined node n:

1. Initialize node n (the predecessor and the finger table).
2. Notify other nodes to update their predecessors and finger tables.
3. The new node takes over its responsible keys from its successor.

The predecessor of n can be easily obtained from the predecessor of $successor(n)$ (in the previous circle). As for its finger table, there are various initialization methods. The simplest one is to execute find successor queries for all m entries, resulting in $O(m\ log\ N)$ initialization time.

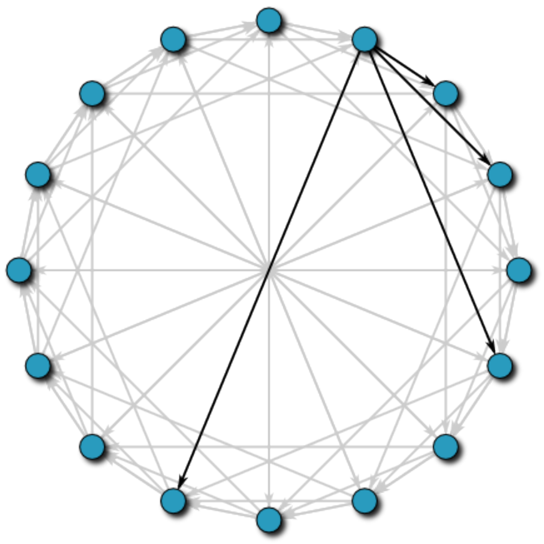

Figure 14: The ”fingers” of one of the nodes

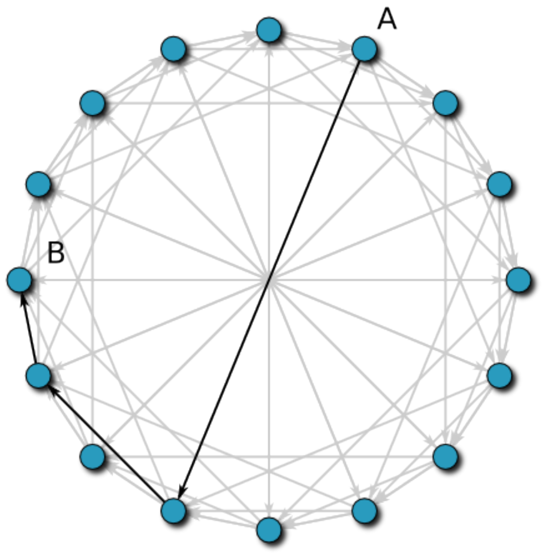

Figure 15: Routing path between nodes A and B

5. BITCOIN BLOCKCHAIN IN CHORD DHT

Based on what has been said about Bitcoin's decentralization, we consider the idea of organizing the Bitcoin peers into a DHT in order to store the Bitcoin data. Such an option is to enable users with lower-end machines to participate in the Bitcoin network by validating transactions and, in general, the correctness of the Bitcoin blockchain by themselves, without having large disk space. This will be achieved as each participating node of the DHT will store locally, on its disk, only a portion of the total volume of the blockchain data. A second and even more important advantage of having a blockchain distributed over a DHT is its contribution to a more decentralized Bitcoin network.

However, a DHT can not ensure that there will be no data loss. Therefore, implementing a DHT to store the blockchain data is neither intended nor can succeed in replacing full nodes. As long as Bitcoin exists and people trade through it, there will be incentives to maintain a full node. This master thesis suggests the creation of a DHT where a part of current Bitcoin nodes and new ones, will have their conventional machine, contributing to a network (that of the DHT overlay), based on simple users and in an attempt to preserve it decentralization of Bitcoin.

The way to maintain Bitcoin's decentralization will be based on DHT scalability. Given the continuous and constant increase in blockchain data volume, it is increasingly costly to send chaindata to a bootstrapper node. Plus, a full node has no immediate benefit or incentive to serve a bootstrapper. These issues solve a DHT as its scaling is a compensating factor for the increasing volume of data, while it is also beneficial to the bootstrapping node service.

The participating nodes of the DHT will be members of the Bitcoin network. Specifically, based on what has been mentioned in Section 2.7 about Bitcoin's full nodes, DHT's nodes will maintain links to other nodes in the network to receive and transmit new transactions and blocks. At the same time, compliance with the DHT protocol is necessary, e.g. communication with neighboring nodes and answering queries, functions that have additional resource requirements. It is also important to define the depth in the blockchain that a block should have to be considered as confirmed and assigned to the DHT. Therefore, the Bitcoin network will be a network layer on top of the DHT network.

A DHT system has some important operating parameters and implications for the use of its node resources. Thus, simulations will be carried out and an assessment will be made as to whether such an implementation is feasible, given the requirements of the Bitcoin protocol.

5.1 DHT-Distributed-Blockchain Base Protocol

The protocol specifies how to find the locations of keys, how new nodes join the system, and how to recover from the failure of existing nodes. This section describes a simplified version of the protocol that does not handle concurrent joins or failures. However, as we will see later on, this kind of joins and failures are very unlikely to happen, because the number of nodes in the Bitcoin network are relative stable through time.

- **Consistent Hashing**:

  At its heart, our DHT provides fast distributed computation of a hash function mapping keys to nodes responsible for them. It uses consistent hashing, which has several good properties. With high probability the hash function balances load (all nodes receive roughly the same number of keys). Also with high probability, when an N^{th} node joins (or leaves) the network, only an $O(1/N)$ fraction of the keys are moved to a different location - this is clearly the minimum necessary to maintain a balanced load. The consistent hash function assigns each node and key an m-bit identifier using the base hash function SHA-1.

- **Hash Chain**:

  Like every DHT, our DHT stores key-value pairs by assigning keys to different nodes; a node will store the values for all the keys for which it is responsible. In our case keys are blocks' hash values, while values are blocks' data.

  However, unlike to other kinds of data, blockchain data have a certain property: Each block contains a reference to the previous block (i.e. the hash value of the previous block). To set the hash value which needed for the first block to be constructed, we hash an arbitrary string:

$$\underbrace{hash(''This\ is\ a\ random\ string''), data_0}_{block_0(genesisblock)}, \dots$$

  Every block after the genesis block, includes the hash value of the previous block:

$$\dots, \underbrace{hash(block_{i-2}), data_{i-1}}_{block_{i-1}}, \underbrace{hash(block_{i-1}), data_i}_{block_i}, \underbrace{hash(block_i), data_{i+1}}_{block_{i+1}}, \dots$$

- **Block Handling**:

  Similarly to the Bitcoin network, mined blocks are broadcast in regular interval in our system.

  Although the number of full nodes can fluctuate in the Bitcoin network, our DHT nodes should be a fixed size, i.e. 2^m. Then, we divide the size of the network to this number. As a result, each DHT node is a set of full nodes. The nodes among the same set maintain the same data with respect to the DHT.

Every node maintains a list (stack) in which it stores all the blocks that is heard of. Every new mined block is pushed to the stack. That is, the blocks within the stack are in chronological order: the oldest block is at the bottom of the stack, while the newest block is at the top of the stack.

Although a node saves all the mined blocks in the list, not all blocks can stay in the list forever. Every node stores some hot data, plus a small portion of the rest of the blocks (the blocks which is responsible for).

With the term $hot\ data$ we mean the data that have a demand much higher than the average. In the case of a blockchain, we already know that such data exists and is the most recent one. It is a common practice, when a transaction is made, both by the sender and the recipient to check for its confirmation. This check, for a lightweight client, is a query to one or more full nodes and concerns the most recent blocks of the blockchain. Usually, it takes at least 6 confirmations for a transaction to be consider as permanent. Bitcoin itself has a rule: The miner can not reclaim and use the mined bitcoins (block reward) until 100 blocks have passed. That is to say, because it may be some forks, the miner have to wait until 100 blocks have passed in order to be sure that the block he had mined is part of the main blockchain. So, to avoid congestion at specific points in the DHT and to enhance that rule, each node will keep the blocks added to the blockchain within the last 24 hours. In this way, the majority of the queries will be answered by the recipient of the question, without the need to pass the query in another node of the DHT. Bitcoin generates an average of 144 blocks per 24 hours (one block every 10 minutes), so each node will hold about 144 MB extra. For the simulations, it will be assumed that 50% of the total queries refer to blocks created in the last 24 hours, made by lightweight clients.

The blocks that a node is responsible to keep lie into two categories:

1. The genesis block is stored in node 0 by default. Every other block uses some (m) bits of the hash of the previous block to determine its position in the DHT. Let's assume that it uses the last m bits of the hash of the previous block (the first m bits of the hash is almost always zero). E.g.: In a 256-node-DHT, block i has a 0x5A suffix in its hash. So, the node number 90 (0x5A) should maintain the next block (block $i + 1$). (primary blocks)
2. Replica blocks (primary blocks of other nodes) that has to be kept due to the replication policy.

For our replication policy we divide our DHT into k segments (k should be a power of 2). Each segment is a full copy of the blockchain data. Blocks are the same every $2^m/k$ nodes. Figure 16 shows a 32-node-DHT split in 4 segments. In this case, it's like a 32-node-DHT where each node sends all of its primary blocks to its 4th finger. If one's 4th finger departs, the node sends its blocks to the next node in the ring (new successor). If a 4th finger joins, the node sends its blocks to the new successor and asks from the old successor to delete these blocks. Each node that holds a replica block can answer a query, although the query does not refer to its primary blocks.

So, every node stores the most recent 144 blocks (hot data), and every older block of the same color (as the node's color). Figure 17 shows red's node block list.

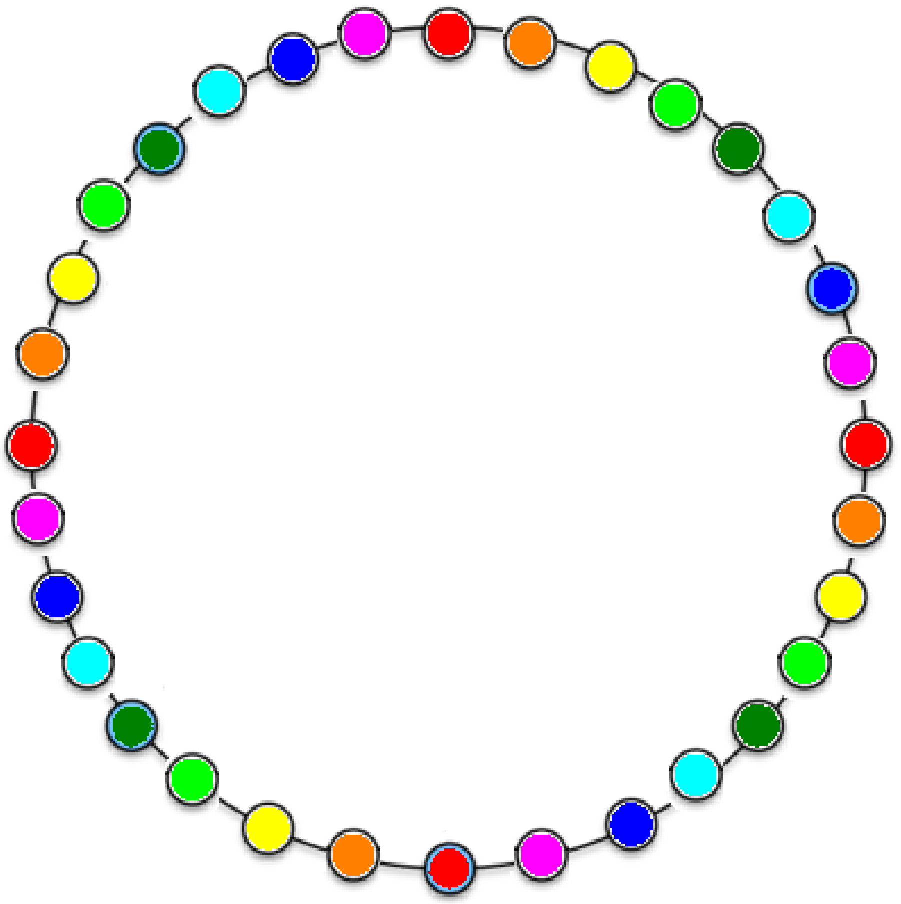

Figure 16: A 32-node-DHT split in 4 segments

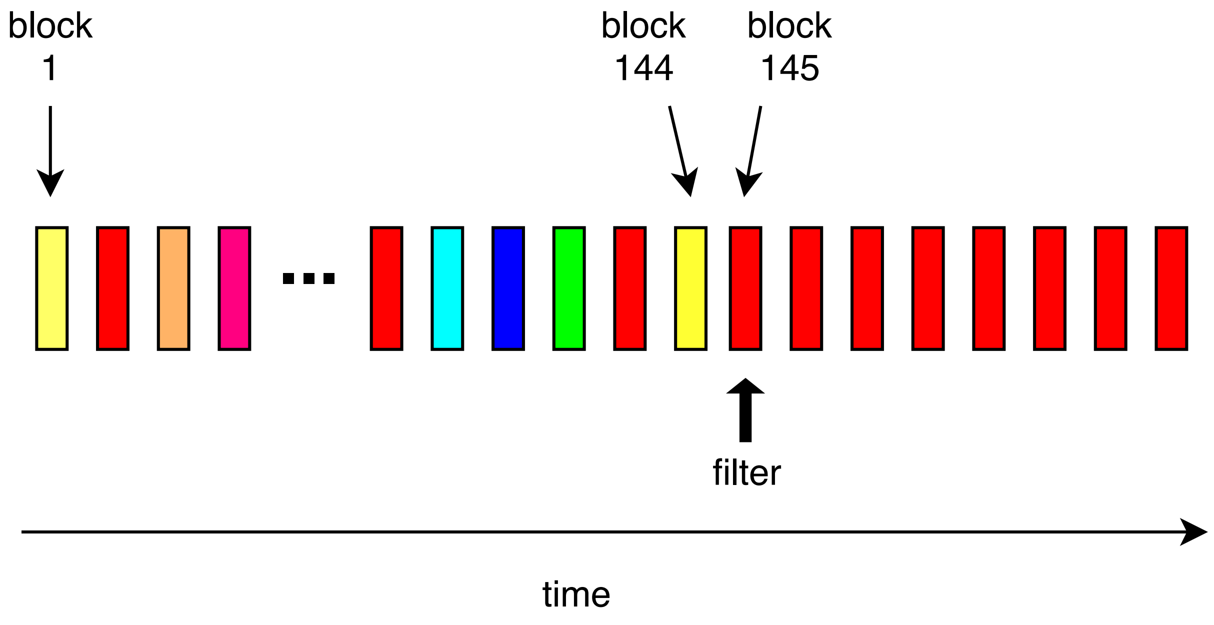

Figure 17: Red's node block list

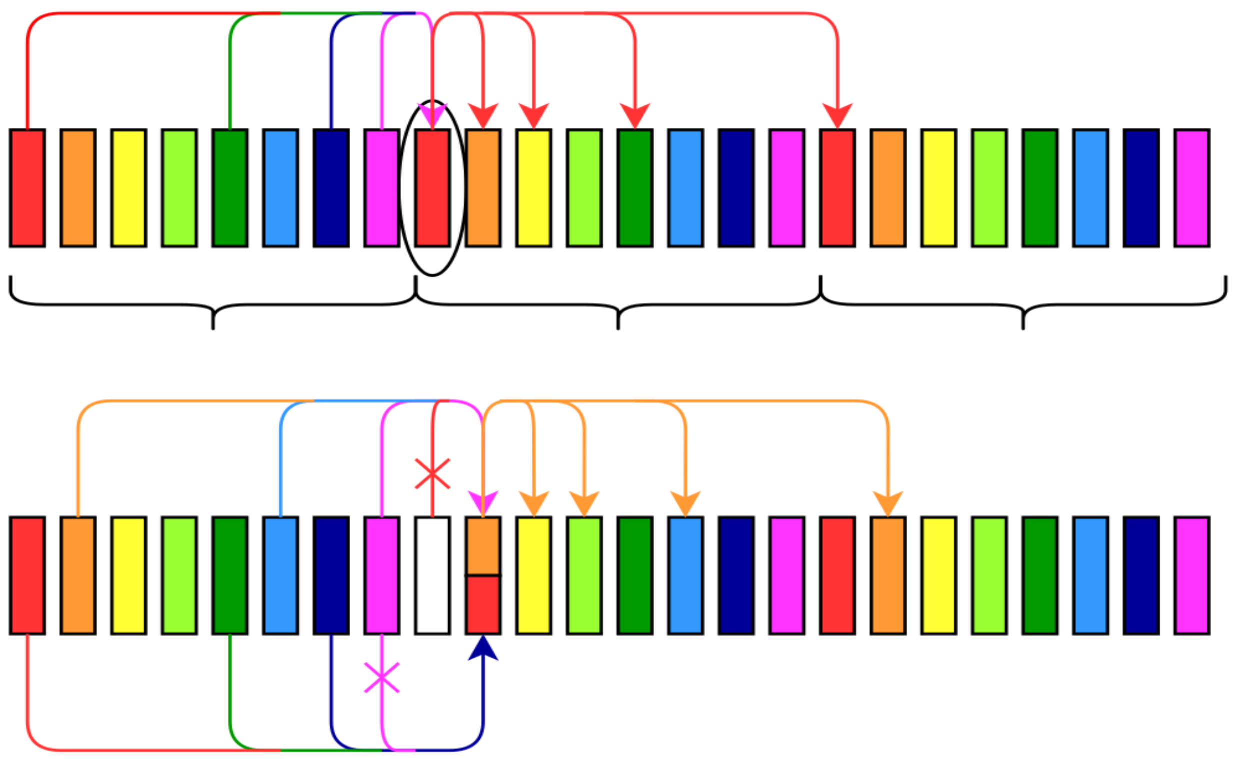

Figure 18: DHT pointers before and after a node's departure

- **Node Departs**:

  Our DHT improves the scalability by avoiding the requirement that every node know about every other node. A DHT node needs only a small amount of "routing" information about other nodes. Because this information is distributed, a node resolves the hash function by communicating with a few other nodes. In our network, each node maintains information only to $1 + log_2(2^m/k)$ other nodes (less than $O(log\ N)$), and a lookup requires up to $log_2(2^m/k)$ messages.

  Figure 18 shows node pointers before and after a node departure in 8-node-segment DHT. On the top (before departure) red block has pointers to orange, yellow, dark green and next red block (red arrows). While, at the same time, it receives a pointer from pink, dark blue, dark green and previous red block. On the bottom (after departure), all pointers that were showing to red node, now show to the orange node (red's primary successor). Moreover, the leftmost red node fills up with its data the new successor. The orange node holds both orange and red blocks.

- **Node Joins**:

  In a dynamic network, nodes can join at any time. When a node n joins the network, it must initialize its predecessor and its fingers, and download all the blocks that where mined as long as it was missing.

  For the initialization process, node n learns its predecessor (p) by asking a random node to look it up. Then, node n asks p for a copy of its complete finger table. n use the contents of that table to find the correct values for its own table. n's i^{th} finger will be the immediate successor of p's i^{th} finger. That leads to a constant execution time, since messages to the fingers of p can be done in parallel. For example, let's assume that in a 32-node-segment DHT, $n = 6$. The finger table of p (node 5), looks like:

Table 1: Finger table of the predecessor of a new joining node

$node_5.finger[i]$	$node$	
$node_5.finger[0]$	$5 + 2^0 = \cancel{6}\ 7$	$(node\ 6\ missing)$
$node_5.finger[1]$	$5 + 2^1 = 7$	
$node_5.finger[2]$	$5 + 2^2 = 9$	
$node_5.finger[3]$	$5 + 2^3 = 13$	
$node_5.finger[4]$	$5 + 2^4 = 21$	
$node_5.finger[5]$	$5 + 2^5 = 37$	

So the finger table of node 6 will be like:

Table 2: Finger table of the new joining node

$node_6.finger[i]$	$node$
$node_6.finger[0]$	$node_5.finger[0] = 7$
$node_6.finger[1]$	$node_5.finger[1].finger[0] = 7 + 2^0 = 8$
$node_6.finger[2]$	$node_5.finger[2].finger[0] = 9 + 2^0 = 10$
$node_6.finger[3]$	$node_5.finger[3].finger[0] = 13 + 2^0 = 14$
$node_6.finger[4]$	$node_5.finger[4].finger[0] = 21 + 2^0 = 22$
$node_6.finger[5]$	$node_5.finger[5].finger[0] = 37 + 2^0 = 38$

For the downloading process, node n investigates the suffix of the last block it was able to download. In that way, the node knows exactly which other node stores the first missing block, so it downloads it from that node. Node n asks from the other node the first block that is younger than the block it already possesses. The other node uses binary search ($O(log\ n)$) on its (chronological) ordered list of blocks and sends the required block. Node n repeats this procedure until it reaches the current height. At the same time, it verifies every block, and keeps only these blocks that is responsible for.

5.2 Simulation Parameters

Having mentioned possible safety hazards of a DHT, an appropriate selection of parameters should be made, to ensure that the system to be studied works smoothly. In addition, involving a machine in a DHT entails additional functions and resources, e.g. bandwidth.

The study will be done using a DHT Chord simulator written in Java. Also, the following considerations will be made for the Bitcoin network, based on actual measurements of a full node:

- Considering the top 10 countries with their respective number of reachable nodes[10], and the average Internet connection speed per country in Q1 2017[11]:

Table 3: Top 10 countries with their respective number of reachable nodes

Rank	Country	Nodes	Average connection speed
1	United States	24.88 %	18.7 Mb/s
2	Germany	19.15 %	15.3 Mb/s
3	France	6.85 %	10.8 Mb/s
4	Netherlands	5.14 %	17.4 Mb/s
5	Canada	3.81 %	16.2 Mb/s
6	China	3.75 %	7.6 Mb/s
7	United Kingdom	3.46 %	16.9 Mb/s
8	Singapore	3.09 %	20.3 Mb/s
9	Russian Federation	2.80 %	11.8 Mb/s
10	Japan	2.37 %	20.2 Mb/s
1-10	Top 10 countries	75.30 %	16.1 Mb/s

Bitcoin nodes average Internet connection speed is that of **16.1 Mb/s**[12].

- **Bitcoin network latency** (Figure 19[13]):
lower average = 6 msec, **median average = 126 msec**, upper average = 308 sec

- **Delay until an unexpected departure is confirmed**: Bitcoin uses a *ping* message to check that the connection is still online. Figure 20[14] shows the number of ping messages per second received by a full node the last 30 days. The minimum value (the worst case scenario) is 0.81 messages per second. Assuming that at least 3 missing replies needed for a connection to be considered as closed (so at least 3.70 sec), we set this parameter to **5 sec**. (In the simulations, all departures will be considered as unexpected)

- **Incoming bandwidth usage** (Figure 21[15]): **avg = 95 Kb/s**, max = 192 Kb/s

- **Outgoing bandwidth usage** (Figure 21): **avg = 1.6 Mb/s**, max = 21 Mb/s

- **Average time to validate (check and accept) a block** (Figure 22[16]):
CheckBlock() + AcceptBlock() = 33 msec + 13 msec = **46 msec**

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

[11]https://en.wikipedia.org/wiki/List_of_countries_by_Internet_connection_speeds

[12] $\frac{24.88\times18.7+19.15\times15.3+6.85\times10.8+5.14\times17.4+3.81\times16.2+3.75\times7.6+3.46\times16.9+3.09\times20.3+2.80\times11.8+2.37\times20.2}{24.88+19.15+6.85+5.14+3.81+3.75+3.46+3.09+2.80+2.37} = 16.1$

[13]https://statoshi.info/dashboard/db/peers

[14]https://statoshi.info/dashboard/db/p2p-messages

[15]https://statoshi.info/dashboard/db/bandwidth-usage

[16]https://statoshi.info/dashboard/db/function-timings

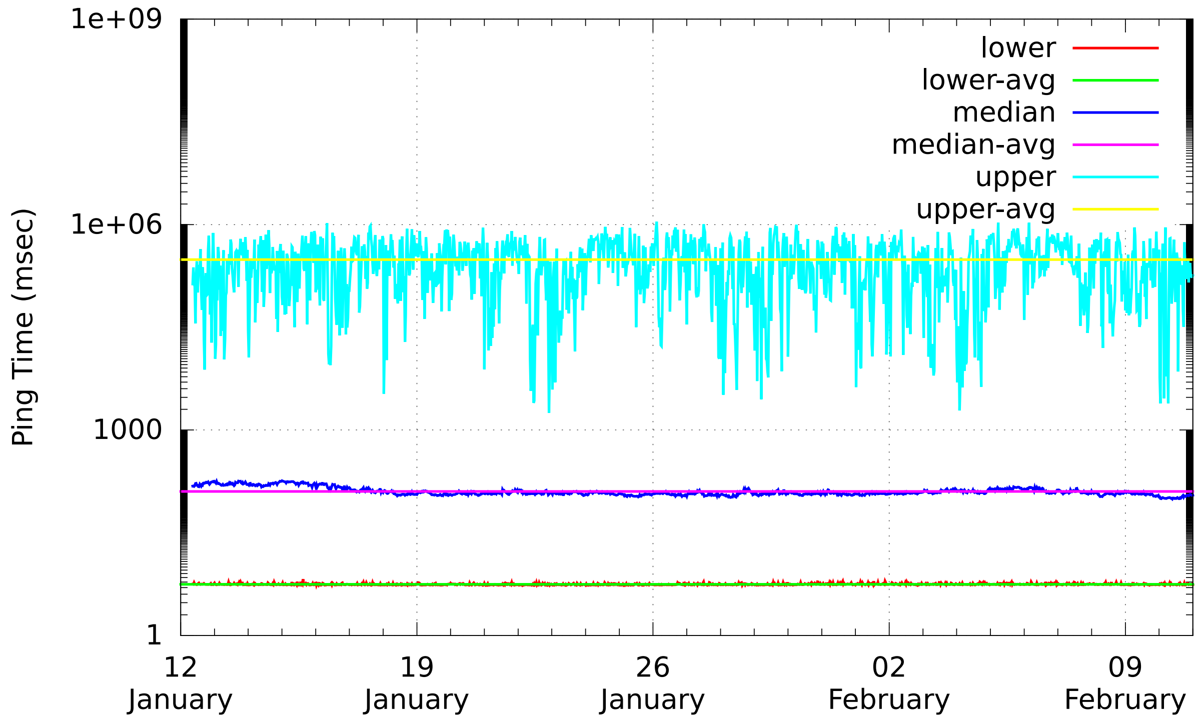

Figure 19: Bitcoin network latency the last 30 days

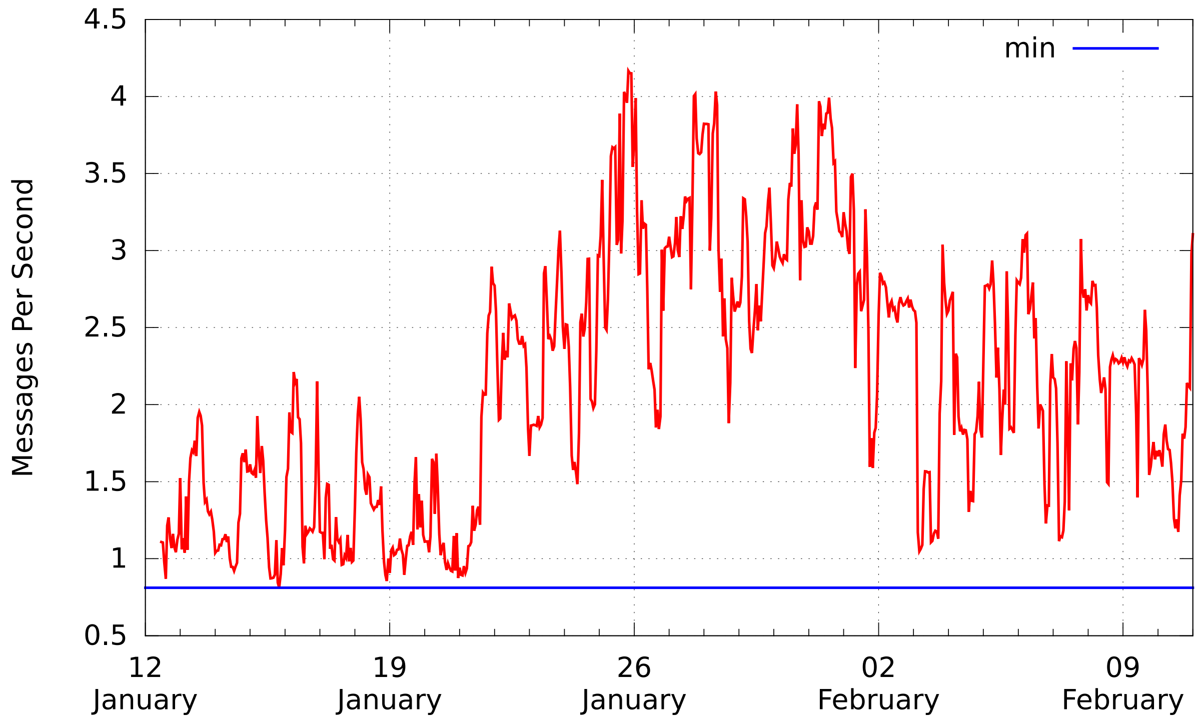

Figure 20: Ping messages per second received by a full node the last 30 days

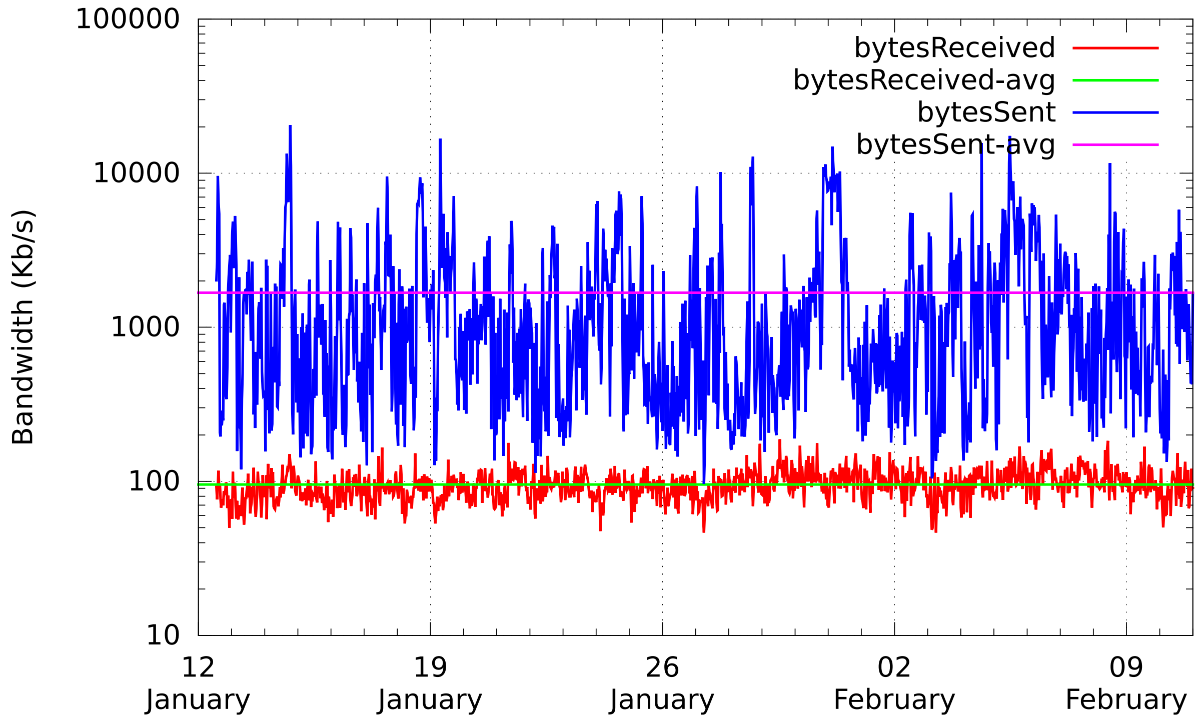

Figure 21: Incoming & Outgoing bandwidth usage per second the last 30 days

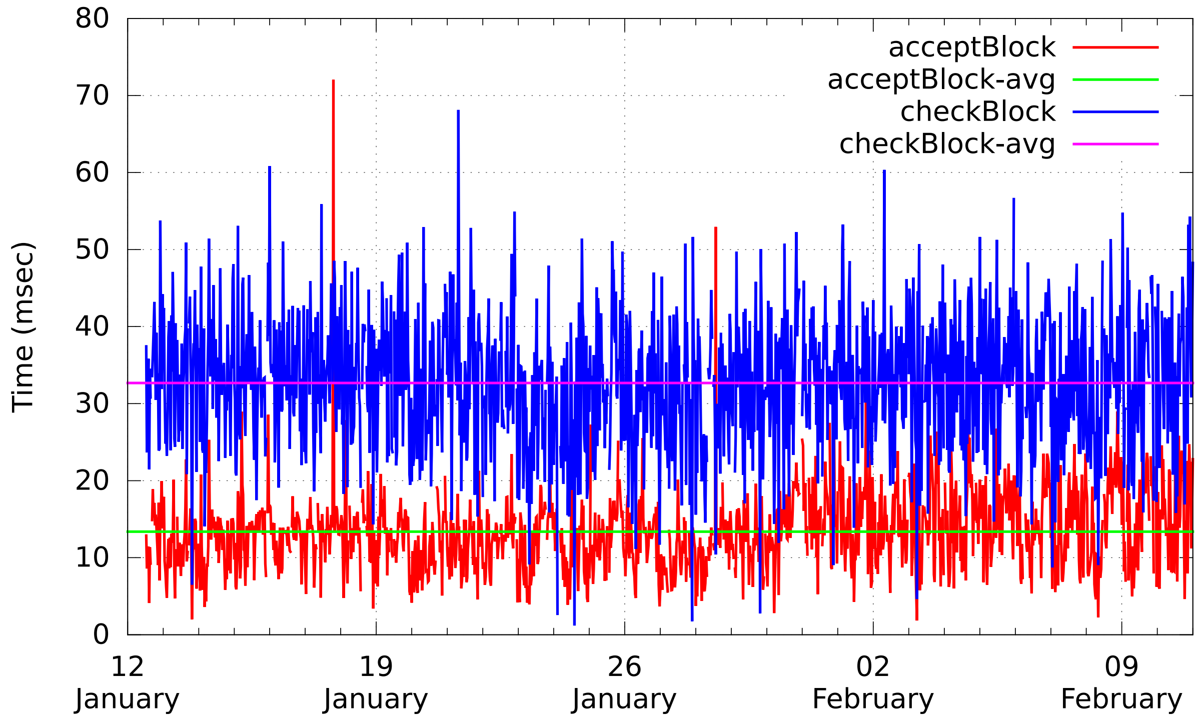

Figure 22: Average time to check and accept a block the last 30 days

Depending on the calculations that have been made on the number of Bitcoin nodes in Section 2.7, it is assumed that the number of Bitcoin full nodes is 144,375, of which 10,500 are listening full nodes. The 133,875 no-listening full nodes can be either archival nodes or not. However, given the cost of maintaining a full copy of Bitcoin blockchain, while keeping your ports closed hence not serving the blockchain to other peers, it is obvious that most of these nodes should be non-archival nodes.

Non-archival full nodes are the ones who will be most motivated to participate in DHT, as most of them will fit into the profile that has been described about DHT participants: users with low-capacity machines who want to maintain Bitcoin's decentralization. For the DHT system simulations to be made, DHT nodes will be considered as the set of the full nodes (archival or not).

In our simulations we set the number of the full nodes to 2^{16} and the number of the DHT nodes to 2^{8}. So, if we divide the size of the network (2^{16}) with the size of the DHT (2^{8}), each DHT node is a set of 2^{8} full nodes. The nodes among the same set maintain the same data with respect to the DHT. Moreover, our DHT is divided into 8 segments, and every segment is a full copy of the blockchain data. This means that blocks are the same every $2^{8}/8 = 32$ nodes.

The use of bandwidth for uploading in Bitcoin protocol results from sending data to synchronized network nodes, bootstrapping new nodes and serving lightweight clients. This service is currently offered by the listening full nodes. An inherent part of the DHT functionality is the response to queries about its data, so DHT nodes must also take part of the upload service. In the simulations, DHT nodes will take over all the uploads that Bitcoin is supposed to serve, and it will be assumed that each uploading query corresponds to the sending of a block, that is 1 MB. We calculate the query rate as follows: $(Uploading\ bandwidth\ (bytes/sec) \times number\ of\ nodes)/(bytes\ per\ query)$, so there are up to $\frac{(1.6/8)(MB/sec)\times 2^{16}}{1(MB/query)} = 0.2 \times 2^{16} qps$.

In bandwidth usage will also be included uploading / downloading due to DHT protocol, which is primarily a copy data transfer as a result of node departure. To define the average rate of arrivals or departures per day, we average the standard deviation of the daily number of nodes from January 15, 2019 to February 15, 2019. We set this number at the +0.5% of the population of the nodes per day.

Currently, Bitcoin blockchain size is over than 190GB and block height is over than 563K blocks. This means, if blockchain size was evenly splitted between all blocks, each block would be around 354KB on average. However, not every block has the same size. There are some parameters that effect the size of each block e.g. the number of transactions in each block and the number of inputs and outputs in each transaction. In the early days of Bitcoin, blocks had too few transactions (~10 on average the first 3 years), while now blocks have much more (~1800 on average the last 3 years). In the simulations, we set every block to 1MB and block height up to 563344 blocks. So, a single copy of the generated blockchain is over than 550GB. This means, in the end of the simulations, if blocks are evenly splitted among the nodes of the DHT, each node will have: 2200 blocks as its primary blocks, 144 blocks as hot data, and 7 $\times$ 2200 = 15400 blocks as replicas.

In order for the blocks to be evenly splitted, we need to use consistent hashing. So we use the SHA-1 hash function to generate the hash values of the blocks.

Table 4 summarizes the simulation parameters:

Table 4: Simulation Parameters

Parameter	Value
Internet speed	$16.1\ Mb/s$
Network latency	$126\ msec$
Delay until an unexpected departure is confirmed	$5\ sec$
Incoming bandwidth usage	$95\ Kb/s$
Outgoing bandwidth usage	$1.6\ Mb/s$
Time to validate a block	$46\ msec$
Query rate	$0.2 \times 2^{16}\ qps$
Arrivals/Departures per day	$+0.5\%\ of\ the\ population$
# full nodes	2^{16}
# DHT nodes	2^{8}
Replication factor	8
Block size	$1\ MB$
Block height	$563344\ blocks$
Block interval	$10\ min$
Hot data	$144\ latest\ blocks$
Queries @ Hot data	50%
Hash function	$SHA-1$

5.3 Simulations

It is worthwhile to study the operation of the DHT at levels of mobility and query service well above the average, as the behavior of both peer-to-peer nodes and query users is unpredictable and may fluctuate considerably. The purpose of these simulations is to observe and evaluate metrics such as capacity savings per node compared to the amount of data that a full node stores today, load balance, query latency, bandwidth usage. To achieve this, we use the Markov Chain Monte Carlo (MCMC) methodology. Each node joins or leaves the network under a certain probability. This probability depends only upon the present state of the system, not on the sequence of events that preceded it (Markov property). As to Monte Carlo method, we run our program 100 times and we average the results.

- **Capacity Savings**:

  Figure 23 shows the gains in capacity that a DHT node has, as a percentage of the volume of data that a full node stores today.

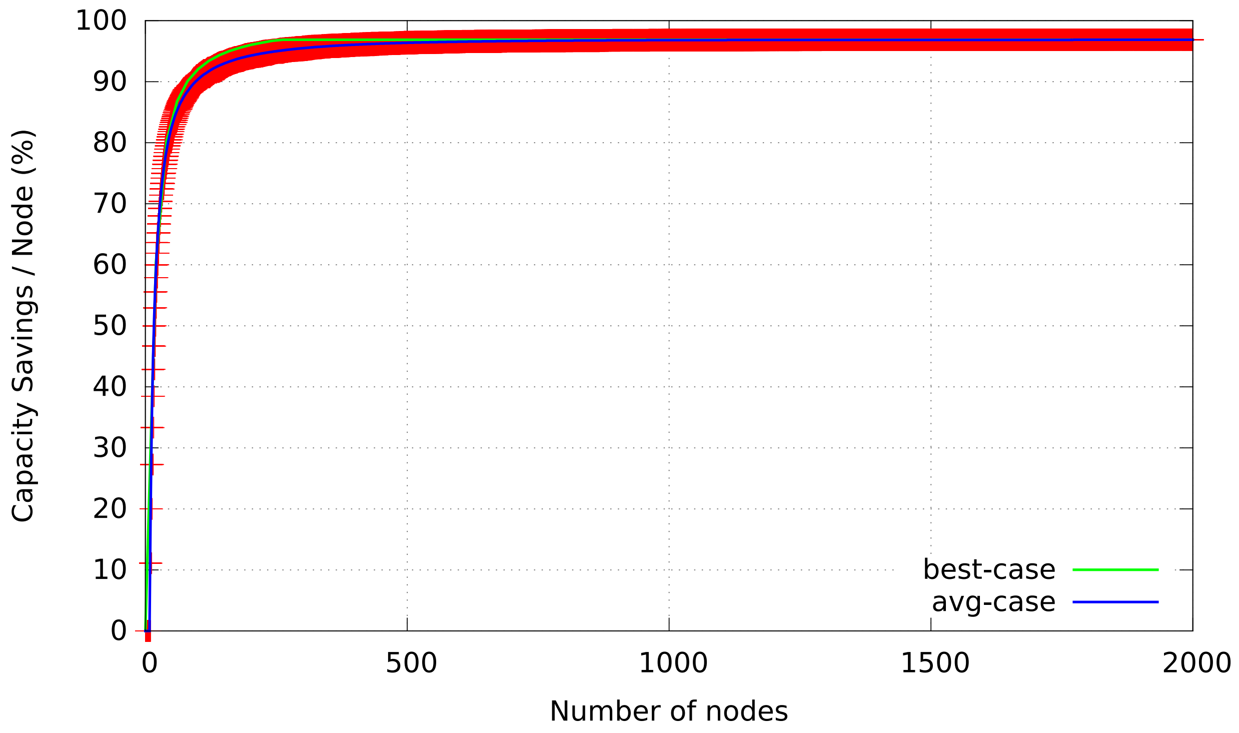

Figure 23: Capacity savings per node

The graph shows a sharp escalation in the capacity savings. In the beginning, the number nodes is less than the replication factor, so each node have to maintain the whole blockchain. As nodes joining our system, capacity saving per node increase, up until there is at least one node at each DHT position. Increasing the number of nodes even further, does not result in further scaling with respect to the capacity savings. However, as we will see later on, it enhances security because the system becomes fault-tolerant, meaning that it can continue to operate even if a component of the network fails. As demonstrated by the graph, the average case of the capacity savings, is very close to the optimal. As for the curve, it is explained by taking into account that the data volume of a DHT blockchain node occurs as the sum of:

$$\frac{(chaindata - hot\ data) \times 8}{2^8} + UTXO\text{-}set + hot\ data$$

where $chaindata$ is the size of the blockchain, 8 is the replication factor, $UTXO$-set is the size of the unspent transaction outputs, and $hot\ data$ is the size of blocks deposited in the blockchain up to 24 hours before.

Consequently, there is a complete scaling in the blockchain volume. Each node, stores its portion of the blockchain data, plus a fixed volume of data, that of the UTXO-set and the hot data. UTXO-set size is almost negligible. UTXO-set depends on the number of active Bitcoin addresses in the network and the number of bitcoins in circulation. Hot data volume is 100% stable under a certain assumption. It can only change by changing the block size.

- **Load Balance / Congestion**:

  Figure 24 shows how the blocks are distributed among the nodes in each segment. The distribution of the blocks is the same in every segment (there is a replica every 32 nodes). As demonstrated by the graph, the average number of blocks per node, is very close to the optimal (that of a perfect split between the nodes). Hence, there is a (almost) perfect load balance between the nodes of the DHT. In addition, the congestion is minimal beacause each node that holds a block (primary or replica) can answer a query.

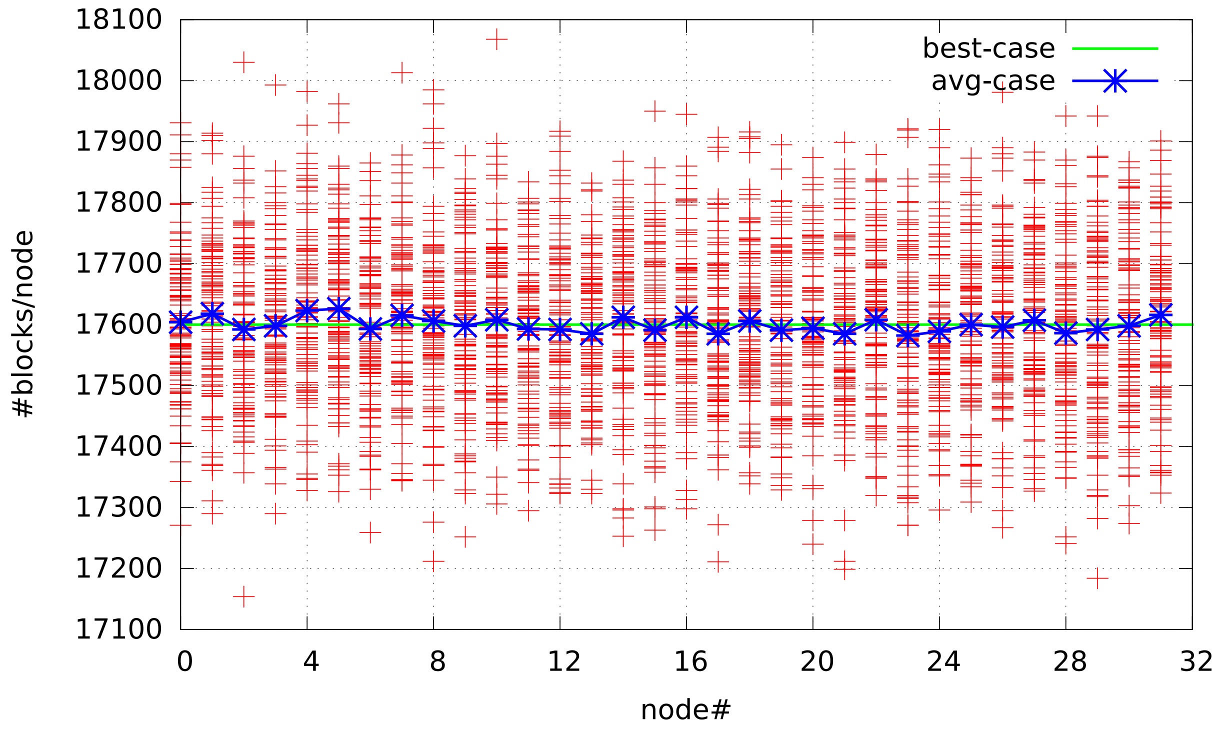

Figure 24: Blocks per node

- **Data Losses**:

  Figure 25 shows the minimum departure rate that has to be achieved in order to have data loss. The minimum departure rate is bounded by the average Internet speed of the network and the amount of data that has to be transferred at a given time. We assume that the Internet speed is constant throughout the years and its value equals to the current one, 16.1 Mb/s.

  As demonstrated by the graph, data losses are in inverse proportion to the number of blocks. The more blocks a node has, the bigger amount of data has to be transferred to its successor in order to keep the same replication factor on the node's departure. The bigger amount of data has to be transferred, the slower it is for the transfer to be finished. In any case, the rate of data transfer between two given nodes in the network can't overtake the average Internet speed. In the current state of the system (2200 primary blocks per node), it must depart at least the 30% of the nodes in a day in order to have data loss. By the way, as it has been mentioned in Section 5.2, the number of the full nodes increasing (not decreasing) through time.

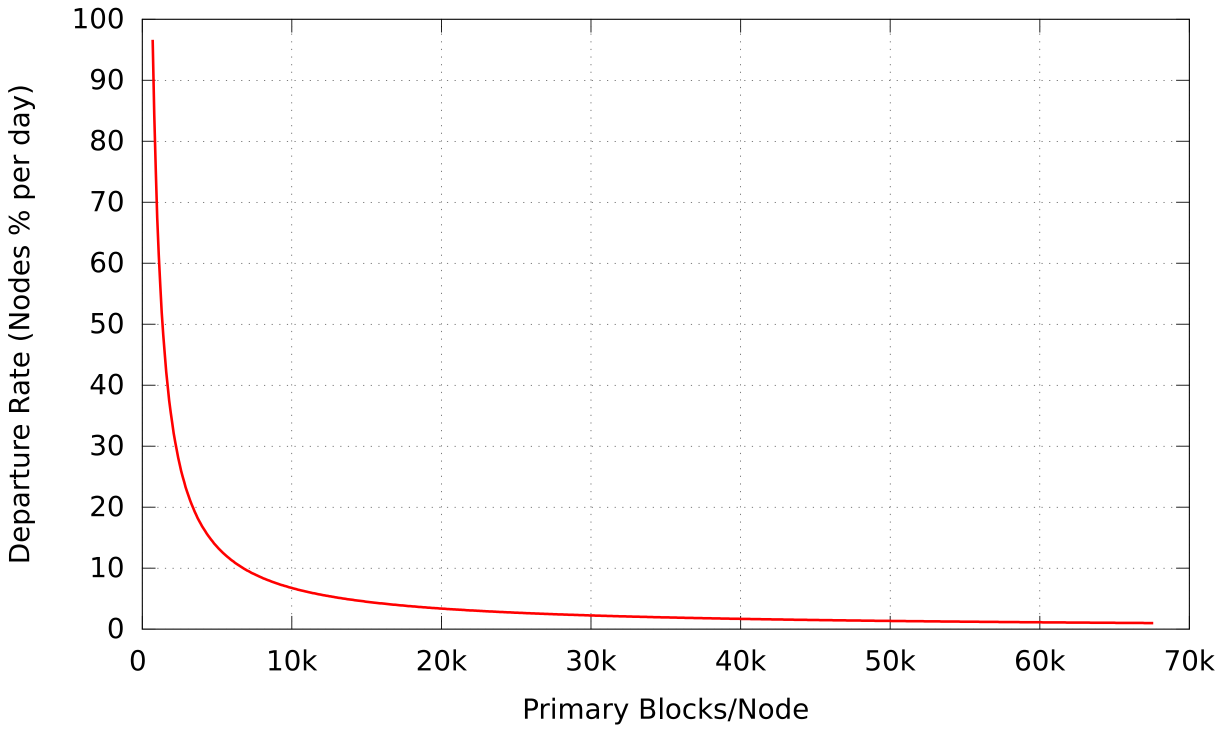

Figure 25: Minimum departure rate for data loss

- **Query Latency**:

  Figure 26 shows the average query latency under the average query rate, in two cases: 32 nodes per segment and 256 nodes per segment. We measure the delays of routing the queries into DHT ring until the right successor to be found. Extra delays due to node failures are not taken into account.

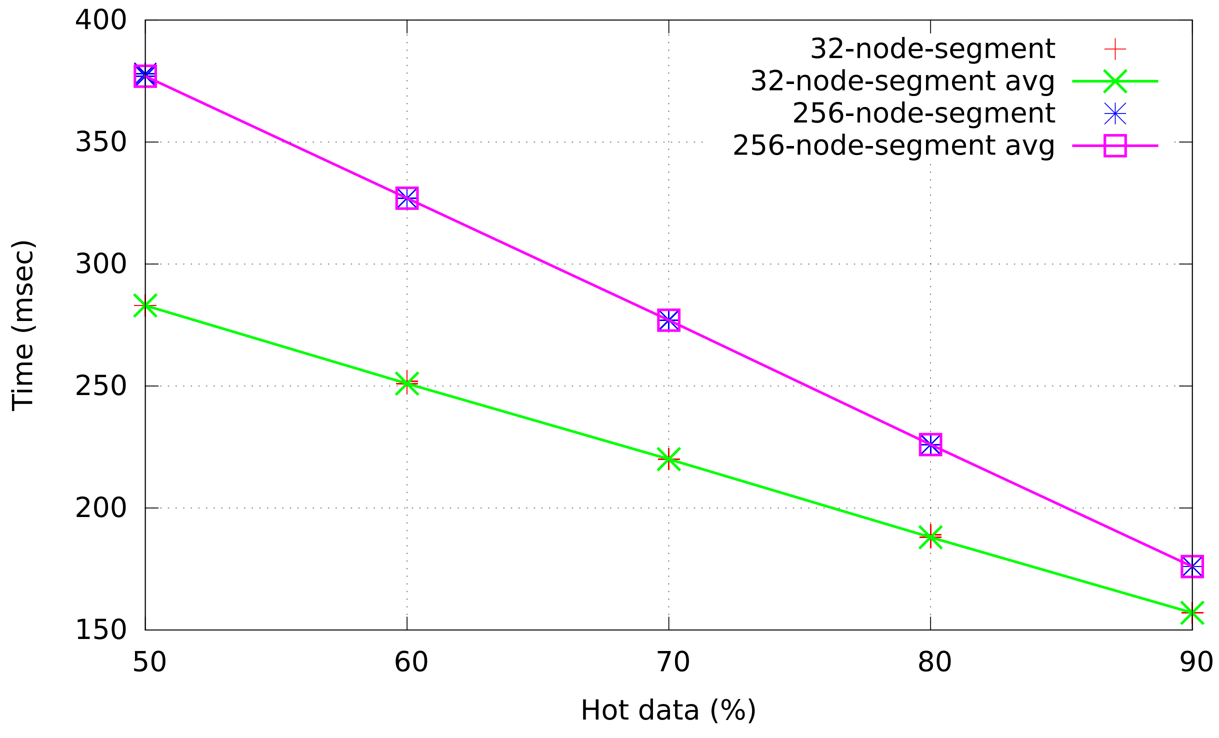

Figure 26: Query latency

In the simulations, we examined the query latency for various distributions of queries between the hot and the cold data. Recall that with the term ”hot data” we refer to the blocks that were created in the last 24 hours. As each node keeps this information locally stored, the corresponding queries are served without the need of forwarding and routing them within the DHT.

32-node-case delays are smaller than 256-node-case delays. This is due to the fact that each node that holds a replica block can answer a query, although the query does not refer to its primary blocks. So, the smaller the segment is, the more blocks a node has. The more blocks a node has, the less likely is for a query to be answered and make another hop. In both cases, the average delay of a query is in the few hundreds of msec and all the samples are so close to the average value. We expect a greater spread of the samples as the segment grows. In the 32-node-case, the maximum number of hops that a query has to do until to be answered is 5. While in the 256-node-case, is 8. Both bounds seem to be low enough in order for the spread of the samples to be significant.

- **Bandwidth Usage**:

  A very important factor is the bandwidth required by a node to participate in the system under study. The total bandwidth used by a node is the result of sending and receiving data for both the Bitcoin and the DHT protocol. In Figure 27, the bandwidth for answering queries (on behalf of the Bitcoin protocol) is not considered. We only measure the bandwidth usage on behalf of the DHT protocol. For instance, data that have to be replicated due to nodes' departure, rearranging keys between the nodes, etc.

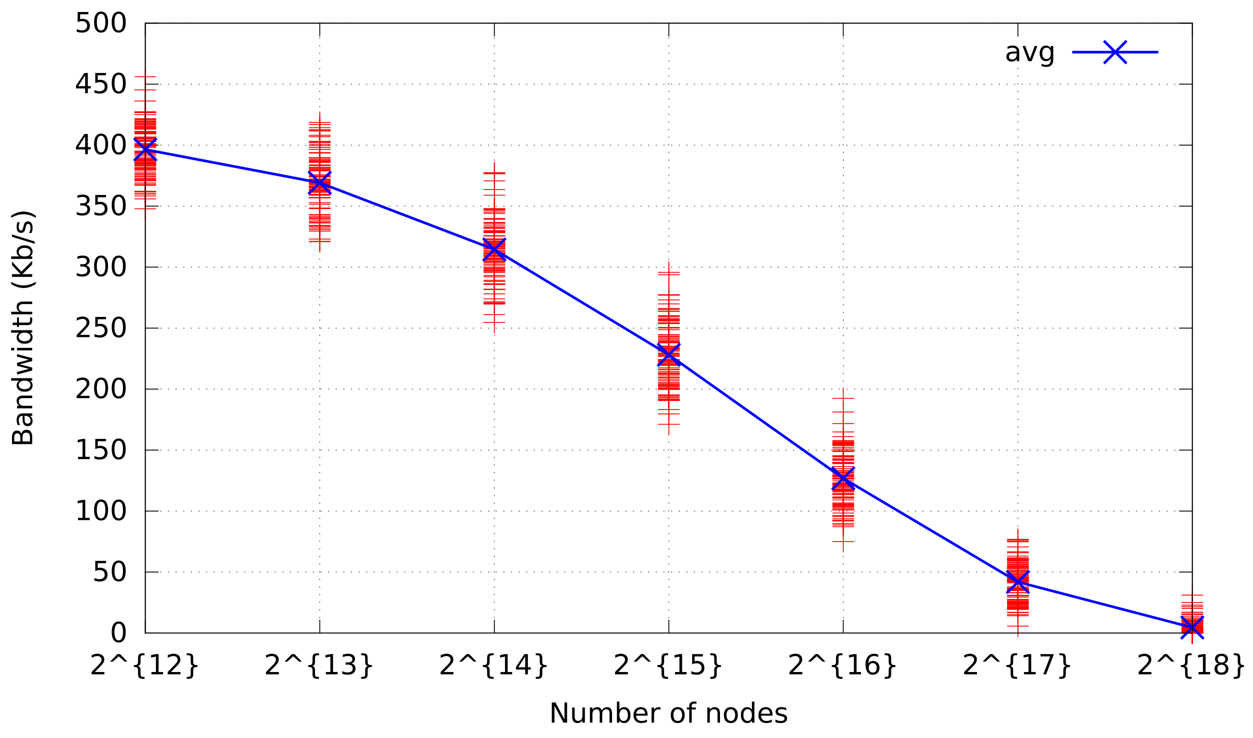

Figure 27: Bandwidth usage per node on behalf of the DHT protocol

As we see, the bandwidth required to keep copies on behalf of the DHT protocol is relative small compared to the average Internet connection speed in Bitcoin network. In fact, it is relatively small even compared to the bandwidth used by a current Bitcoin full node. The average bandwidth for answering queries at Bitcoin network is currently 1.6 Mb/s and the maximum is up to 21 Mb/s. The average Internet connection speed is about 16.1 Mb/s.

Moreover, since the number of the participating full nodes in the network can not be predicted, we use seven different network sizes so we can get a picture of both small and large networks. The graph shows that, although the amount of chain data remains the same in all cases, by increasing the number of nodes in the network, we have less amount of data needed to be transferred due to the DHT protocol. This means that the increase in the number of nodes in the network, it enhances security since each DHT node is replicated to multiple full nodes. That makes the system even more fault-tolerant, meaning that it can continue to operate with higher probability even if a component of the network fails.

5.4 Evaluation of the DHT Blockchain

Measurements made on critical metrics that determine the function of a participating node in a DHT, indicate that such an application could work efficiently. This means that the user of a common machine, to which this proposal refers to, would be able to participate in a ”decentralized full node” without high resource requirements. In particular, making as realistic assumptions as possible based on a current Bitcoin full node and the Bitcoin network, the following conclusions are drawn:

- A significant percentage of capacity savings per node appears even with a small number of participating nodes in our DHT. The capacity savings can easily exceed 95% of the volume of data that a full node stores today.
- Keys (blocks) are evenly distributed among the participating nodes, because of the consistent hashing. Hence, there is an almost perfect load balance between the nodes of the DHT.
- There is no substantial risk of data loss from the DHT, if the replication factor is selected appropriately and the necessary security measures against malicious nodes are taken. In addition, replication helps with congestion, beacause each node that holds a block (primary or replica) can answer a query.
- The query latency in the system is relatively low and predictable. The maximum number of hops that a query can make until to be served is too low, due to DHT’s amazing scalability.
- The resulting bandwidth usage per node, as overhead due to DHT, is relatively low and declines as the number of the participating nodes increases. More full nodes per DHT node means that the DHT node becomes more robust, so it is less likely to fail.
- By imposing storing blocks created in the last 24 hours, in every DHT node, no congestion is observed. Meaning that nodes are not disproportionately burdensome, and the system can provide high data availability.

6. CONCLUSION

6.1 Concluding Remarks

This master thesis has extensively referred to the blockchain, analyzing its technology, referring to its escalation problem and the main solutions proposed so far. In addition, a new proposal was introduced to store the blockchain in a DHT, which aims to reduce the amount of data stored by the nodes of a blockchain network. This suggestion was evaluated based on simulations that were made using a Java-developed tool that involves storing the Bitcoin blockchain in a Chord DHT. The results show that such an implementation is possible, as critical metrics of the nodes of such a system, such as bandwidth and congestion, range in normal levels and escalate by increasing nodes. In fact, according to the study, it is possible to allocate the blockchain into a DHT with significant levels of capacity savings, of 98%.

6.2 Future Work

Regarding future work that can evolve our system, we propose the following:

- Properly evaluate the system's scalability using a cluster of nodes, namely the creation of a DHT Chord. Thus, to be possible to study the scaling achieved for a variable number of nodes and track replica losses of the DHT in real sizes.
- Investigate the ability to combine the logic the ”decentralized full node”, with other scaling solutions like sharding. Can we organize shards of a blockchain in a DHT manner in order to create a global history?

REFERENCES

[1] bitcoin.org. Running A Full Node. https://bitcoin.org/en/full-node#minimum-requirements. Accessed: 2019-02-05.

[2] bitnodes.earn.com. Global Bitcoin Nodes Distribution. https://bitnodes.earn.com/. Accessed: 2019-02-05.

[3] Vitalik Buterin. A next-generation smart contract and decentralized application platform. 2014.

[4] Kyle Croman, Christian Decker, Ittay Eyal, Adem Efe Gencer, Ari Juels, Ahmed E. Kosba, Andrew Miller, Prateek Saxena, Elaine Shi, Emin Gün Sirer, Dawn Song, and Roger Wattenhofer. On scaling decentralized blockchains - (A position paper). In *Financial Cryptography and Data Security - FC 2016 International Workshops, BITCOIN, VOTING, and WAHC, Christ Church, Barbados, February 26, 2016, Revised Selected Papers*, pages 106–125, 2016.

[5] Luke Dashjr. Bitcoin Core nodes percentage. http://luke.dashjr.org/programs/bitcoin/files/charts/software.html. Accessed: 2019-02-05.

[6] John R. Douceur. The sybil attack. In *Peer-to-Peer Systems, First International Workshop, IPTPS 2002, Cambridge, MA, USA, March 7-8, 2002, Revised Papers*, pages 251–260, 2002.

[7] Cynthia Dwork and Moni Naor. Pricing via processing or combatting junk mail. In *Advances in Cryptology - CRYPTO '92, 12th Annual International Cryptology Conference, Santa Barbara, California, USA, August 16-20, 1992, Proceedings*, pages 139–147, 1992.

[8] en.wikipedia.org. List of countries by electricity consumption. https://en.wikipedia.org/wiki/List_of_countries_by_electricity_consumption. Accessed: 2019-02-05.

[9] Anna Irrera and Gertrude Chavez-Dreyfuss. Bitcoin 'clone' sees a slow start following split. https://www.independent.co.uk/news/business/news/bitcoin-cash-latest-news-clone-digital-currency-encrypted-privacy-dark-web-a7872081.html. Accessed: 2019-02-05.

[10] David R. Karger, Eric Lehman, Frank Thomson Leighton, Rina Panigrahy, Matthew S. Levine, and Daniel Lewin. Consistent hashing and random trees: Distributed caching protocols for relieving hot spots on the world wide web. In *Proceedings of the Twenty-Ninth Annual ACM Symposium on the Theory of Computing, El Paso, Texas, USA, May 4-6, 1997*, pages 654–663, 1997.

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

[12] Satoshi Nakamoto. Bitcoin P2P e-cash paper. http://www.metzdowd.com/pipermail/cryptography/2008-November/014823.html. Accessed: 2019-02-05.

[13] Satoshi Nakamoto. Bitcoin: A peer-to-peer electronic cash system. 2008.

[14] U.S. Department of Commerce/NIST. FIPS 180-1, Secure Hash Standard. https://csrc.nist.gov/publications/detail/fips/180/1/archive/1995-04-17. Accessed: 2019-02-05.

[15] Joseph Poon and Thaddeus Dryja. The bitcoin lightning network: Scalable off-chain instant payments. *See https://lightning.network/lightning-network-paper.pdf*, 2016.

[16] QuantumMechanic. Proof of stake instead of proof of work. https://bitcointalk.org/index.php?topic=27787.0. Accessed: 2019-02-05.

[17] raiden.network. Raiden Network. https://raiden.network/101.html. Accessed: 2019-02-05.

[18] Rüdiger Schollmeier. A definition of peer-to-peer networking for the classification of peer-to-peer architectures and applications. In *1st International Conference on Peer-to-Peer Computing (P2P 2001), 27-29 August 2001, Linköping, Sweden*, pages 101–102, 2001.

[19] statoshi.info. Realtime Bitcoin Node Stats. https://statoshi.info/. Accessed: 2019-02-05.

[20] Ion Stoica, Robert Tappan Morris, David R. Karger, M. Frans Kaashoek, and Hari Balakrishnan. Chord: A scalable peer-to-peer lookup service for internet applications. In *SIGCOMM*, pages 149–160, 2001.

[21] Peter Todd. Clearing up some misconceptions about full nodes. https://lists.linuxfoundation.org/pipermail/bitcoin-dev/2016-February/012435.html. Accessed: 2019-02-05.

[22] usa.visa.com. Visa Inc. at a Glance. https://usa.visa.com/dam/VCOM/download/corporate/media/visa-fact-sheet-Jun2015.pdf. Accessed: 2019-02-05.

[23] Pieter Wuille. Segregated Witness (Consensus layer). https://github.com/bitcoin/bips/blob/master/bip-0141.mediawiki. Accessed: 2019-02-05.