Trinity College Dublin
Coláiste na Tríonóide, Baile Átha Cliath
The University of Dublin

School of Computer Science and Statistics

An Efficient Peer-to-Peer Bitcoin Protocol with Probabilistic Flooding

Huy Vu
Supervisor: Dr. Hitesh Tewari

April 12, 2019

A Dissertation submitted in partial fulfilment
of the requirements for the degree of
Master in Computer Science

Declaration

I hereby declare that this project is entirely my own work and that it has not been submitted as an exercise for a degree at this or any other university.

I have read and I understand the plagiarism provisions in the General Regulations of the University Calendar for the current year, found at `http://www.tcd.ie/calendar`.

I have also completed the Online Tutorial on avoiding plagiarism 'Ready Steady Write', located at `http://tcd-ie.libguides.com/plagiarism/ready-steady-write`.

Signed: _____ Date: _____

Summary

Cryptocurrencies are rising in popularity with each passing day. The interest and popularity of cryptocurrencies can be mainly attributed to the excitement and hype surrounding the biggest and most popular cryptocurrency, Bitcoin. Bitcoin was launched in 2009, becoming the worlds first decentralised digital currency. It has since been the source of motivation behind many alternative cryptocurrencies (altcoins) and introduced the world to blockchain, the technology behind which bitcoin is founded upon. A blockchain is an immutable digital ledger in which transactions made in bitcoin or another cryptocurrency are recorded chronologically and publicly. As the popularity of altcoins and in particular bitcoin grows, the number of transactions in the blockchain grows proportionally, meaning that there is a need to make the system as efficient as possible in order to sustain this growth in popularity.

This dissertation aims to make the current Bitcoin system more efficient by targeting the current flooding protocol implemented by Bitcoin. As Bitcoin is a decentralized peer-to-peer network, nodes participating in the network are only aware of their connected neighbours and therefore are not aware of the network as a whole. The flooding protocol implemented by Bitcoin is the mechanism in which information such as transactions and blocks are propagated to all the nodes in the network. However, the current flooding protocol is wasteful, producing a large number of duplicated and redundant messages.

The current flooding protocol works as follows:

1. If Node A wants to broadcast a transaction to the network, Node A will send an *INV* message to its neighbours. The INV message contains the hash of transactions Node A wants to broadcast to its neighbours, rather than the transaction as a whole.
2. If Node A's neighbour, Node B, wants a transaction contained in the INV message sent from Node A, Node B will respond to Node A with a *GETDATA* message, indicating which transactions it requires.
3. However, if Node B does not require any transactions contained in the INV message sent by Node A, Node B will simply ignore the INV message.
4. Node A will send the required transactions outlined in the GETDATA message to Node B through a *tx* message.

The main contribution of this dissertation is the proposal of a novel protocol that aims to reduce the number of redundant and duplicated messages being generated by the current flooding protocol. The proposed protocol changes the current flooding mechanism

employed by Bitcoin to a probabilistic flooding approach. The proposed probabilistic flooding approach is based on the idea that nodes in the Bitcoin network have a wide variance in the number of neighbours they are connected to. Node A may be connected to 8 neighbours, whereas Node B may at the best case, be connected to 125 neighbours. Therefore, if Node A sends an INV message to one of its neighbours that is highly connected in the network, it is unlikely that they will respond to the INV message with a GETDATA message as they are likely to have received the transactions contained in the INV message already from their other neighbours. However, it is likely that if Node A sends an INV message to a less well-connected neighbour, they will with a higher probability, respond to the INV message with a GETDATA message when compared to the neighbour that is well-connected. Therefore, Node A will send an INV message to its neighbours based on a calculated probability. The probability is based on the number of INV messages sent to its neighbour and the number of GETDATA messages received in response to the INV message. This idea forms the basis of the proposed novel protocol presented in this dissertation.

The proposed protocol was evaluated through the results of multiple simulations. The simulation simulates 58 hours of Bitcoin network usage. The proposed probabilistic flooding protocol was implemented in the simulation and was evaluated against the results obtained from the simulation when the current flooding bitcoin protocol was implemented. The results showed that the probabilistic flooding approach reduced the number of INV messages sent per node and the total number of messages sent per node by **29%** and **14%** respectively. It is important that along with the reduction of the number of messages, the integrity and reliability of the system remained unchanged from the proposed changes. The results show that along with the large reduction of messages on the network, the system maintained its 100% transactions commit rate and the time taken for a transaction to be committed remained unchanged. The results indicate that a reduction of redundant messages is possible whilst maintaining the integrity and reliability of the system, therefore meeting the main objectives of this dissertation.

Abstract

An Efficient Peer-to-Peer Bitcoin Protocol with Probabilistic Flooding

Huy Vu

The main aim of this dissertation is to adjust the flooding protocol that is currently being implemented within the bitcoin network in order to reduce the number of redundant messages that each peer in the network receives. The flooding protocol is adjusted to take a probabilistic approach in which a node will send their transactions or blocks to their neighbours based on a calculated probability.

The probabilistic flooding approach was implemented within a bitcoin simulation. The simulation results showed that the number of redundant messages exchanged within the network and duplicated messages received by nodes had been reduced without any negative impact on the number of transactions or blocks committed.

Acknowledgements

First and foremost, I would like to express my sincere gratitude to my supervisor, Dr. Hitesh Tewari, for his continued support, knowledge and guidance throughout this dissertation, without which this dissertation would not have been possible.

I would like to thank my family for their constant support, understanding and encouragement throughout all my endeavours in life.

I would also like to thank my peers for their camaraderie and support throughout my time in college.

Contents

1 Introduction **1**

1.1 Motivation . . . 3

1.2 Dissertation Structure . . . 3

2 Bitcoin Background **5**

2.1 Bitcoin introduction . . . 5

2.2 Ownership - Keys and Addresses . . . 6

2.3 Transactions . . . 7

2.4 Blocks . . . 10

2.5 Mining . . . 11

2.5.1 Proof-of-Work . . . 11

2.5.2 Difficulty . . . 12

2.5.3 Incentives . . . 12

2.5.4 Forks . . . 13

2.5.5 Mining pools . . . 14

2.6 The Blockchain . . . 15

2.6.1 Merkle Tree . . . 16

2.6.2 Simplified Payment Verification . . . 16

2.7 The Bitcoin Network . . . 18

2.7.1 Network Discovery . . . 18

2.7.2 Neighbour Selection . . . 21

2.7.3 Information Propagation . . . 22

2.8 Bitcoin Security . . . 25

3 Related Work **31**

4 Probabilistic Flooding **36**

4.1 The Problem . . . 36

4.2 Probabilistic Flooding . . . 39

4.3 Implementation . . . 41

5 Evaluation **44**
5.1 Results 45
5.1.1 Percentage of committed transactions 45
5.1.2 Transaction Commit Time 46
5.1.3 Total Sent Messages 46
5.1.4 Other Metrics 47
5.1.5 Varying the Probability Percentage 49

6 Conclusion **52**

List of Figures

2.1 Generating a Bitcoin Address from a Private Key. 7
2.2 Example of Transaction Showing How Change is Generated. 8
2.3 Example Transaction 9
2.4 Example blocks in the blockchain [1] 11
2.5 Number of Bitcoins in Circulation [2] 13
2.6 Hashrate Distribution Amongst the Largest Mining Pools. [3] 15
2.7 Blockchain representation example [4] 16
2.8 Merkle Tree example [4] 17
2.9 Initial Handshake [5] 20
2.10 Address Propagation and Discovery [5] 21
2.11 Algorithim for Selecting a New Outbound Connection [6] 22
2.12 Message Exchange For The Propgation of Information across the Bitcoin Network [7] 23
2.13 Three Methods of Block Relay [8] 25
2.14 An example of the double-spending attack 27
2.15 Selfish-Mining [9] 28
2.16 A simple example of an eclipse attack [9] 29

3.1 Message Exchange After the Changes Described are Applied [7] 32

4.1 Messages Exchanged between Two Nodes for Information Propagation . 37
4.2 Node Receiving Duplicate INV Messages for the Same Transaction . . . 38
4.3 Probabilistic Flooding Example 40

5.1 Comparison of the Percentage of Transactions committed. 45
5.2 Comparison of the average time taken to commit a transaction between the two protocols. 46
5.3 Comparison of the average total sent messages per node. 47
5.4 Comparison of the total number of sent messages during the simulation period between the two protocols. 47
5.5 Comparison of the number of blocks created during the simulation period. 48

5.6 Comparison of the number of forks created during the simulation period between the two protocols. 49
5.7 Comparison of the total number of sent messages. 50
5.8 Comparison of the total number of uncommitted transactions. 50
5.9 Chart indicating the exponential increase of uncommitted transactions. . 51

1 Introduction

Cryptocurrencies, of which Bitcoin is the most popular, have risen greatly in popularity in recent times. A cryptocurrency is a digital currency designed to work as a medium of exchange, without the need for a central authority. Cryptocurrencies make use of strong cryptography in order to secure and verify transactions. With the popularization of cryptocurrencies comes an increase in daily users. As a result of this increase in daily users, countless more transactions are made within the network leading to an increase in network resources and power consumption by the systems in order to maintain the cryptocurrencies. For example, Bitcoin between 2011 and 2012 averaged approximately 7,000 transactions per day, but at the time of writing, Bitcoin currently averages approximately 270,000 transactions per day, with the daily trading value estimated at around $500 million [3].

There are a number of reasons as to why cryptocurrencies have increased in popularity in recent times:

1. **Financial Gains** - Cryptocurrencies interests many individuals, especially those in the financial sector, as they are very similar to equities. Unlike fiat currency that is declared legal tender by a government, cryptocurrencies are not legal tender and not backed by the government. Hence, the value of cryptocurrencies is set by supply and demand. This leads to a very unpredictable and volatile market, where the investment returns can be enormous. For example, Bitcoin showed an approximate increase of 1,500% while Ethereum showed an increase of over 10,000% in 2017. [3]
2. **Technology** - Never before in human history has there been a way to digitally transfer currency between two parties without relying on a trusted intermediary. The underlying technology behind which Bitcoin and many other cryptocurrencies are founded upon is known as blockchain. Blockchain was introduced to the world in 2009 by the pseudonym Satoshi Nakamoto, in his paper 'Bitcoin: A Peer-to-Peer Electronic Cash System' [4]. A blockchain is an immutable digital ledger in which transactions, made in Bitcoin or any other cryptocurrency that uses blockchain technology, are recorded chronologically and

publicly. This distributed ledger technology allows for a more open and trustworthy way for people and companies to store information in a decentralized fashion. This revolutionary technology has attracted many developers and technologically-interested individuals as developers are now able to create decentralized applications (dApps) on top of existing blockchain-based platforms such as Ethereum[10].

3. **Psychology** - The main attribute of cryptocurrencies is that there is no central authority. Individuals who advocate for freedom and dislike the control that governments or companies have can find solace in cryptocurrencies. With cryptocurrencies, the user is solely responsible for the management and security of their funds. Another issue that is tackled by cryptocurrencies is privacy. If an individual wants to stay anonymous, cryptocurrencies like Bitcoin or Monero offer tremendous value to protect the individual's online identity.

4. **Word-of-mouth** - With massive popularity in recent times, it would be difficult to find someone who has not heard of Bitcoin or cryptocurrencies. With such enthusiasm and excitement surrounding cryptocurrencies, individuals are intrigued to learn more about them. Whether it be the enormous return on investments that are being generated or the innovative technology that underpins how these cryptocurrencies work, there are many reasons why the cryptocurrency industry is gaining so much traction.

Due to the rise in popularity of Bitcoin and other cryptocurrencies, there is a need to make the underlying system of these cryptocurrencies as efficient as possible in order to sustain this growth.

Bitcoin and other blockchain-enabled cryptocurrencies usually work as follows:

- Users exchange digital assets between each other through transactions.
- Transactions are placed together in blocks.
- Blocks are linked together in a chain, creating what is known as the blockchain.
- The linking of the blocks establishes a chronological order of the blocks and transactions, forming a distributed ledger.

As Bitcoin is a peer-to-peer (P2P) network, transactions and blocks are relayed to each peer in the network through a mechanism known as **flooding**. Flooding is when a node receives a new transaction or block, they will attempt to send it to their connected neighbour peers, who will then, in turn, send it to their neighbours, who will then send it to their neighbours etc., until the transaction or block has reached every peer in the network.

The broadcasting of transactions and blocks usually requires the use of three messages - an INV message, GETDATA message and a tx/block message. If there were only two nodes in the Bitcoin P2P network, and we wanted to broadcast the 270,000 daily transactions to the network, this would require at the worst-case, 810,000 exchanged messages between the two peers. However, this number is much bigger in the real Bitcoin network, as there are approximately over 10,000 nodes in the network [11]. With over 10,000 nodes in the network, and with each node required to forward a message to all of its neighbours due to the flooding protocol, a lot of redundant and duplicated messages will be generated in the network.

In this dissertation, we explore a way to adjust the current flooding protocol that is currently implemented in the Bitcoin network to a probabilistic approach with the aim to reduce the number of redundant messages that are currently being generated whilst still maintaining the integrity and correctness of the system.

1.1 Motivation

The main objective of this dissertation is to reduce the number of redundant and duplicated messages that are currently being generated in the Bitcoin network. Reducing the number of redundant messages that are generated and received by nodes in the network is beneficial to the node as it will help decrease unnecessary power consumption and CPU cycles being wasted on duplicated messages. Within the network as a whole, reducing the number of redundant messages being broadcast throughout the network will help reduce the bandwidth consumption.

Whilst changing the flooding protocol, we need to also ensure that we are not affecting the integrity, security or the distributed nature of the system in any way. As we are changing the protocol in which information is disseminated in the Bitcoin network, it is important to ensure that all valid, created transactions are confirmed in blocks eventually. It is also important to ensure that the system is still resilient to the major security issues of Bitcoin such as the 51% attack or double-spending, as explained in section 2.8.

1.2 Dissertation Structure

The dissertation is structured as follows:

1. **Chapter 2** will describe the Bitcoin system in detail, outlining and explaining the main building blocks of Bitcoin. The objective of this chapter is to give the reader the necessary background information regarding Bitcoin and to briefly

outline some of the issues the Bitcoin system faces.

2. **Chapter 3** discusses recent and related work associated with trying to improve the Bitcoin system or potential P2P network improvements that could be applied to the Bitcoin system.
3. **Chapter 4** outlines the problem identified currently in the Bitcoin system and describes the suggested protocol changes proposed - the protocol changes that form the basis of this dissertation.
4. **Chapter 5** describes the results gathered from the simulations. Compares and contrasts the results received from simulating the normal Bitcoin protocol and the adjusted Bitcoin protocol that implements the proposed protocol changes.
5. **Chapter 6** concludes the dissertation by summarising the objectives of the dissertation, whether or not the objectives were achieved and possible future work.

2 Bitcoin Background

2.1 Bitcoin introduction

Bitcoin was the world's first decentralized digital currency, proposed in 2008 under the pseudonym Satoshi Nakamoto in their paper titled 'Bitcoin: A Peer-to-Peer Electronic Cash System' [4]. The objective of Bitcoin is to create a means of exchange, without dependence on a central authority, that could be transferred electronically in a secure, verifiable and immutable way. The most important attribute of Bitcoin is the decentralization nature of it - the lack of dependence on a central server or trusted parties. As mentioned in a forum post shortly after Bitcoin was launched, Satoshi wrote that 'The root problem with conventional currency is all the trust that's required to make it work.' [12]

In the Bitcoin network, users are able to exchange bitcoins for a multitude of purposes, including buying and selling goods, sending money to people or charities or buying services. Bitcoin can also be sold, purchased and exchanged for other cryptocurrencies at cryptocurrency exchanges. Bitcoin is an entirely virtual currency, with no physical bitcoins in existence. In order to exchange bitcoins between two parties, both parties must have their own Bitcoin address, which is derived from their public/private key-pair. These keys prove ownership of bitcoins in the network. With these keys, the payer can digitally sign transactions which unlock the value of the bitcoin and transfer the ownership of the currency to the payee. These transactions are then registered in an immutable ledger known as the blockchain. The blockchain consists of a chain of linking blocks, where each block is added approximately every ten minutes and contains the transactions made within the network. The blockchain, combined with a P2P network that uses proof-of-work (PoW)[4] as its consensus mechanism to record a public history of transactions is the method in which Bitcoin prevents double-spending in the network.

The network operates as follows, as outlined directly from Satoshi Nakamoto's original paper:

1. New transactions are broadcast to all nodes.
2. Each node collects new transactions into a block.
3. Each node begins to mine their block
4. When a node successfully mines their block, it broadcasts the block to all nodes.
5. Nodes accept the block only if all transactions in it are valid and not already spent.
6. By updating their ledgers with this block, miners will begin mining the next block on top of this block.[4]

2.2 Ownership - Keys and Addresses

Bitcoin is based on cryptography. Cryptography and encryption are often thought of as synonyms for each other, however, cryptography encompasses a lot more than just encryption. Bitcoin uses cryptography to prove knowledge of a secret (digital signature) and to prove the authenticity of data (digital fingerprint). These cryptographic proofs and mathematical tools are key to how Bitcoin operates.

Bitcoin uses public key cryptography in order to create a key pair that allows users to claim ownership and spend their bitcoins. The key pair consists of a private key and a public key. The public key is used to derive a user's Bitcoin address from which they can receive funds, whereas the private key is the key that allows users to spend the funds by digitally signing transactions. As key pairs are used to store and spend funds in the Bitcoin system, every user that wants to use bitcoin must have their own public/private key-pair.

The private key used in Bitcoin is a 256-bit number. The public key is derived from the private key through elliptic curve multiplication. Elliptic curve multiplication is a one-way cryptographic function, meaning that we can generate the public key from the private key but not vice versa [13]. The public key is then passed through a one-way cryptographic hash function (SHA-256 & RIPEMD-160), as well as using Base58Check encoding, to generate a user's Bitcoin address [14]. Figure 2.1 shows the process of generating the Bitcoin address from a private key.

As we will see in further detail in the next section, when sending bitcoins through the Bitcoin network, a new transaction must be created. The payer must present their public key as well as digitally signing the transaction with their private key. Through this method, everyone in the Bitcoin network may now verify and accept the transaction as valid as the digital signature verifies the authenticity of the transaction.

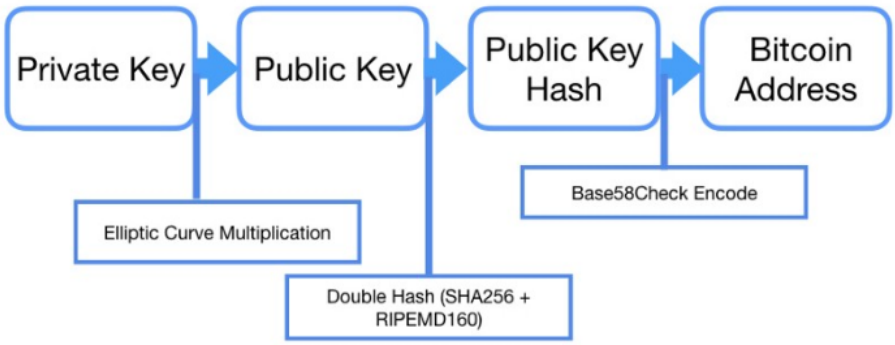

Figure 2.1: Generating a Bitcoin Address from a Private Key.

A Bitcoin wallet is normally used to store a user's key-pair. However, storing just the private key is also possible as you can generate your Bitcoin address from just the private key. Ownership of the private key allows full control of the Bitcoin address associated with that private key. The private key can be used to move the funds associated with the corresponding Bitcoin address by creating the digital signature that is required by transactions. It is important to keep the private key secret at all times as allowing third parties access to the private key will give them control of the funds associated with the key. If a user loses their private key through accidental loss, unless they have the key backed up, there is no method to be able to retrieve the private key and the funds associated with the key.

2.3 Transactions

Transactions are the most important part of the Bitcoin system. Everything in the Bitcoin system from mining to the blockchain is designed so that transactions are able to be created, propagated, validated and added to the globally distributed ledger (the blockchain).

Transactions can be broken down into *transaction outputs*, *transaction inputs* and the *transaction ID*. The transaction inputs are the accounts of the payers and the transaction outputs are where the bitcoins are being sent to i.e. the payee's account. The transaction ID uniquely identifies each transaction [7]. Transaction outputs are fundamental in Bitcoin transactions. Transaction outputs are indivisible chunks of Bitcoin currency that are recorded on the blockchain and are seen as valid and spendable by the Bitcoin network. Unspent transaction outputs or *UTXO* are available and spendable transaction outputs. A user's Bitcoin 'balance' is the sum of all the UTXO associated with the user's Bitcoin address.

Just as Euros can be broken down to two decimal places, transaction outputs can be broken down to eight decimal places, known as satoshis. A satoshi represents the

smallest unit of Bitcoin currency, it is one hundred millionth of a Bitcoin (0.00000001 BTC) [15]. However, an important characteristic of transaction outputs is that once it is created, it is indivisible and must be consumed in its entirety during a transaction. If a UTXO is larger than the value of a transaction, it must still be consumed in its entirety and change will be generated. For example, if Alice wants to send Bob 7BTC, but she has a UTXO that is worth 5BTC and another UTXO worth 5BTC, she can combine the two UTXO to satisfy the transaction value. The transaction between Alice and Bob will consume Alice's combined UTXO worth 10BTC, sending 7BTC to Bob's Bitcoin address and generating 3BTC as change back to Alice's address. Figure 2.2 shows this transaction example.

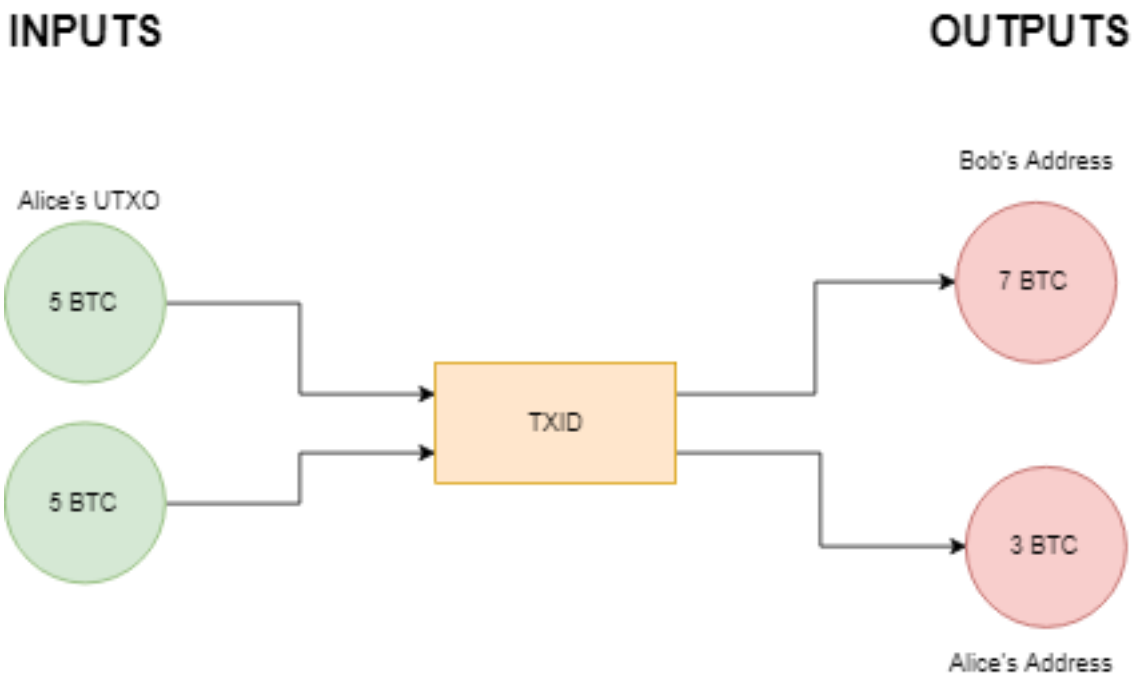

Figure 2.2: Example of Transaction Showing How Change is Generated.

Transactions consume UTXO which in turn creates new transaction outputs that can be spent by the payee. Every output of a transaction will also contain one or more inputs that indicate where the Bitcoin originated from before the transaction. This list of inputs in every transaction creates a chain of previous owners, which you can follow and will eventually lead you back to the *coinbase transaction*.

The coinbase transaction is a special transaction which contains no inputs. It is the first transaction of any given block. It is placed there by the miner who successfully mined the block and creates brand-new, unspent bitcoins that will be sent to the mining node as a reward for mining the block. This is the method by which the entire supply of bitcoins is generated.

In order to transfer bitcoins to an account, the public key of the payee's account must

be listed as the destination of the transaction. The payer must also sign the transaction. They do this by digitally signing a hash of the previous transaction and the public key of the next owner as shown in Figure 2.3 [4]. The previous transactions in Figure 2.3 represent where the payer received the current bitcoins they are using in the current transaction.

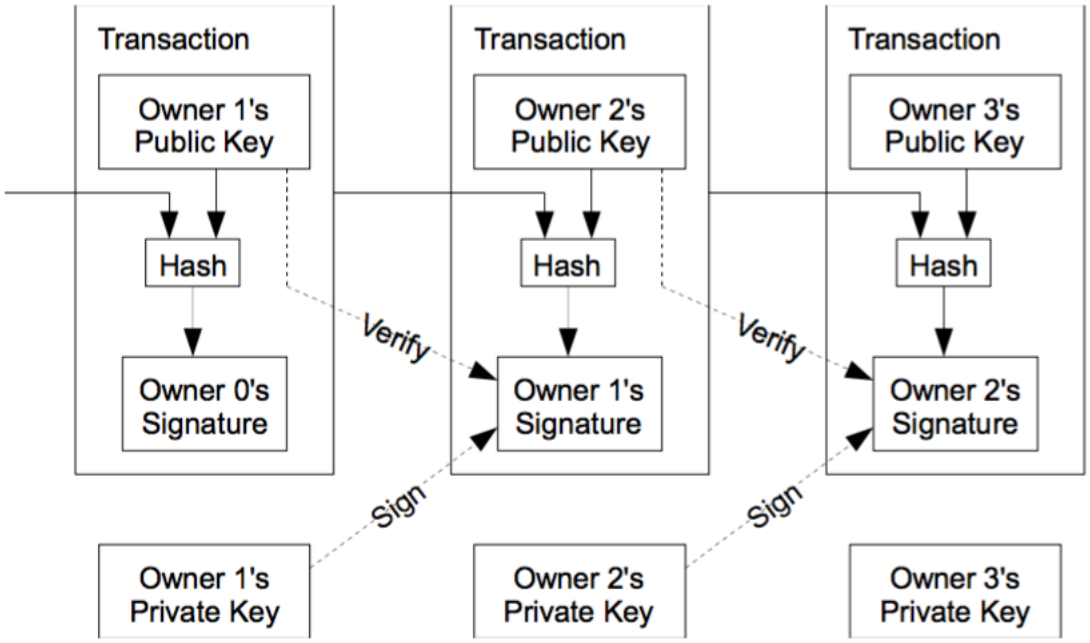

Figure 2.3: Example Transaction [4]

In order for a transaction to be valid, the following criteria must be fulfilled by the outputs claimed and created:

- An output may be claimed at most once.
- New outputs are created solely as a result of a transaction.
- The sum of the values of the inputs has to be greater than or equal to the sum of the values of the newly allocated outputs [1]. [7]

As new transactions are being propagated through the Bitcoin network, the state of the local ledger is constantly changing. Transactions are propagated through the network to keep the nodes informed of the latest UTXO. As the Bitcoin network is a P2P network, transactions may reach different nodes at different times. This may introduce inconsistencies across nodes local ledgers. A nodes local ledger may:

- Receive a transaction sending bitcoins from a certain bitcoin address but may not yet have received the transaction which made those coins available to that address.

[1] If inputs > outputs, miners can collect the difference as a transaction fee or may be sent back to the payee's address as change.

- Receive multiple transactions from the same bitcoin address that attempt to spend the same coins multiple times. This is known as the double-spending attack. [7]

The double-spending attack occurs when a user attempts to spend the same coins multiple times. In real life, a double-spending attack would be equivalent of a user attempting to send multiple transactions to their bank, attempting to spend that balance multiple times. In this situation, the bank will recognize the attempt to spend the balance multiple times and decline it. However, as Bitcoin is a decentralized system, attempting to prevent double-spending is far from trivial. When a node receives the first transaction of multiple transactions attempting to spend the same coins, they will verify and validate that transaction. As more transactions attempting to spend the same coins are received by the node, the node will reject the transactions as the outputs have already been spent by the first transaction. However, it is not guaranteed that nodes in the network will receive conflicting transactions in the same order. As a result, the nodes will disagree about the validity of the conflicting transactions and all other transactions that were built on top of the claimed outputs.

Therefore it is extremely important that a common order of transactions are agreed among all the nodes within the network. We will see how Bitcoin solves this problem in the following sections.

2.4 Blocks

A block is a data structure that is composed of a set of transactions and a block header. Blocks on the blockchain are identified via the block header hash. The block hash is calculated by running the block header through the SHA-256 algorithm twice [15]. The block header is 80 bytes long and consists of the following six fields [15]:

Size	Field	Description
4 bytes	Version	The bitcoin version number
32 bytes	Previous Block Hash	The previous block header hash
32 bytes	Merkle Root	A hash of the root of the Merkle tree of this block's transaction
4 bytes	Timestamp	The timestamp of the block in UNIX
4 bytes	Difficulty Target	The difficulty target for the block
4 bytes	Nonce	The counter used by miners to generate a correct hash

The block size limit of blocks in bitcoin is limited to 1MB per block, therefore there is a maximum number of transactions allowed in each block. Currently, there is an

average of approximately 2,000 transactions included per block [3].

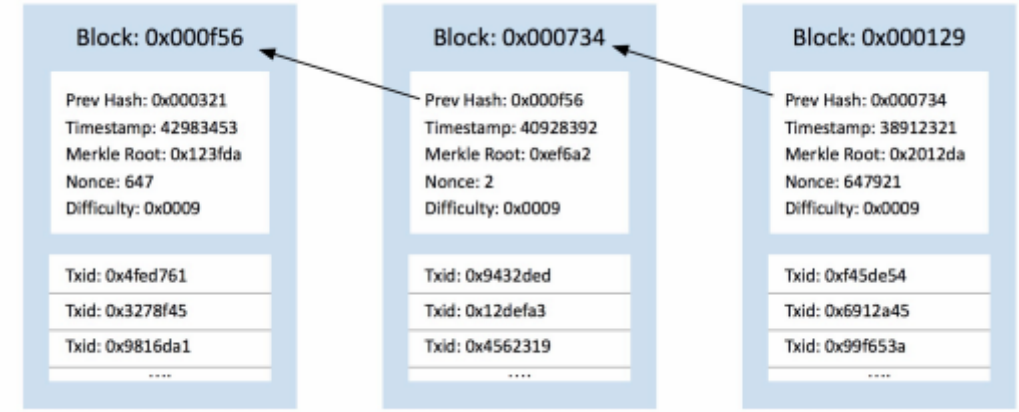

Figure 2.4: Example blocks in the blockchain [1]

2.5 Mining

When a new transaction is propagated through the Bitcoin network, it is stored in each node's local mempool. The Bitcoin mempool is a pool of unconfirmed transactions in the Bitcoin network. Each node has their own mempool. The transactions in the mempool may be valid transactions but are not yet confirmed by the Bitcoin network. The transactions are not seen as confirmed transactions until they are included in a block that is on the blockchain. The process in which transactions are taken from the mempool and included in blocks is known as *mining*.

As mentioned in section 2.4, nodes in the network may not receive transactions in the same order which leads to inconsistent and conflicting transactions. There is a need for a common order of transactions to be agreed among all the nodes in the Bitcoin network. Mining, along with blockchain, are the mechanisms used in Bitcoin which allows for a network-wide consensus without the need for a central authority.

As seen in section 2.5, transactions are bundled into data structures known as blocks. The block is then added to the blockchain, the public ledger of all transactions that every node in the network agrees upon to be valid [2].

2.5.1 Proof-of-Work

The process of mining a block is a computationally difficult process. The nodes which attempt to mine a block, known as miners, must find the solution to Bitcoin's PoW problem. The PoW problem consists of finding an integer value, known as a *nonce*, that when combined with the block header, will provide a hash with a given number of

[2] Nodes may have temporary conflicting ledgers caused by forks, as explained in section 2.5.4.

leading zeroes, known as the difficulty [7]. As cryptographic hashes are a one-way function, the only solution for miners to find the nonce that will satisfy the difficulty of the block is to use a brute-force approach, testing different values for the nonce until a suitable hash is found. The nonce which satisfies the difficulty check of the block, known as the golden nonce, is therefore very difficult to find but once found, is straight-forward to verify it. When mining a block in bitcoin, there is no such thing as being '15%' to solving a block; a miner's chance of mining a block is the same as it was when they started mining the block [1].

2.5.2 Difficulty

The Bitcoin network attempts to produce a block, on average, every ten minutes[3] [4]. As the hash power of the Bitcoin network increases, the rate at which blocks are solved would increase and vice versa, as the hash power of the Bitcoin network decreases, the rate at which blocks are solved would also decrease. The difficulty to solve a block is therefore adjusted every 2016 blocks or approximately every two weeks. If the 2016 blocks solved during the two week period is solved faster than average, the Bitcoin difficulty increases proportionally. If it took longer than average to solve the 2016 blocks, the difficulty is decreased proportionally.

2.5.3 Incentives

As mentioned in section 2.3, a special type of transaction known as the coinbase transaction is the first transaction in any given block. The coinbase transaction is the reward given to a miner for successfully mining a block. As the Bitcoin protocol specifies that there can only be 21 million bitcoins mined in total [15], the amount of bitcoins awarded to the miner as part of the coinbase transaction decreases over time as mining is the sole method in which bitcoins are created. When the first block, known as the genesis block, was mined by Satoshi Nakamoto on January 3rd 2009, the block contained just one transaction - the 50 BTC reward for mining it. The Bitcoin reward for mining a block is halved every four years or every 210,000 blocks. The first Bitcoin halving occurred on November 28, 2012 when block 210,000 was solved [15]. The Bitcoin reward was then changed from the original 50BTC to 25BTC until block 420,000 was solved. The next Bitcoin halving date is expected to be on May 24th, 2020 where the coin reward will be decreased from 12.5 BTC to 6.25 BTC. It is estimated that the last bitcoin to be mined will be in 2140.

As mentioned in section 2.4, the block size limit is 1MB, therefore there is a maximum

[3] Ten minutes was chosen as a compromise between block rate and the risk of a fork in the chain occurring.

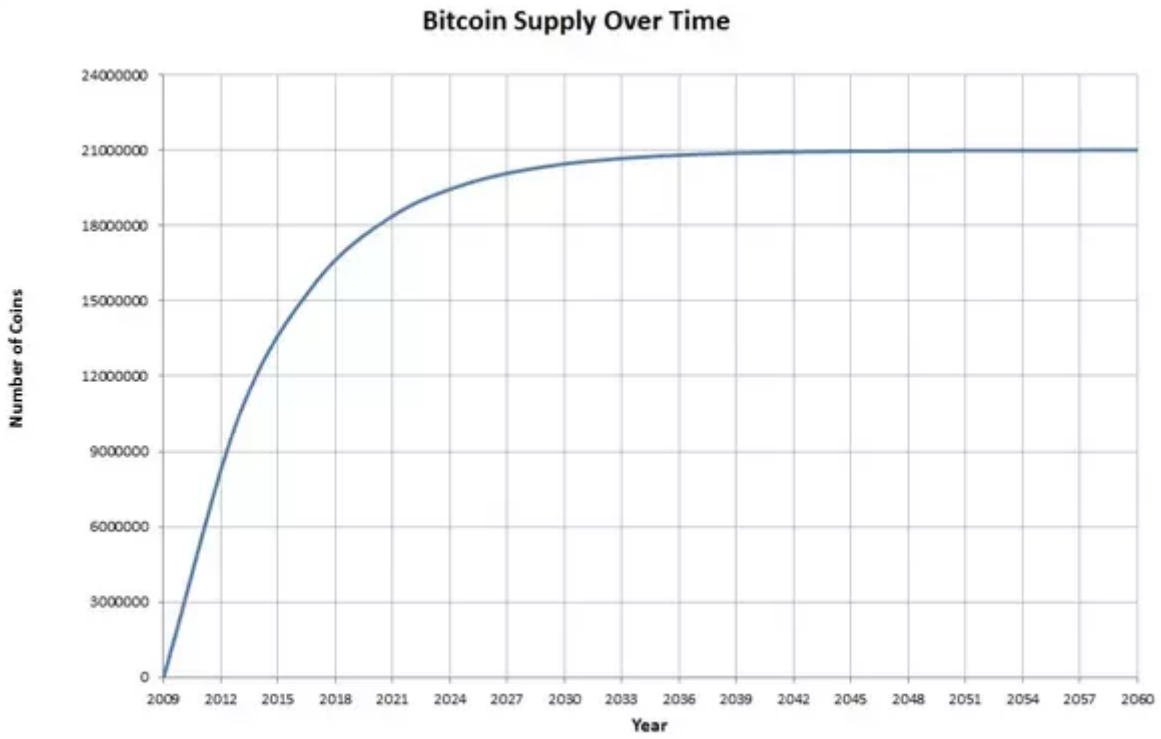

Figure 2.5: Number of Bitcoins in Circulation [2]

number of transactions that can be placed in a block. Block rewards are not the only incentives for miners as miners also collect *transactions fees*. Every transaction may have a transaction fee attached to it - the higher the transaction fee, the higher priority the transaction has to be placed in the next block by the miner. If there are two transactions of similar byte size but only one will fit in the block, the miner will prioritize the transaction that has the higher attached transaction fee. The transaction fees associated with every transaction the miner includes in the block they mined will be collected as payment by the miner. As mentioned above, the mining reward for mining a block will continue to halve every four years until the 21 million Bitcoin limit is hit and there are no more Bitcoin left to mine. When this happens, transaction fees are then left as the only incentive for miners to continually mine blocks. It is important to note that the purpose of mining is not to create new bitcoins but to provide an incentive for nodes in the network to mine blocks which maintain the network's security.

2.5.4 Forks

As the Bitcoin network is a decentralized P2P network, nodes within the network may sometimes have an inconsistent view of the blockchain. This inconsistency may occur when two nodes in the network discover and propagate different blocks at approximately the same time. The two different blocks will propagate through the

Bitcoin network, arriving at nodes at different times. The nodes will accept the first block that they received and reject but save the other block when they eventually receive it [4]. Nodes in the network will now have a temporary inconsistent view of the blockchain, as there are now two blocks claiming to be the blockchain head. In order to resolve this inconsistency, nodes can work on either branch of the fork but are likely to work on the block that they received first [16]. The fork would likely be decided when the next block is mined and one branch becomes longer than the other. The longer branch will become the correct one and nodes working on the other branch will then switch to the longer one. This occurs as nodes always consider the longest chain to be the correct one and will keep working on extending it [4].

The fork may not always be resolved after the first block after the fork is found, it may be prolonged by the partitions of the network finding more blocks h+1, h+2, h+3... until one branch becomes longer than the other. The partitions that did not adopt the branch will then switch over to the longest branch. At this point, the blockchain fork is resolved and there is no longer an inconsistency among the nodes in the network. The blocks that are discarded when the fork is resolved are known as *orphan blocks* [7].

2.5.5 Mining pools

As mining is extremely competitive and it is very unlikely that individual miners by themselves will find the solution to the proof-of-work to produce the next block; miners are encouraged to join mining pools. Miners collaborate with each other by pooling their hashing power and sharing the rewards among each other. By participating in a mining pool, you receive a smaller reward for mining the block as opposed to mining the block individually. However, the frequency in which you are rewarded is higher when participating in a mining pool as the miner's hash power in the pool are combined, leading to a higher probability of mining the next block.

Mining pools, although helpful to the average miner, unfortunately, concentrate a lot of the networks hashing power to the mining pool's owner. The issue of having such a large concentration of the network's hashing power is the potential of the 51% attack, which will be discussed in more detail in section 2.8.

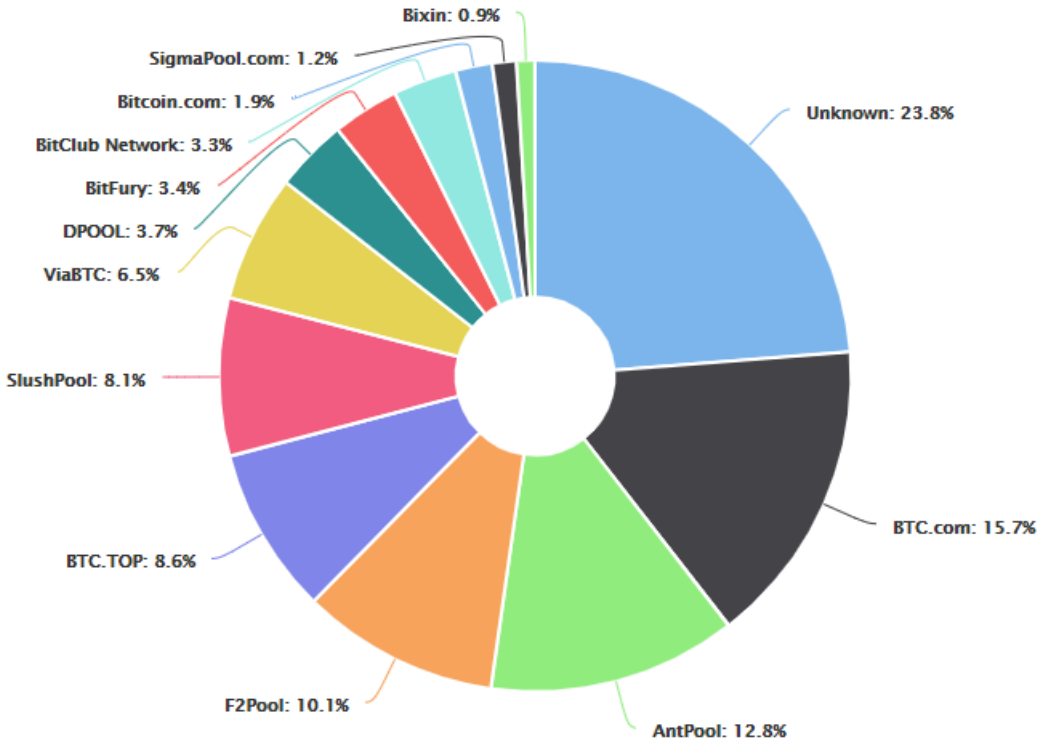

Figure 2.6: Hashrate Distribution Amongst the Largest Mining Pools. [3]

2.6 The Blockchain

Thus far, when blocks are mined and transactions are placed in the blocks, the blocks do not offer any synchronization or chronological ordering of the transactions. However, this changes when blocks are linked together sequentially, creating a chronological ordering over the blocks and therefore the transactions in the blocks [7]. This sequential formation of blocks is known as the blockchain [17]. When a block is created and propagated through the network, it is added to the blockchain by creating a reference to the latest block on the blockchain (the previous block). The chaining of each block to the previous block is what creates a chronological ordering of transactions in the network. The referenced previous block is known as the *parent* block. Blocks may only have one parent block, but can temporarily have multiple children during a blockchain fork, as seen in section 2.5.4. As every block references the previous block, the blockchain is made up of a single sequence of blocks from the first block, or the genesis block, to the latest generated block [17]. The blocks can be thought of as a directed tree, with the genesis block being the root of the tree and the subsequent blocks represented as leaves of the tree [7]. The distance between a block and the genesis block is referred to as its *block height*, and the block that is furthest

away from the genesis block is known as the *blockchain head* [7].

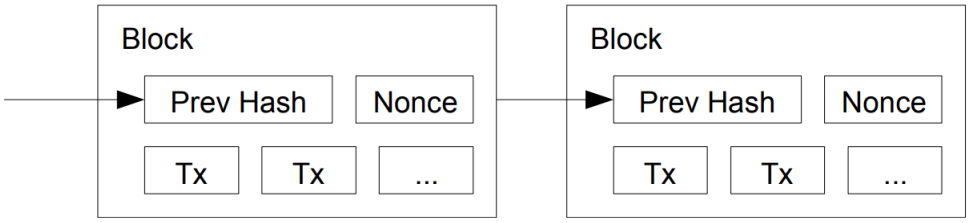

Figure 2.7: Blockchain representation example [4]

2.6.1 Merkle Tree

The set of transactions included in blocks are summarized using a *Merkle tree* [4].

A Merkle tree, also called a hash tree, summarizes all the transactions in a block by producing a digital fingerprint of the entire set of transactions [18]. The digital fingerprint enables nodes to efficiently verify the integrity of the data and whether or not a transaction is included in a block.

As mentioned in section 2.4, the byte header consists of six different fields, one of which is the *Merkle root*.

The Merkle root represents the final hash of a Merkle tree. A Merkle tree is created bottom up from the hashes of the transaction IDs from the set of transactions included in the block. The hashes are then repeatedly hashed together until there is only one hash left i.e. the Merkle root. As Merkle trees are binary, if the number of transactions is not a power of 2, the last hash will be duplicated in order to create an even number of leaf nodes.

Satoshi discusses the use of Merkle trees as a way to save disk space [4]. By using a Merkle tree, it is only necessary to store the most recent transactions and the hashes of the tree, the spent transactions can be pruned to save disk space, as seen in figure 2.8 [4].

2.6.2 Simplified Payment Verification

Merkle trees are used extensively by SPV (simplified payment verification) nodes. SPV nodes are nodes that only downloads the headers of the blocks during the initial syncing process, as opposed to downloading the full blockchain [2]. The SPV node may then request transactions from full nodes as needed [2]. The Merkle root located in the block headers, along with the Merkle path can prove to the SPV node that a transaction exists in a block in the blockchain. When the SPV node figures out the Merkle root associated with the transaction, it will request the respective Merkle path

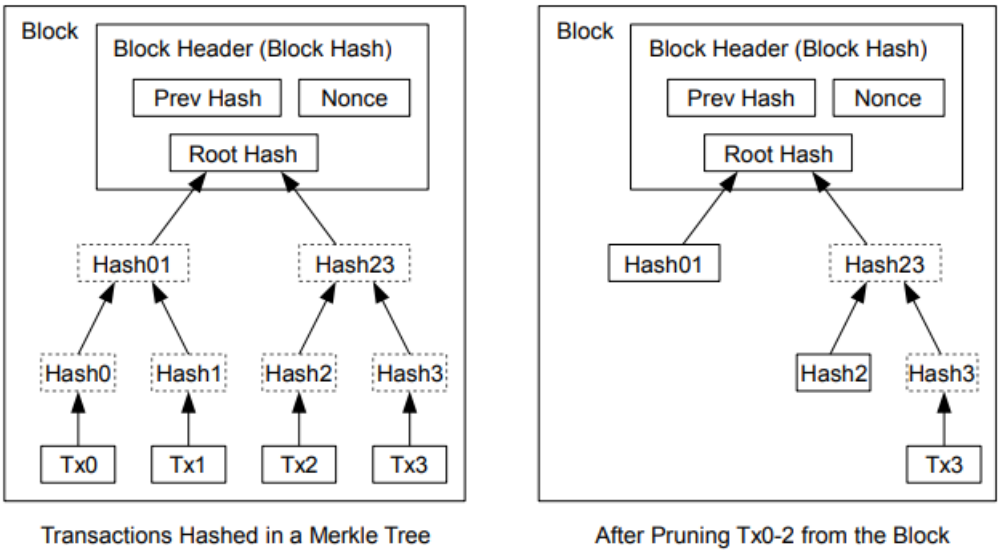

Figure 2.8: Merkle Tree example [4]

from a full node. A full node is a node that fully downloads and validates the entire blockchain, from the genesis block to the latest block. Once the SPV node receives the Merkle path, they can confirm if the transaction is included in the block. The SPV node may then look at the block depth for transaction validity and confirmation. The more blocks built on top of the block for which the transaction was located in, the more secure and valid the transaction may be. The block depth represents the number of blocks built on top of the block for which the transaction is located. The higher the block depth, the more secure and valid the transactions located in the block are. It is suggested to wait until a transaction is six blocks deep for it to be considered as 'confirmed' [2]. This is to prevent the double-spending attack, which is explained in section 2.8.

2.7 The Bitcoin Network

The Bitcoin network is structured as a decentralized P2P network. In a P2P network, nodes participating in the network are seen as peers. Peers are all treated as equal, with shared responsibility in providing network services. The Bitcoin network consists of over 10,000 nodes [6]. Each node in the network implements a version of the Bitcoin protocol through the use of a Bitcoin client. Although there are several Bitcoin clients available to use, the Bitcoin client used by the majority of the nodes in the network is Bitcoin Core, also known as the reference client or the Satoshi client [19].

Although all the peers in the network are equal, they may have different roles based on the different functions they support. For example, as discussed in section 2.6.2, SPV nodes do not keep a copy of the full blockchain and do not participate in mining. They are lightweight nodes that integrate the wallet and routing function, designed for peers with limited resources. In contrast, full nodes may be either a full blockchain node when it includes routing and full blockchain functions, or a solo miner when it includes routing, the full blockchain and mining functions [6]. However, in order to participate in the Bitcoin network, nodes must implement a mandatory routing function. The routing function includes network discovery of new peers, establishing inbound and outbound connections, validating transactions and blocks and propagating information through the network [6].

2.7.1 Network Discovery

Nodes on the Bitcoin network are identified based on their public IP address. A node in the network may have a maximum of 8 outbound connections and 117 inbound connections, however, a node with a private IP address only initiates eight outgoing connections [20]. Connections are created and maintained over TCP.

When a new node joins the Bitcoin network, it must discover nodes already participating in the network. The geographic location of the connecting node and other nodes on the network is irrelevant as the Bitcoin network is not geographically defined. There are several mechanisms for peer discovery in the Bitcoin network [6]:

- **Local Address Database** - The primary method for which nodes in the network to locate clients is to connect to the nodes in their address database. This is a list of nodes stored locally from a previous connection to the network. However, this method does not work for nodes joining the network for the first time or joining after a very long period of disconnection.
- **Asking DNS Seeds** - If the node has no nodes in their address database, the primary fallback option is to query DNS seeds. DNS seeds are DNS servers that

provide a list of stable Bitcoin listening nodes.

- **Seed Node** - The command-line argument -seednode can be used to connect to one node, known as the seed node[4]. The connection to the seed node is used to introduce the node to other peers in the network. After the seed node is used to receive information about other peers in the network, the node may disconnect from the seed node to prevent too many connections to the seed node.
- **User-specified Commands** - The node may run user-specified commands on the command line (-addnode and -connect)[5] to connect to specific nodes where the IP address is known.
- **GETADDR messages** - The node may request for other peers by sending the GETADDR message to their neighbours. The neighbours will respond with a list of possible IP addresses of Bitcoin nodes in the network.

Using one of the methods mentioned above, a node connecting to the Bitcoin network will now have at least one known IP address of a node in the Bitcoin network to connect to. To connect to one of the peers in the network, the node must establish a TCP connection to the node, usually on port 8333 which is generally known as the port used by Bitcoin [5].

Once a connection over TCP has been established, the nodes will initiate a 'handshake' to determine compatibility between the nodes. The handshake is initiated by the transmission of a version message, which contains the following basic information [5]:

Field	Description
nVersion	The current Bitcoin P2P protocol version the node implements
nLocalServices	A list of local services supported by the node
nTime	The current time
addrYou	The IP address of the node that you are attempting to connect to
addrMe	The IP address of the current node
subVer	Type of software running on this node (e.g., /Satoshi:0.17.1/)
BestHeight	The block height of the node's blockchain

The version message is always the first message sent by peers attempting to establish a connection to another peer. The version message is sent by first to determine whether or not the peers or compatible. The node receiving the version message will examine the received nVersion field of the version message and determine whether or not they are compatible. If the peer is compatible, the version message is acknowledged and the

[4]This argument is specific to Bitcoin Core.
[5]Specific to Bitcoin Core.

connection is established by the sending of a *verack*, as shown in figure 2.9.

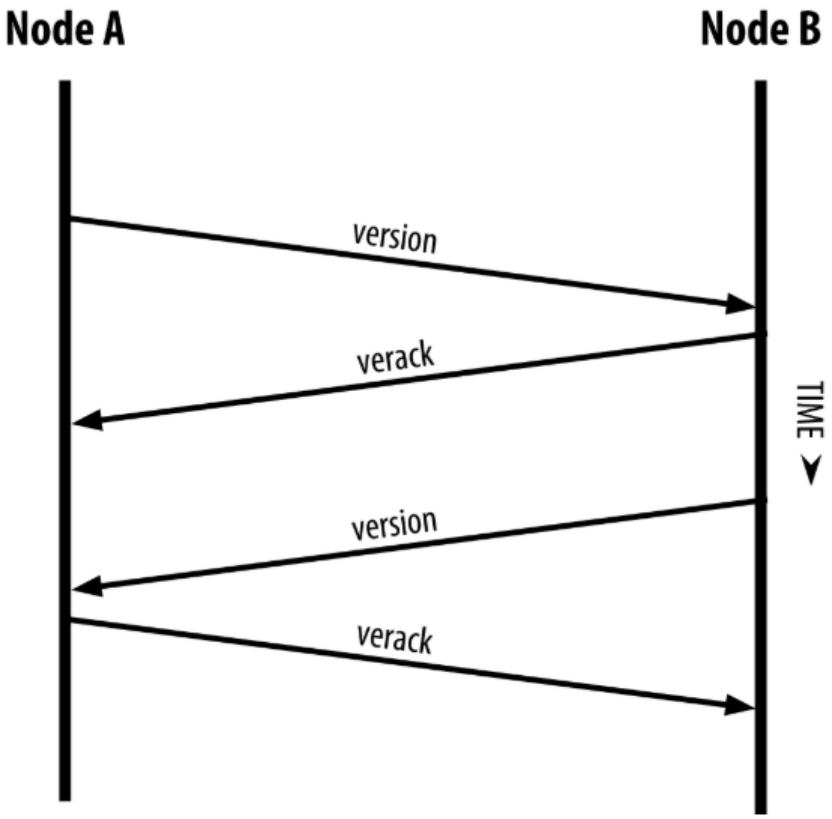

Figure 2.9: Initial Handshake [5]

Once a connection or one or more peers in the network have been established, the node may request for more IP addresses in the Bitcoin network by sending a GETADDR message to their connected peers. As mentioned earlier, each node maintains a local address database containing discovered peers in the network. For each entry in the database, the database includes the peer's IP address, port number, the last time the peer was seen on the network and the timestamp of the last connection [6]. All entries are classified and added into buckets. There are two buckets in the database, buckets for *tried addresses* and buckets for *new addresses*. The peer addresses that have had at least one outgoing/incoming successful connection to it are placed in the tried addresses whereas the peer addresses for which there were no connections established are placed in the new addresses. There are 256 buckets for the tried addresses and 1024 buckets for the new addresses, with each bucket able to contain a maximum of 64 entries each. Therefore, the max number of entries in the database is limited to 81920 addresses [6]. The local database is called the *addrMan* [21]. When a Node X wants to request a list of node addresses from a Node Y, it will issue a GETADDR message to Node Y. In response to the GETADDR message, Node Y will respond with up to 1,000 entries from their addrMan database, chosen uniformly at random [21].

The vast majority of entries in addrMan do not always necessarily correspond to active addresses in the Bitcoin network as the entries in the addrMan database are mainly composed of addresses that the node has learned from other ADDR messages. Figure 2.10 shows the protocol explained above for further address discovery.

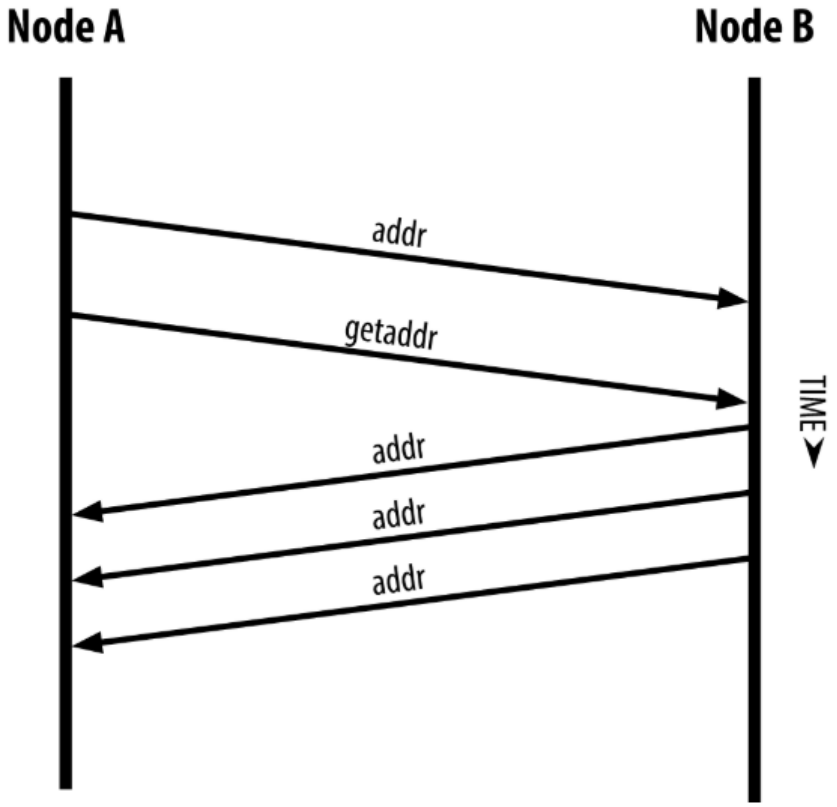

Figure 2.10: Address Propagation and Discovery [5]

2.7.2 Neighbour Selection

As mentioned previously, a Bitcoin node may have up to a maximum of eight outbound connections and a maximum of 117 inbound connections. Nodes may refuse any inbound connections if desired, however, this is bad for supporting the Bitcoin network. When a node's number of outbound connections is less than the maximum of eight, a selection algorithm is applied to decide the new outbound connection [6]. Although a node can reject any inbound connections, it should never drop its outbound connections. A Bitcoin node never deliberately drops a connection unless the connected peer is blacklisted (e.g., if the peer sends ADDR messages that are too large) [20]. The selection algorithm is shown in figure 2.11, with the main idea of the algorithm being the use of unequal selection probabilities favouring fresh peers [6].

Algorithm 3 Outbound neighbour selection.

nOutbound: Number of current outbound connections
nTriedAddr: Total number of current tried addresses
nNewAddr: Total number of current new addresses
nReject : Number of rejected addresses
nNow : Current time
1. A peer is selected from tried buckets with the probability:

$$\frac{\sqrt{\frac{nTriedAddr}{nNewAddr}} \times (9 - nOutbound)}{(nOutbound + 1) + \sqrt{\frac{nTriedAddr}{nNewAddr}} \times (9 - nOutbound)}$$

2. $nReject \leftarrow 0$
3. From the selected type of buckets:

- select randomly (with more weight for fresher timestamps) an entry i
- Accept i with the probability

$$\min\left(1, \frac{2^{nReject}}{1 + (Timestamp[i] - nNow)}\right)$$

- Otherwise, *nReject*++ and go to step 3

4. Connect to the *Address*[i]. If fails, go to 1.

Figure 2.11: Algorithim for Selecting a New Outbound Connection [6]

2.7.3 Information Propagation

When new transactions or blocks are created in the Bitcoin network, they must be broadcasted to the entire network to inform the peers in the network of the new transactions/block. As the Bitcoin network is a decentralized P2P network, there is no central authority to distribute the transactions/blocks to every peer in the network. As nodes in the network are only aware of their directly connected neighbours, Bitcoin implements a gossip-based flooding protocol to propagate transactions and blocks across the network.

When propagating information across the Bitcoin network, a node maintains a message queue for all of their connected neighbours. This message queue may contain different types of messages that a node may want to send to their neighbours, such as transaction hashes or block hashes etc. Along with the message queue, there is a timer associated with each neighbour. All the messages within the message queue will be sent to the associated neighbour when the timer elapses. The time out is calculated using a Poisson distribution [11].

In order for a node not to send the same transactions or blocks that their neighbouring peers may already have, transactions and blocks are not forwarded directly to their neighbours. Instead, an **INVentory** message or **INV** message is sent to their neighbours. The INV message transmits one or more inventories of objects known to the transmitting peer and are now available to be requested from the transmitting peer if the receiving node is missing one or more of the inventories of objects in the INV message[7]. If the receiving node requires any of the transactions or blocks within the INV message, they will respond to the sender node with a **GETDATA** message, which contains the hashes of the information the node requires. Once the GETDATA message is received, the sender node will send the requested block or transaction via individual *block* or *tx* messages [7]. However, if a node receives an INV message that contains transactions and blocks that the node already possesses, the node will simply ignore the INV message and not respond with a message to the sender node.

Although sending INV messages to neighbouring peers will prevent the peers from receiving duplicate transactions, peers may still receive duplicate INV messages for the same transaction. This occurs as a node's neighbours does not know which transactions the node currently has or is missing. Therefore, if a node's neighbour recently received new transactions, they will add it to the INV message that will be sent to the node as they assume the node might not have the transactions they just received. As every node's neighbours may think the same, a node may receive an INV message for the same transaction from all of their connected neighbours (125 worst case) whereas 1 INV message would have sufficed to send the transaction to the node.

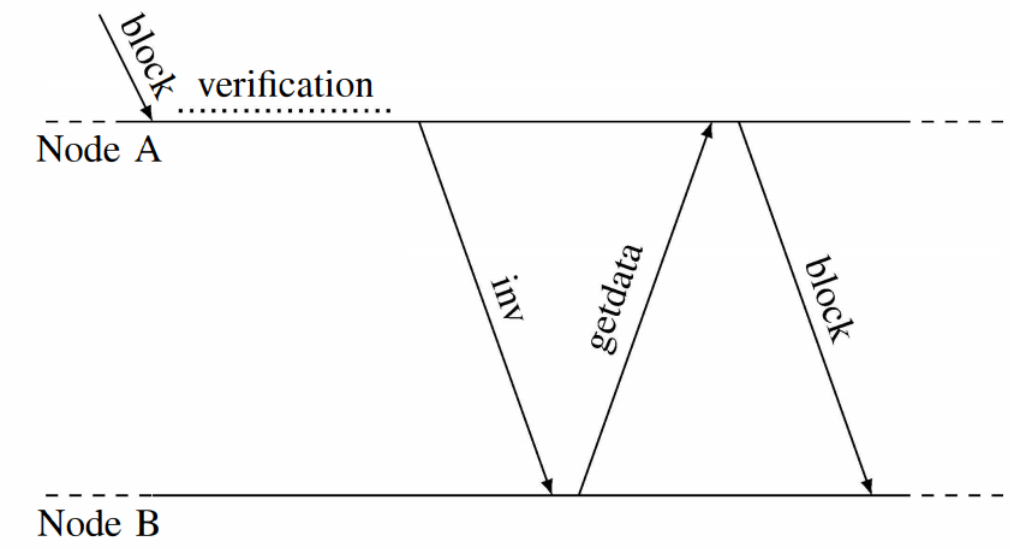

Figure 2.12: Message Exchange For The Propgation of Information across the Bitcoin Network [7]

Compact Blocks

Historically, the Bitcoin P2P protocol for the dissemination of blocks had not been very bandwidth efficient. This is due to the fact that every transaction was included in the blocks when propagated through the network, even though the nodes in the network already had the majority, if not all, the transactions in their local mempool.

Blocks would be advertised in INV messages, just as transactions are. If a node does not have the broadcasted block, they will again respond with a GETDATA message to the transmitting node. The node will then send the entire block containing the full set of transactions to the node, regardless of the fact that the receiving node may already have all the transactions in their local mempool. Bitcoin Core developers identified this waste of bandwidth during block propagation and introduced *compact block relay*, BIP 152, in 2016 [8].

In compact block relay, peers send compact block 'sketches' to receiving peers. The sketches include the following information:

1. The 80-byte header of the new block
2. Shortened transaction identifiers (txids)
3. Some full transactions which the transmitting peer predicts the receiving peer may not have in their mempool

The receiving peer will attempt to reconstruct the block from the 'sketch', using the received information and transactions in its mempool. If the peer is still missing transactions to completely reconstruct the block, it will request the missing transactions from the transmitting peer by sending a *getblocktxn* message [8].

The main advantage of compact block relay is that in the best case, transactions only have to be sent once - when they are originally broadcasted. This reduction in duplicated transactions reduces the overall bandwidth significantly.

Compact block relay has two modes of operation - high bandwidth relaying and low bandwidth relaying. In high bandwidth mode, the receiving peer asks its neighbouring nodes to send new blocks without asking for permission i.e. without the INV message. This increases the bandwidth as two or more nodes may be sending the same block to the receiving node at the same time, however, this reduces the amount of time it takes for a block to arrive at the receiving node [8]. The high bandwidth relaying option may be of use to mining nodes, who want to receive the latest block as soon as possible so they can be working on mining the next block. In low bandwidth mode, the receiving peer will wait for an INV message from the transmitting peer before replying with a GETDATA message for the compact block if needed. Figure 2.13 displays the three

methods of relaying blocks - the two modes of operation of compact block relay and the legacy version of block relay. The grey box represents the validation time.

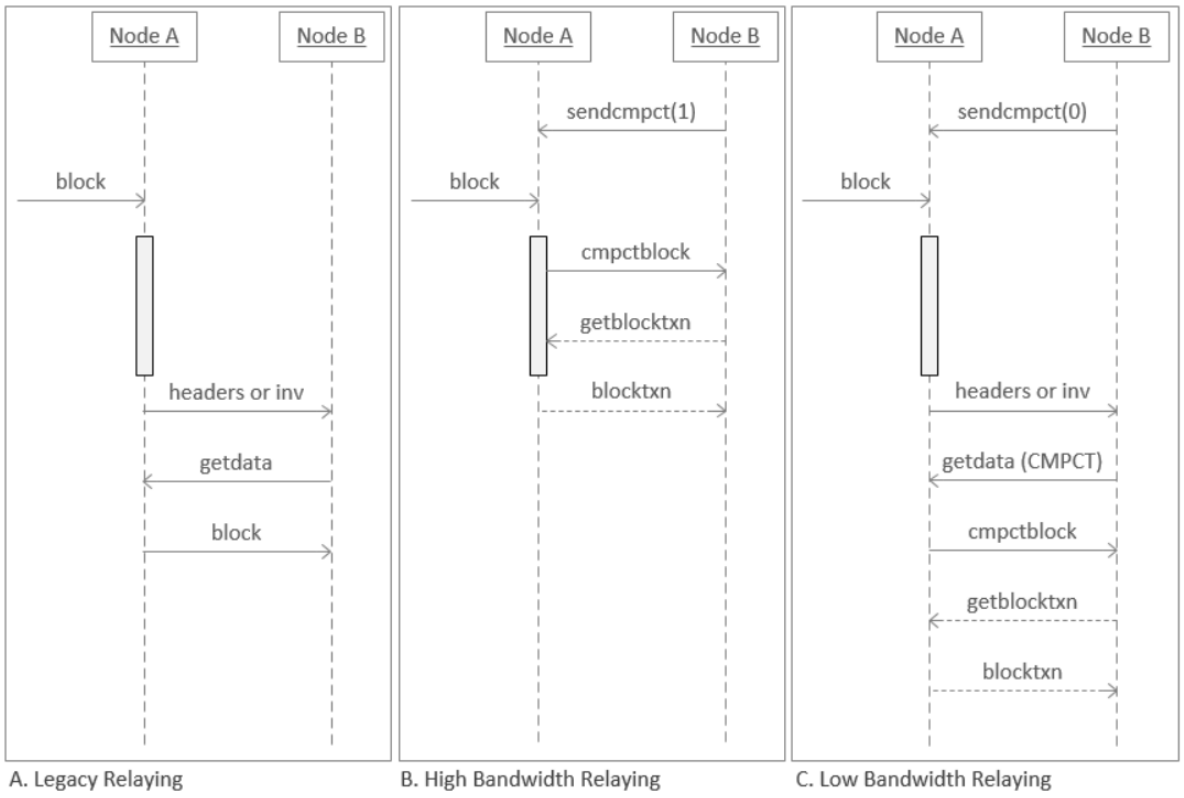

Figure 2.13: Three Methods of Block Relay [8]

2.8 Bitcoin Security

As Bitcoin is the world's most popular cryptocurrency with a market cap of approximately $70 billion at the time of writing, it is vitally important that the system is resilient to potential attacks on the system. As the main attribute of a decentralized digital currency is the lack of a need for a central authority, security and privacy become exponentially more important than before. In this section, we will discuss known security vulnerabilities in relation to Bitcoin.

51% attack

The 51 percent attack is an attack on the Bitcoin network where an entity controls 51% or more of the computing power, meaning it can control the blockchain [22]. With the PoW consensus mechanism that is implemented within Bitcoin, the probability of mining a block is based on the computational power of a user relative to the rest of the network's computational power. The better their CPU/GPU, the more hashes they are able to check per cycle, leading to a higher chance of finding the nonce in order to mine the block. As the odds of a person mining a block with just a single

CPU/GPU is extremely low, people are encouraged to join mining pools, as discussed in section 2.5.5. In mining pools, miners harmonize their computational power in order for a higher chance to successfully mine a block. When a mining pool becomes too large and too powerful that it can manage to control 51% of the computational power, it then becomes a problem to the Bitcoin network. Controlling over 51% of the computational power would allow the entity to:

1. Prevent new transactions from gaining confirmations.
2. Allow the entity to halt payments between some or all users.
3. Allow for the possibility of double-spending coins.

However, due to the effective computational power in the Bitcoin network, a 49.1% share of the computational power in the network is enough for an attacker to carry out the above attacks [7].

Hackers and Cyber-Attacks

As Bitcoin is a decentralized digital currency, cyber attacks directed at Bitcoin exchanges and wallets is a real concern. The attackers are not attacking the blockchain itself, as described above, as it is quite difficult to attack the blockchain as it requires the majority of computational power within the Bitcoin network. Attackers are more likely to target the major bitcoin/cryptocurrency exchanges, where bitcoins and other cryptocurrencies are exchanged and stored. An example of a successful attack on a Bitcoin exchange is the attack on Mt. Gox. Mt. Gox was a Bitcoin exchange based in Tokyo, Japan. Launched in 2010, by 2014 it was handling over 70% of all Bitcoin transactions worldwide, meaning it was the largest Bitcoin intermediary and the world's leading Bitcoin exchange. The attack in 2014 led to Mt. Gox losing approximately 850,000 bitcoins (around 7% of all bitcoins in existence at the time), valued at the equivalent of $460 million at the time, or approximately $3.3 billion in today's market [23]. As a result of this attack, Bitcoin users are highly recommended not to store their bitcoins on an exchange but rather transfer them to their own personal cryptocurrency wallet.

Double-Spending Attack

Double-spending means spending the same money twice. Double-spending is and will always be an issue with digital currency as opposed to traditional, fiat currency. If you go to the shop and buy a sandwich worth $10, you hand the vendor $10 in cash in exchange for the sandwich - there is no possibility that you would be able to spend that $10 again. However, whenever you are making a transaction with digital currency, you have to broadcast that transaction to your neighbouring nodes who then confirm

the transaction is valid. A double-spend attack occurs when a user attempts to send their coins to two users at the same time by forwarding their transaction to two different nodes within the network. As the Bitcoin network is a distributed P2P network, there is a propagation delay between nodes and due to this delay, there would be a period in time where the two nodes both believe that they have the correct and valid transaction from the node attempting to double-spend their coins. However, transactions are not confirmed in Bitcoin until they are placed in blocks. If an entity has over 51% of the computational power, they can spend their bitcoin on the 'truthful' version of the blockchain while they do not include their transactions in their 'secret' blockchain that they are mining. As the secret blockchain gets longer than the 'truthful' version of the blockchain, the entity will then broadcast their 'secret' and longer blockchain to the network. Truthful miners and nodes will always work on the longest chain [4], meaning that they will work on the malicious entities chain when it is broadcasted. This means that the old chain, where the malicious entities transactions were spent, will be abandoned and disregarded as it is the shorter chain, so that data is now irrelevant. Now, the malicious entity can spend their coins again as the new version of the blockchain does not contain any of their transactions that they spent - the old, now abandoned, version of the blockchain contained their transactions [24]. The ability to double-spend in Bitcoin is highly dependant on the ability to control over 51% of the computing power within the network. An example of the double-spending attack is shown in figure 2.14.

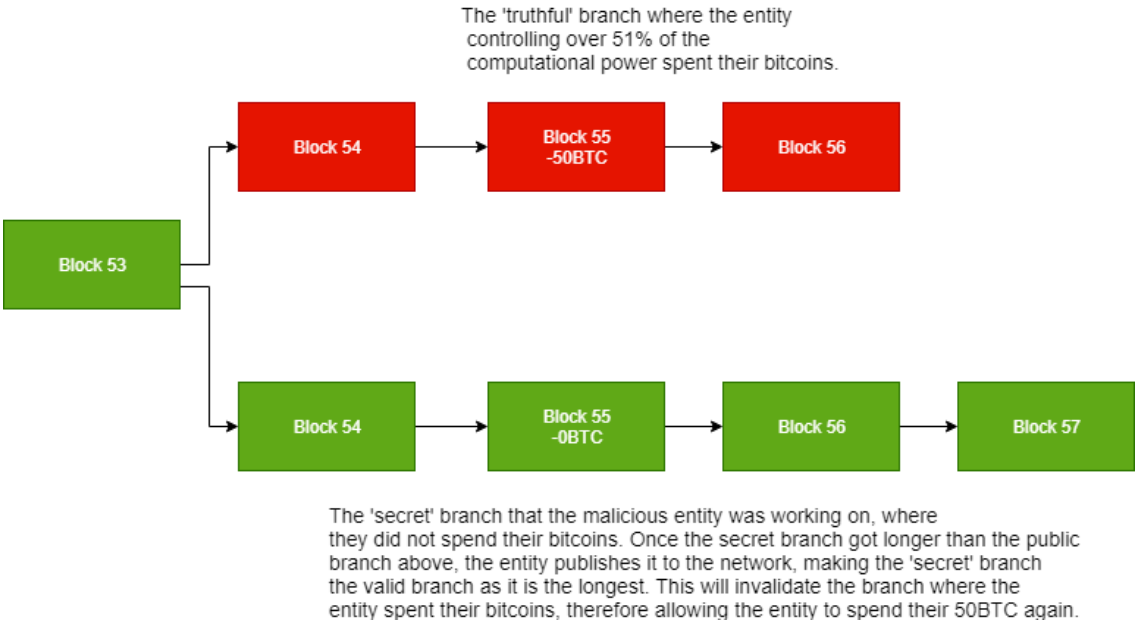

Figure 2.14: An example of the double-spending attack

Selfish Mining

Courtois and Bahack [25] indicated that miners could have a specific mining strategy known as *selfish mining*, also known as *block discarding attack*. In selfish mining, nodes purposely withhold mined blocks from the network, only revealing the mined block(s) in a selective way which benefits the selfish miners. Through selfish mining, the selfish miners purposely create a fork in the blockchain. The selfish miners will continue working on their private chain, while honest miners are working on the public chain. If the selfish miners can get a lead on the honest miners (i.e. have more blocks mined than the public branch), they can maintain the lead for longer, therefore increasing their rewards as well as wasting the honest miners time and resources. In order to avoid any losses, the selfish miners will publish their private branch when the public branch and honest miners are about to catch up to the private branch. By doing this, the miners in the network will have to work on the selfish miners newly publish branch, as miners always work on the longest branch [4]. The honest miners will now lose their rewards for the previously mined blocks, as the blocks are now invalidated. Eyal and Sirer [26] show that through the use of selfish mining, the selfish pool's reward exceeds its share of the networks computational power. Figure 2.15 shows an example of selfish-mining.

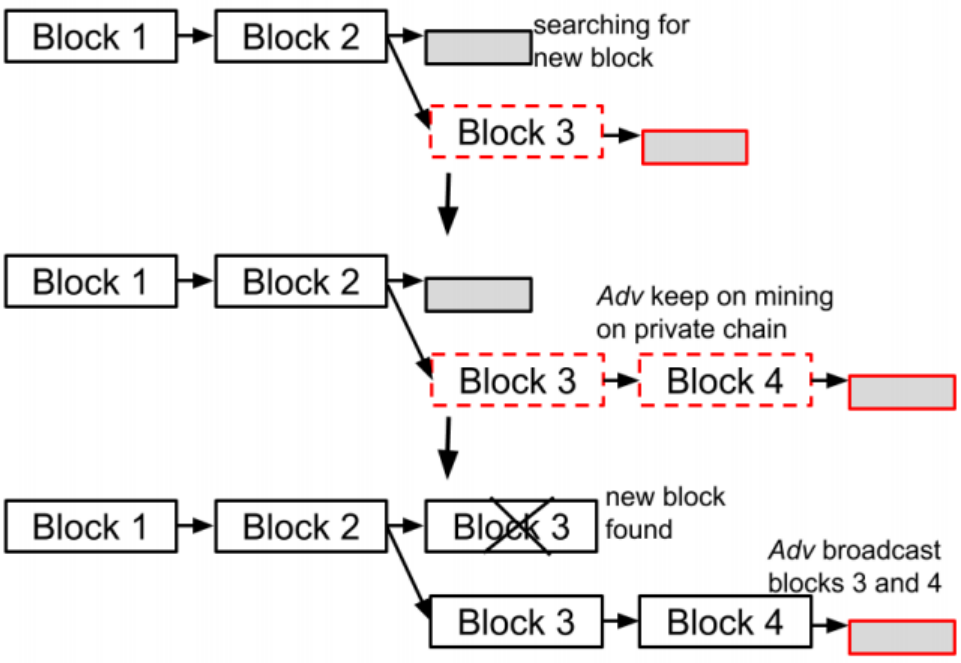

Figure 2.15: Selfish-Mining [9]

Eclipse Attack

An eclipse attack is an attack on a specified user/node(s) in a decentralized P2P network. The attacker attempts to isolate their victim node by controlling all the neighbouring nodes it is connected to, preventing the victim node from receiving a true picture of the network and the network's activities/ledger state. This is able to occur as in decentralized P2P networks, nodes are not able to connect simultaneously to all the nodes within the network. Instead, they are only able to connect to a subset of nodes, who in turn are connected to their own subset of nodes etc. Eclipsing a victim node is beneficial to the malicious actor as they would be able to exploit them. The malicious actor would be able to double-spend their coins against the victim node. For example, if User A is the malicious actor, User B is the victim and User C is a normal node within the network, User A can send the same transaction to User C and User B. However, since User B is eclipsed and isolated, User B would not know that the same transaction has been sent to User C and will accept the transaction as valid. However, when User B later connects to the true blockchain, they will realize that the coins for the transaction has already been spent and they will receive nothing as the transaction is invalid. Figure 2.16 represents a simple example of an eclipse attack.

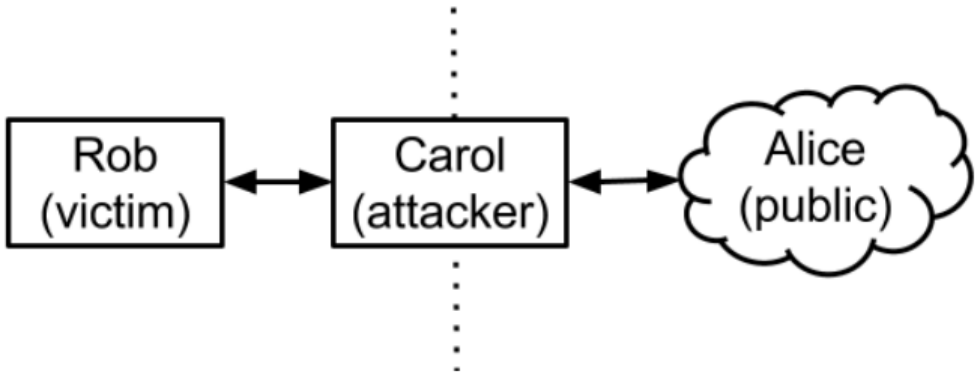

Figure 2.16: A simple example of an eclipse attack [9]

Sybil Attacks

A Sybil attack is an attack on a P2P network by creating multiple fake identities on the network. To nodes within the network, these multiple identities seem to be unique but in reality, they are all controlled by a single entity. This may help the entity influence the network through additional power gained by having multiple nodes within the network [9]. However, Bitcoin mitigates a Sybil attack through mining and PoW. In order to create a new mining node, there needs to be a computer with processing power in order to mine. Therefore, creating hundreds or thousands of pseudonymous nodes comes at a significant cost that is not worth it for the malicious entity. Sybil attacks are different to eclipse attacks as eclipse attacks are targeting a single node whereas Sybil attacks is a network targeted attack.

3 Related Work

In this section, we will discuss multiple studies that have focused on improving the Bitcoin system.

Decker and Wattenhofer [7] explores the limits of the Bitcoin protocol and whether or not changes in the node's behavior can change to improve the blockchain fork rate. They attempt to improve information propagation within the Bitcoin network by:

1. Minimize verification
2. Pipelining block propagation
3. Connectivity increase

They limit the changes in a unilateral manner in order to assess the effectiveness without major changes in the Bitcoin protocol, which would have to be vetted and agreed upon by the Bitcoin community.

The major contribution to propagation delay is the time a node spends on verifying a block before further propagating the block to the network. Block verification can be divided into two phases:

1. Initial difficulty check
2. A transaction validation

The initial difficulty check is to ensure the correct nonce was found to solve the PoW associated with the block. It also ensures that the block is not a duplicate block and it references a recent block as its predecessor. The majority of the verification time is spent verifying the individual transactions included in the block. The suggested change is to propagate the block as soon as the initial difficulty check is verified. Figure 3.1 shows the changes described above.

Another improvement suggested by Decker and Wattenhofer was pipelining block propagation. This is achieved by immediately forwarding incoming INV messages to the node's neighbours. This is to amortize the round-trip times between the node and its neighbours by pre-emptively announcing the availability of a block before it actually is

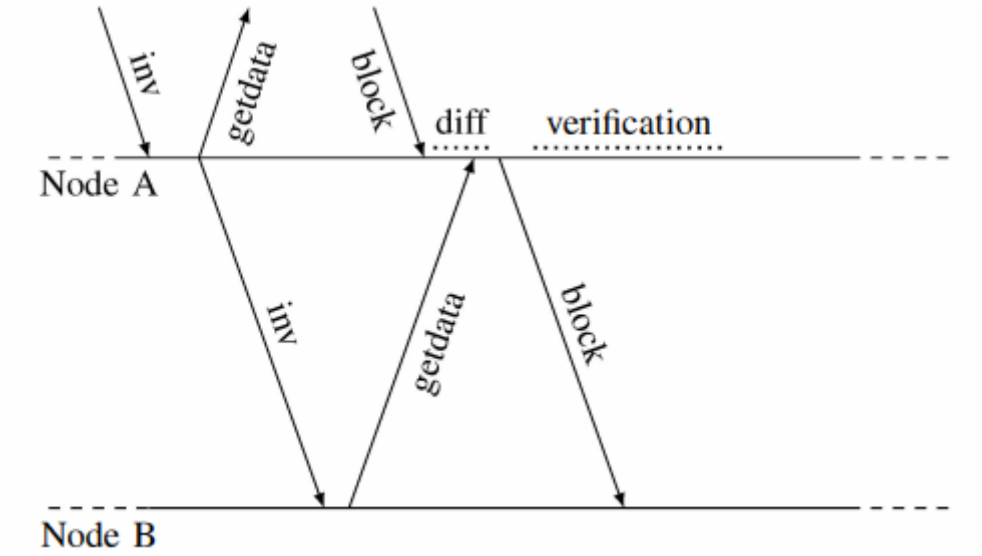

Figure 3.1: Message Exchange After the Changes Described are Applied [7]

available. However, as stated in the study, these two changes are unlikely to result in a large improvement if only implemented by a single node in the network.

The third change implemented was to decrease the distance between the transactions/blocks between nodes. Decker and Wattenhofer attempted to connect to every node, therefore creating a central communication hub between the nodes in the network which resulted in the distance between two nodes to be 2 [7].

The results of the studies showed that the blockchain fork rate dropped from 1.69% to 0.78%. The pipelining and verification, as mentioned previously, had a very small effect on the system. The last change had high bandwidth spikes up to around 100 MB/s and resulted in a total upload of 2.31TB of raw data during the testing period. However, the issue with the approach taken by Decker and Wattenhofer is the third change, which essentially centralizes the network. As stated previously, one of the main and most important characteristics of Bitcoin is the decentralization nature of it, making it highly resilient to the potential attacks described in section 2.8. By creating a central hub of communication, the author risks a targeted attack on the central hub which could decimate the network.

Fadhil et al. present a new protocol, Bitcoin Clustering Based Super Node (BCBSN) as a mechanism to speed up information propagation in the Bitcoin network [27]. In this protocol, the Bitcoin network is divided into geographically diverse clusters. Within each cluster, there is a cluster head or super node responsible for maintaining the cluster. Each peer is connected to a cluster head, and each cluster head is connected to other cluster heads. The claim is that this would reduce propagation delay as it reduces the number of non-compulsory hops that blocks or transactions required to

reach all the peers in the network. Nodes at each cluster are geographically localized, with the hope of reduction in the link latencies between nodes at each cluster. The BCBSN protocol resulted in a reduction of the transaction propagation time variances, compared to that of the normal Bitcoin network. Possible limitations of the BCBSN protocol may include a successful attack on a cluster head. By successfully attacking a cluster head, the nodes in the associated cluster are unable to connect to the rest of the network as the cluster head was their means of contact to the rest of the network. If the cluster head was infiltrated by a malicious node, they have essentially partitioned the nodes within the cluster from the rest of the network and may carry out an eclipse attack, as described in section 2.8. As nodes in the clusters are geographically localized, this may make the network highly prone to partitioning.

Following on from BCBSN, Fadhil et al. proposed a proximity-aware extension to the current Bitcoin protocol, named Bitcoin Clustering Based Ping Time Protocol (BCBPT) [28]. Based on their previous work BCBSN, which placed nodes in clusters based on their geographic location, BCBPT will place nodes in clusters based on their ping latency. Nodes that are geographically close could be quite far away from each other on the physical internet [28]. The results of BCBPT show that the protocol maintains an improvement in variances of delay over their previous work, BCBSN. This may be due to the fact that in BCBSN, clusters are based on geographic location, meaning they could be close geographically but far away on the physical internet. By creating clusters based on ping latencies, Fadhil et al. concluded that proximity awareness in the physical internet improves delivery latency with a higher probability than clusters based on geographic locations. The protocol is split into two phases:

1. Distance calculation
2. Cluster creation and maintenance

Distance Calculation

In the distance calculation phase, each node is responsible for gathering proximity knowledge regarding discovered nodes. This is done by calculating the distance in the physical internet between the node and the discovered nodes. Proximity is defined as how far a node is from another node in the physical internet.

Cluster Creation and Maintenance

When joining the network for the first time, a node N will learn about other available Bitcoin nodes in the network from a list of DNS services, as mentioned in section 2.7.1. The node N will calculate the proximity distance to each of the discovered nodes. The

node N will then send a JOIN request to the closest node K of the discovered nodes. Once node N establishes a connection with node K, it will receive a list of IPs of nodes that are in the same cluster as node K. Node N will then connect to all the nodes in the cluster. If node N discovers a node that is physically closer than the current cluster, node N will leave to join the nearer cluster.

Although the transaction propagation time and variances are lowered in the proposed protocol, the same issues from BCBSN can be applied to the proposed protocol. As mentioned by Fadhil et al. in the paper, they identify that eclipse and network partition attacks have great potential due to the clustering based on countries. An attacker might concentrate a number of bad peers within a cluster in order to create a malicious cluster on the network. [28].

Marçal [11] proposes a new protocol for the dissemination of transactions in the Bitcoin network. The protocol proposes a bias to disseminate transactions to neighbours that are more likely to reach miners quickly, as miners are the nodes that need knowledge of the transactions in the network as they are responsible for placing the transactions in blocks, and subsequently placing the block on the blockchain.

The protocol encompasses three changes to the Bitcoin dissemination protocol:

1. Nodes maintain for each of their neighbours, a list of transactions sent by their neighbour and how long it took for these transactions to be included in a block.
2. Nodes maintain for each of their neighbours, the time it took to disseminate a new block to the node.
3. Use the metrics collected above to rank their neighbours and prioritize the dissemination of transactions based on the rankings.

The proposed protocol was able to reduce the bandwidth usage by 10.2% and reduce the number of messages exchanged in the network by 41.5%. Some issues with the aforementioned protocol are that the commit time of transactions may increase as transactions are reaching miners, but may not necessarily reach the miner who is going to mine the next block [11].

It is important to note that during the research for this dissertation, some P2P algorithms unrelated to Bitcoin were studied to see if there were any new P2P algorithms better fitted to suit the Bitcoin network.

Barjini and Othman [29] suggests a novel algorithm, SmoothFlood, which divides the flooding scheme into two phases. The first stage of the algorithm will follow the standard flooding protocol before switching to a super-peer (nosy node) phase. The flooding scheme discussed in this study, however, is a flooding scheme for file-sharing

rather than the flooding for information propagation used in bitcoin. However, the concepts discussed in the paper could still prove to be useful.

In the first phase of SmoothFlood, pure flooding is implemented for a high coverage growth rate of peers, accompanied by low redundant messages. Conversely, the second phase will be implemented when the coverage rate is low and the number of redundant messages is high. The second phase does not follow the pure flooding algorithm. Instead, noisy nodes are selected as super peers. Super peers create an index table (cache), which indices all the files their neighbours would like to share. When a query reaches a noisy node, it will look up in its cache if the file is there and return the address of the client for getting the file. Otherwise, the super-peer will query other super-peers on his level. Results from the study show a decrease of 65% of redundant messages and save up to 70% in searching traffic.

Applying a similar concept to the Bitcoin system could prove to be difficult. Identifying the noisy node in the Bitcoin network may prove to be impossible as currently, there are no direct ways to get information about a node's neighbour and ways to do so are considered a security threat. It would prove to be impossible to verify a node's claim that they are the nosiest node with X amount of neighbours if we are unable to receive information about a node's neighbour.

4 Probabilistic Flooding

This chapter will describe the issue that we have identified with the current Bitcoin propagation method and the proposed protocol changes that we have implemented in order to improve the identified issue.

4.1 The Problem

As described in section 2.7.3, when new blocks are mined or new transactions are created, they must be propagated across the entire network to ensure that nodes in the network are aware of the new transactions/blocks. As Bitcoin is a decentralized peer-to-peer network, nodes in the network do not have a global view of the network topology. Nodes in the network are only aware of their immediate, connected neighbours. In order to propagate the transactions or blocks across the network, nodes must send the information to their neighbouring nodes, who in turn will send it to their neighbouring nodes etc. until eventually every node in the network is aware of the new transactions/block. The process of disseminating information across the network in this manner is known as gossip-based flooding [30].

The broadcasting and sending of a transaction requires the exchange of three messages:

- INV message
- GETDATA message
- tx/block message

This message exchange is required to prevent sending the entirety of a transaction or block to a node that may already possess it. Through the use of INV messages, the transmitting node only sends hashes of transactions or blocks to the nodes, which is only a fraction of the size of the transaction or block as a whole.

As mentioned in section 2.7.3, a node contains a message queue and a timer for all of their connected neighbours. When the timer elapses for the neighbour, the node will

send an INV message to the associated neighbour. The INV message transmits one or more inventories of objects (transaction hashes and/or block hashes) to the neighbour [2].

There are two options when a node receives an INV message:

1. If the node receiving the INV message already has knowledge of all the transactions/blocks contained in the INV message, the node will ignore the INV message. No GETDATA message or any additional messages are needed to be exchanged between the nodes.
2. If the node receiving the INV message requires one or more of the transactions/blocks contained in the INV message, the node will respond to the node that sent the INV message with a GETDATA message. The GETDATA message will request the missing transactions/blocks that the node requires [2]. On receiving a GETDATA message, the node will respond with the requested transactions or blocks with a *tx* message or a *block* message respectively [2]. Figure 4.1 illustrates the scenario described.

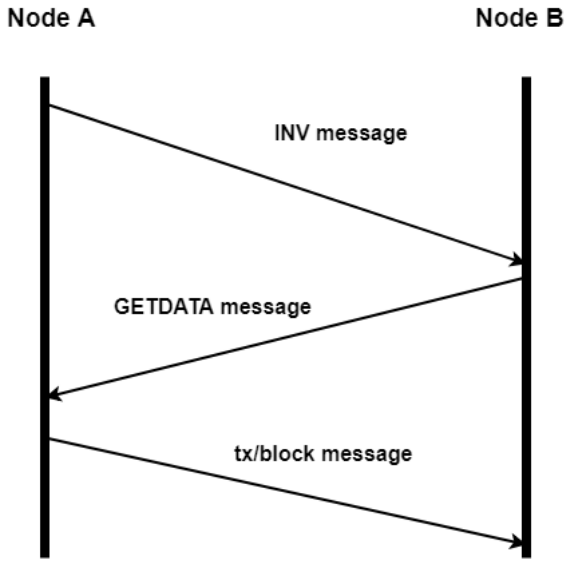

Figure 4.1: Messages Exchanged between Two Nodes for Information Propagation

Even though INV messages in Bitcoin are used to prevent nodes from sending the entirety of a transaction or block to a node that already possesses it, it does not prevent duplicated INV messages being received by nodes. If node A had just received a new transaction, the node will add that transaction to the message queues of its connected neighbours as node A does not know if its connected neighbours have the transaction or not. However, it does not add the transaction to the message queue of the neighbour that sent them the transaction as node A knows that the neighbour who sent them the transaction already possesses the transaction. However, a node may at

the worst case, receive 125 INV messages for the same transaction, therefore receive 124 duplicated INV messages whereas one INV message would have sufficed to have received the transaction.

The example in figure 4.2 shows the propagation of transaction X. The green nodes, nodes A - H, have already received transaction X through the standard INV message protocol shown in figure 4.1. However, node I has not had transaction X propagated to itself yet. As one can see in the diagram, four out of the seven neighbours that node I is connected to have transaction X and are going to send an INV message to node I, which will contain the transaction hash of transaction X. Although node I only needs one INV message to be able to successfully receive the transaction, it receives a total of four INV messages. Three of the INV messages that node I received are redundant and node I will not respond to three of the nodes. Node I will, however, respond to one of the INV messages (most likely the first INV message received) and respond with a GETDATA message, which will inform the transmitting peer that node I requires transaction X. The transmitting peer will then send transaction X to node I with a *tx* message.

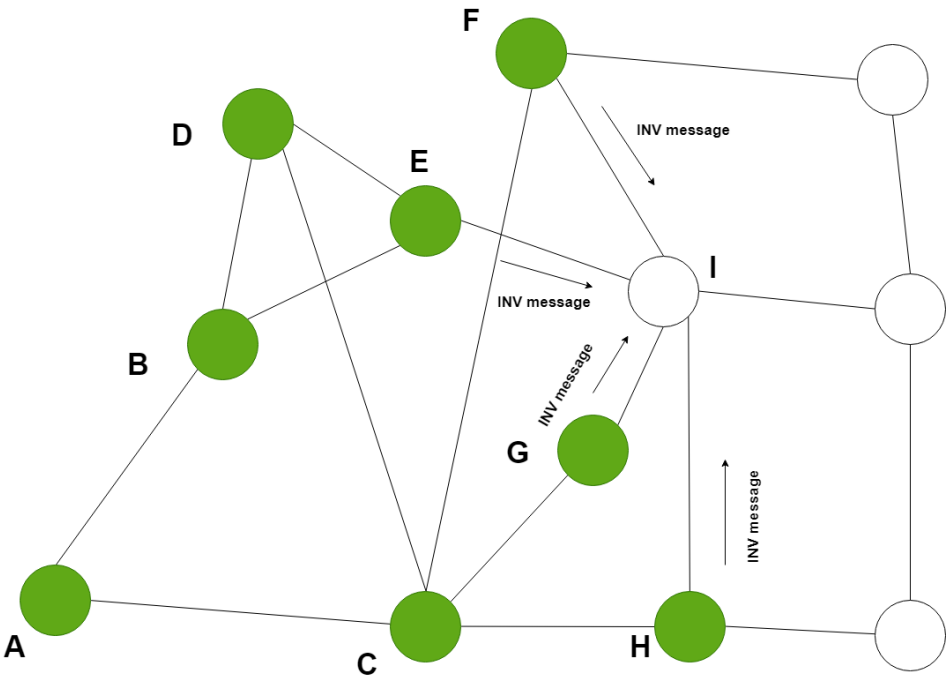

Figure 4.2: Node Receiving Duplicate INV Messages for the Same Transaction

As one can see from the example above, there are a large number of duplicate INV messages that are produced in the Bitcoin network using the current flooding mechanism. The more neighbours a node is connected to, the more well-connected the node is and as a result, the more duplicated INV messages the node will receive. Each node currently receives 6.6 duplicated INV messages for each transaction in the network [11].

4.2 Probabilistic Flooding

As described above, when propagating a transaction through the network, an INV message will be sent to the node's neighbours 100% of the time. This flooding mechanism implemented by Bitcoin produces many duplicated INV messages being received by nodes in the network.

The solution and protocol change that we propose changes the current flooding mechanism approach that was described above to a probabilistic flooding approach. The probabilistic approach will aim to maintain a probability for each of the node's neighbours. This probability is the probability that a node will send an INV message to the associated neighbour. The probability is calculated based on the number of INV messages sent to the neighbour and the number of GETDATA messages received in return from the neighbour.

Formula for Calculating a Neighbours Probability

$$neighbourProbability = \frac{totalGetdataFromNeighbour}{totalInvSentToNeighbour}$$

The idea of sending INV messages based on probability is centered around the fact that nodes in the Bitcoin network have a large variance in the number of connected neighbours. A node may be well-connected, and in the best case, have 125 neighbours whereas another node may have as low as 8 neighbours. The node with 125 neighbours is more likely to have already received the transactions contained in the INV message that it received and therefore will not reply to the INV message with a GETDATA message. The idea of the protocol change to a probabilistic flooding approach is based on the criteria that well-connected nodes will already have the transactions contained in an INV message and will not need to receive an INV message 100% of the time, whereas a node that is less connected may need to receive an INV message the majority of the time.

For example in figure 4.3, node A will send an INV message to node B with a higher probability than sending an INV message to node C. This is due to the fact that node B has a total of three neighbours and is less connected than node C, who has a total of five neighbours. As node C is more well-connected, it is more likely that node C may already have the transactions contained in the INV messages, whereas node B is less likely to have the transactions as it has two fewer neighbours than node C. The probability is based on previous message exchanges between the nodes. In this case,

node A may have previously sent 54 INV messages to node C and may have only received 34 GETDATA messages in return. In this case, the probability that node A will send an INV message to node C, based on the formula mentioned above, will be 63% (34/54). The probability of sending an INV message from node A to node B is also based on the exchange of previous messages between the two nodes. In this case, node A sent 77 INV messages to node B, whilst receiving 64 GETDATA messages in reply. Based on the formula of calculating neighbour probability, the probability node A will send an INV message to node B will be 83% (64/77).

From the example, node B replies to INV messages more times than node C, and therefore will have a higher probability of receiving an INV message in the future from node A. The higher probability can be attributed to the fact that node B only has three neighbours and is not as well-connected as node C, who has five neighbours. As node B has fewer neighbours, this leads to fewer options for which it may receive an INV message for certain transactions in the network, leading to a higher GETDATA response rate when it receives an INV message. Conversely, node C is better connected than node B, having five neighbours. This leads to more avenues for which node C may receive INV messages, therefore leading to a lower response rate to INV messages. As node C has more neighbours, this leads to a higher probability that they have already received the transactions contained in the INV message.

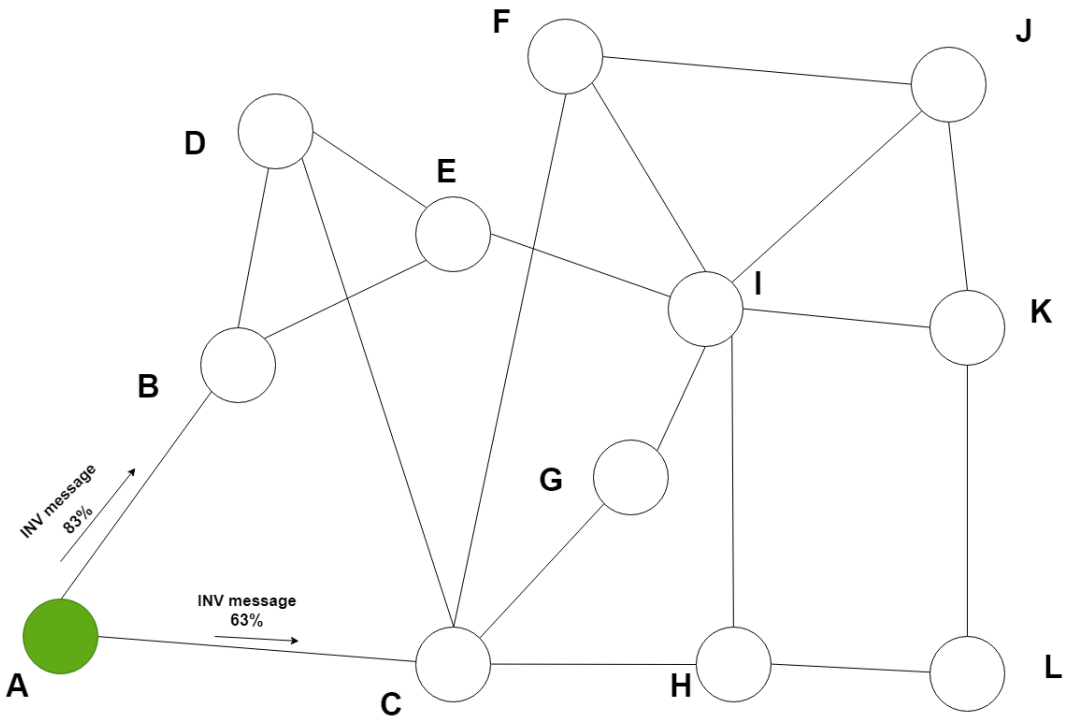

Figure 4.3: Probabilistic Flooding Example

4.3 Implementation

In order to test our protocol changes, multiple simulations were run.

There were a number of Bitcoin simulators to choose from, however, many of them were out of date and did not support the newest changes to Bitcoin such as the introduction of compact blocks. The Bitcoin simulators *shadow-plugin-bitcoin*[1] and *Arthur Gervais' bitcoin-simulator*[2] did not support the latest versions of Bitcoin and were no longer supported.

However, *Joao Marcal's bitcoin-simulator*[3] [11] did support the newest versions of Bitcoin and recorded a number of important metrics that would be vital to compare and contrast the current Bitcoin protocol and the proposed probabilistic flooding approach. The metrics recorded by the Bitcoin simulator are as follows:

- Average number of INV messages sent per node
- Average total number of sent messages per node
- Percentage of duplicated messages received per node
- Total transactions created
- Percentage of transactions created and committed
- Total number of forks created

The simulator is an event-driven simulator, where the behaviour of each node in the network is defined by a deterministic state machine, that consumes events and produces events. The simulator and consequent code changes required to implement the probabilistic flooding mechanism was coded in Python2.7.

The simulator allows for custom configurations, allowing the user to adjust certain properties within the simulation such as:

- Number of nodes in the network
- Number of miners
- Minimum neighbourhood size of each node
- Number of cycles

Each cycle in the simulation represents a second in real-time. The default settings and the settings for which the results in section 5 are formed are based on the following

[1] https://github.com/shadow/shadow-plugin-bitcoin
[2] https://github.com/arthurgervais/Bitcoin-Simulator
[3] https://github.com/JoaoBraveCoding/bitcoin-simulator

configurations:

- Number of nodes in the network - 625
- Number of miners - 5
- Minimum neighbourhood size of each node - 8
- Number of cycles - 208800

As each cycle represents a second in real-time, the experiments were run for a simulation time of 58 hours. The first five hours and the last five hours of the simulation were discarded in order to study the system in a stable state. As the Bitcoin protocol is highly complex, simulating the full network resulted in a resource-intensive simulation that lasted for days. Originally the simulation set the number of nodes to 6,000 nodes. However, the number of nodes was scaled down to 625 nodes and produced similar results to the network with 6,000 nodes. Reducing the number of nodes by almost tenfold will reduce the run-time required to run a simulation, allowing for more simulations, experimentation with different scenarios and run multiple instances of each scenario. The simulator is tuned to generate blocks at the Bitcoin desired rate of 1 block per 10 minutes, as well as creating 2 transactions per second.

Algorithm 1 represents how the probability of sending an INV message to a specific neighbouring node is calculated.

Algorithm 1 Function to calculate the probability of sending INV message to each neighbouring node

```
1: function GET_PROBABILITY(myself, neighbouring_node)
2:     total_inv_sent ← get_total_inv_sent(myself, neighbouring_node)
3:     total_getdata_received ← get_total_getdata_received(myself, neighbouring_node)
4:     probability_to_send ← total_getdata_received/total_inv_sent
5:     return probability_to_send
```

Algorithm 2 is the function that will determine whether or not a node will send an INV message to its neighbouring node. Algorithm 2 is called every cycle for every node, as long as the adjusted probabilistic flooding mechanism is enabled in the simulation.

Algorithm 2 will firstly get the current time of the simulation. For each of the node's neighbours, the algorithm will receive the calculated probability of sending an INV message to that specific neighbour based on algorithm 1. As mentioned in section 2.7.3, associated with each neighbouring node is a timer which is calculated using a Poisson distribution. The node will receive the timer for the neighbouring node and will

determine whether or not the timer elapsed for sending a message to the neighbouring node, based on the current time received at the start of the algorithm. If the timer elapses and the probability of sending an INV message is satisfied, the node will send the INV message to the neighbouring node and increment the INV messages sent to that neighbour counter. This is to ensure that the data used in algorithm 1 to calculate the probability of sending an INV message to neighbouring nodes is up to date. However, if the timer elapsed but the probability of sending is not satisfied, the INV message scheduled to be sent to the node is ignored, with the contents on the message discarded and the ignored messages count increased.

Algorithm 2 Broadcast Inventory Messages

```
1: function BROADCAST_INVS_PROB_FLOODING(myself)
2:     now ← get_current_time()
3:     for node in neighbourhood do
4:         probability_to_send ← get_probability(myself, node)
5:         time_to_send ← get_time_to_send(node)
6:         timeout ← now > time_to_send
7:         send_inv_based_on_prob ← random.random() < probability_to_send
8:         if timeout and send_inv_based_on_prob then
9:             sim.send(myself, node, INV_message)
10:            myself.increaseInvSentToNeighbour(node)
11:        if timeout and not send_inv_based_on_prob then
12:            myself.deleteTransactionQueue(node)
13:            myself.increaseIgnoredMessagesCount()
```

5 Evaluation

In this chapter, the results of the simulations for the proposed changes to the Bitcoin flooding protocol to a probabilistic flooding approach is presented. The probabilistic flooding approach is evaluated by comparing the results gathered from the simulations when the probabilistic flooding approach was implemented, to the results from the simulations when the normal Bitcoin flooding protocol was implemented.

As mentioned in section 4.3, several important metrics are recorded during the simulations. The recorded metrics are essential in order to compare and contrast the proposed probabilistic flooding approach to the flooding mechanism currently implemented in the Bitcoin network.

The most relevant and important metrics to compare the two protocols are:

- **Percentage of Committed Transactions** is the most vital metric when comparing the two protocol changes. The percentage of committed transactions indicates whether or not every transaction that was created during the simulation period was eventually committed into a block. As Bitcoin is the most popular cryptocurrency and has a market cap of approximately 72$ billion, it is essential that every transaction that is created is eventually committed in a block to maintain the reliability of the system. The main objective of the protocol change is to reduce the number of redundant messages being exchanged on the network. However, if the protocol change negatively impacts the percentage of committed transactions, reducing the 100% commitment rate of transactions then regardless of the potential reduction of redundant messages, a less then 100% committed transactions rate would be detrimental to the system and unacceptable.
- **Total Number of Messages Sent Per Node** is an important metric when comparing the two protocols. As the main objective of the probabilistic flooding approach is to reduce the number of redundant messages exchanged on the network, comparing the total number of messages sent per node between the two protocols would indicate exactly how many messages were saved as a result from the protocol.

- **The Commit Time of Transactions** is also an important metric to consider when comparing the two protocols. The commit time represents the time between when a transaction was created to when it was placed in a block. As commit times of transactions is an extremely important aspect in cryptocurrencies, having an increased commit time when implementing the proposed probabilistic flooding approach may not be worth the tradeoff in potential messages saved within the network.

5.1 Results

The results that are presented in this section are based on a 58 hour simulation time, where the first 5 hours and the last 5 hours are discarded in order to observe the system at a stable state. The network is comprised of 625 nodes, 5 of which are miners. Each node in the network may have a different number of neighbours, however, there is a minimum of 8 neighbours and a maximum of 125 neighbours a node can have when participating in the network.

5.1.1 Percentage of committed transactions

As mentioned previously, the most important metric when comparing the proposed probabilistic flooding protocol to the current Bitcoin flooding protocol is the percentage of committed transactions. Both protocols produced a 100% transaction commitment rate, committing every transaction to a block during the simulation period. The results are displayed in figure 5.1 and we can conclude from these results that the adjustment of the flooding protocol to a probabilistic flooding approach did not have an effect on the number of committed transactions during the simulation. A 100% transaction commitment rate ensures that the system remains reliable when the probabilistic flooding approach is implemented.

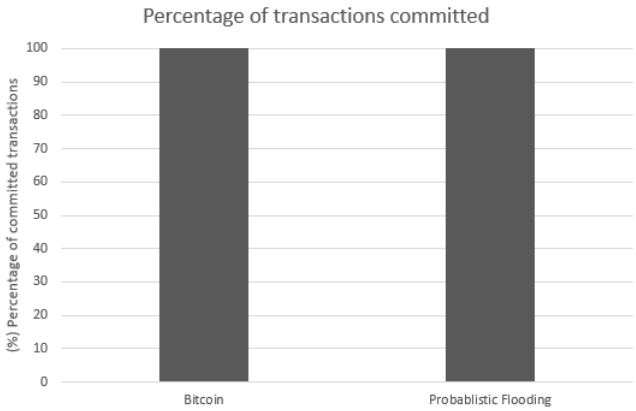

Figure 5.1: Comparison of the Percentage of Transactions committed.

5.1.2 Transaction Commit Time

Another important metric that was previously discussed when comparing the two protocols is the time taken to commit a transaction. Figure 5.2 represents the average time taken for a transaction to be committed into a block for the two protocols. As you can see from figure 5.2, the time taken for a transaction to be committed into a block for both protocols were very similar. The small difference between the two protocols is negligible. This is very important as the results indicate that changing the flooding protocol to a probabilistic flooding approach does not have an effect on the transaction commitment time. As transaction commit time is an extremely important aspect for cryptocurrencies, if there was a significant increase in transaction commit time when changing to the probabilistic flooding approach, the potential reduction in redundant messages may not be worth the trade-off in increased transaction commit time.

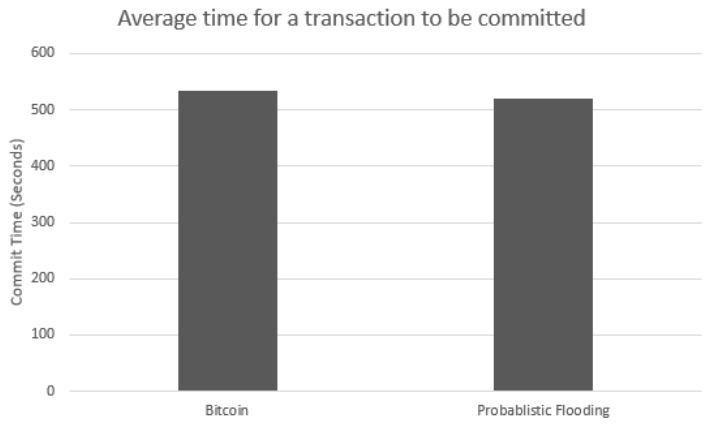

Figure 5.2: Comparison of the average time taken to commit a transaction between the two protocols.

5.1.3 Total Sent Messages

As mentioned in section 1.1, the main objective of changing the current bitcoin flooding protocol to the probabilistic flooding approach is to reduce the number of redundant and duplicated messages that are currently being generated in the Bitcoin network. Figure 5.3 represents the number of total sent messages per node gathered from our simulations for the two protocols.

The results show that when running the probabilistic flooding approach, there was a significant decrease in the total number of messages sent per node during the simulation. When running the Bitcoin flooding protocol, the simulation showed that there were approximately 790,000 total messages sent per node, whereas when the simulation was run with the probabilistic flooding protocol implemented, there were approximately 675,000 total sent messages per node. This results in a **15%** reduction

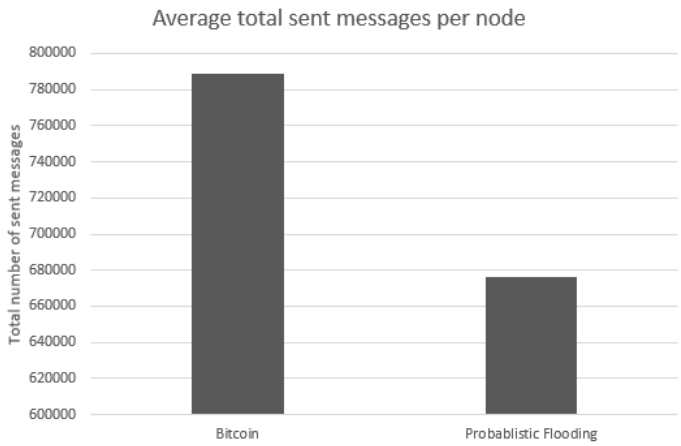

Figure 5.3: Comparison of the average total sent messages per node.

in the total number of messages sent per node during the simulation period. As the 115,000 reduction in messages mentioned is the number of messages saved per node, the total number of messages saved throughout the entire network can be estimated at approximately **70.5** million messages as there are 625 nodes participating in the network during the simulation. Figure 5.4 represents the total number of sent messages during the simulation period for both protocols.

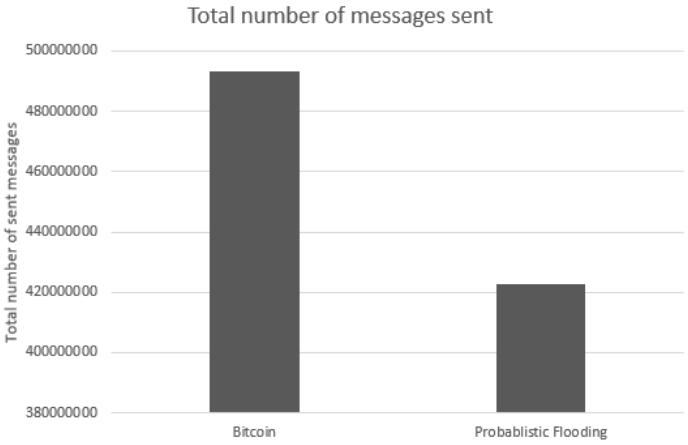

Figure 5.4: Comparison of the total number of sent messages during the simulation period between the two protocols.

5.1.4 Other Metrics

As mentioned in section 1.1, the main objective of switching from the current Bitcoin flooding protocol to a probabilistic flooding approach is to reduce the number of redundant messages that are generated within the network. However, when implementing this change in protocol, we need to also ensure that the reliability and resilience of the system are not affected. Metrics such as the percentage of transactions committed and transaction commit time is a good indicator of ensuring

the reliability of the system has not been affected as a result of the protocol change. However, there are other metrics such as the number of blocks/forks created, that can provide additional information on whether or not changing flooding protocols has had an undesired effect on the system.

Figure 5.6 shows the number of blocks created for both protocols during the simulation. The number of blocks created during the simulation of both protocols are very similar, with the difference between the two minuscule and negligible. As a key concept of Bitcoin is to generate a block at an average time of 10 minutes, we need to ensure that the change in protocols does not affect the 10-minute block creation time. There were approximately 350 blocks created by both protocols during the 58-hour simulation. If a block was created every 10 minutes during this 58 hour period, a total of 348 blocks would be created. The difference between the number of blocks created during the simulation and the theoretical number of blocks that should have been created is negligible. The slight difference in the number of blocks created between both protocols and the theoretical number of blocks can be attributed to the fact that not every block is mined every 10 minutes. A block can be mined in 2 minutes if a miner gets lucky and solves the proof-of-work or it could take 15 minutes to mine a block. This variance in times needed to create a block explains the difference in blocks generated during the simulation.

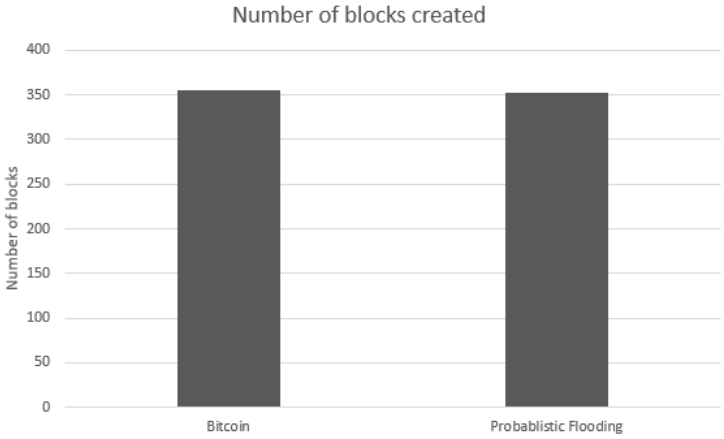

Figure 5.5: Comparison of the number of blocks created during the simulation period.

Another metric that can be used to ensure the resilience and reliability of the system is not affected or compromised by the proposed changes is the number of forks generated during the simulation. Figure 5.6 represents the number of forks generated during the simulation for both protocols. As with the previous metrics, the results show that there is a small and negligible difference between the two protocols when discussing the number of forks generated. The results indicate that changing the current Bitcoin flooding protocol to the probabilistic flooding approach does not affect the fork rate in the system.

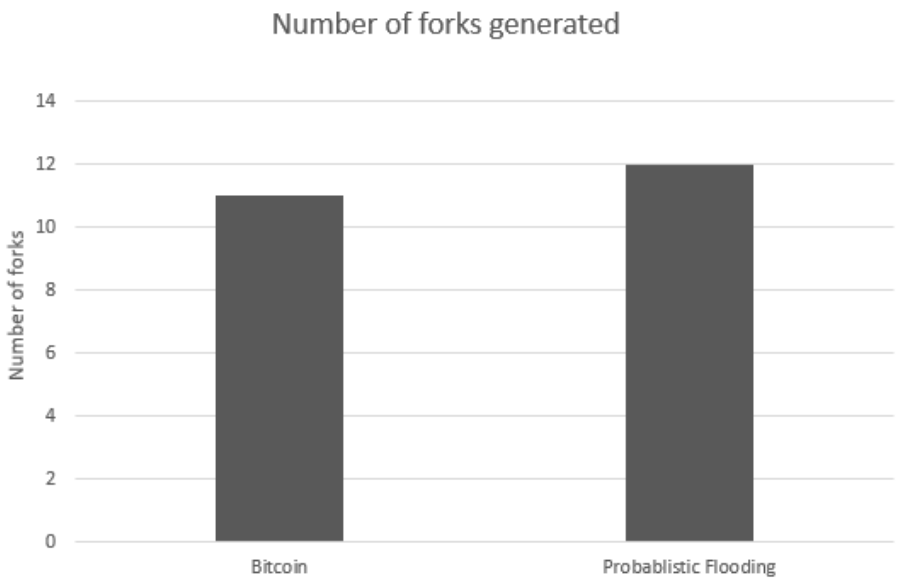

Figure 5.6: Comparison of the number of forks created during the simulation period between the two protocols.

5.1.5 Varying the Probability Percentage

The above results are based on the current Bitcoin flooding protocol and the probabilistic flooding protocol, with the probabilities calculated from the formula in section 4.2. However, the results described below are based on setting the probability of all neighbours in the network to a default percentage, as opposed to a calculated percentage based on the number of INV messages sent and GETDATA messages received in return. The purpose of setting default probabilities is to determine at what percentage does the number of non-committed transactions increase drastically which in turn creates an unreliable system.

Figure 5.7 represents the total number of sent messages per node during the simulation for the default percentages tested - 60%, 50%, 40%, 30%, as well as the current Bitcoin protocol and the probabilistic flooding protocol. The chart indicates that as the default percentage of sending an INV message to neighbours in the network decreases, the total number of messages exchanged in the network also decreases as a result. With every 10% decrement in percentage, the total number of messages sent also decreases proportionally. The default percentages all produce a total number of sent messages that is lower than both the Bitcoin protocol and the probabilistic flooding protocol, however as mentioned in section 1.1, the reliability and resilience of the system must not be negatively affected by the changes.

The reliability of the system is of the utmost importance as if Bitcoin were to ever be adopted by the mainstream. A user must have full confidence that when they make a transaction in the network that eventually the transaction will be committed into a

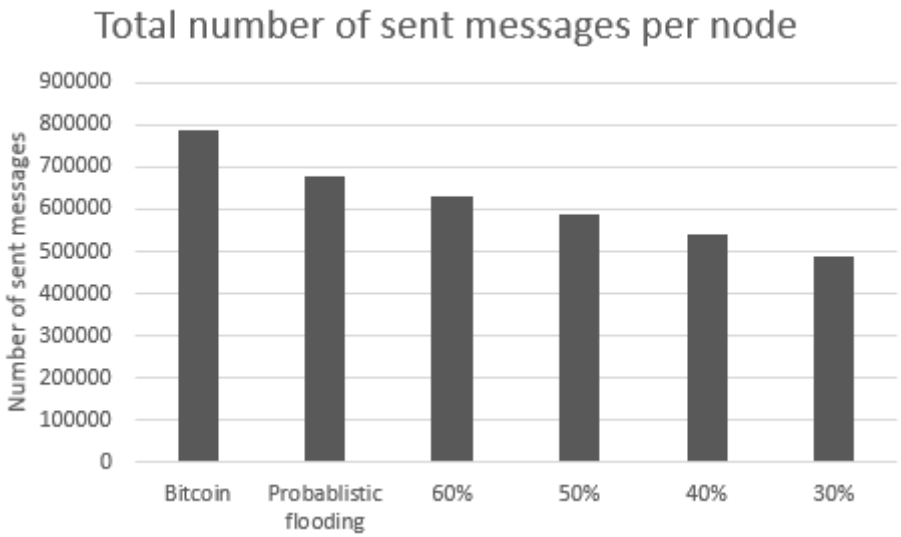

Figure 5.7: Comparison of the total number of sent messages.

block. Figure 5.8 represents the number of uncommitted transactions for the default percentages tested along with the Bitcoin and probabilistic flooding protocols. The chart shows that for the Bitcoin and probabilistic flooding protocol, there are 0 uncommitted transactions during the simulation period.

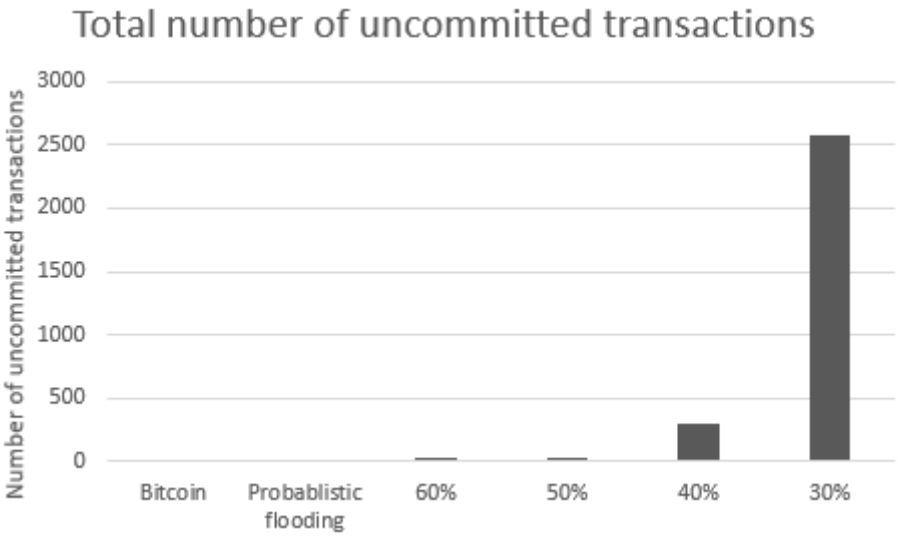

Figure 5.8: Comparison of the total number of uncommitted transactions.

The default percentage of 60% and 50% indicate a small number of uncommitted transactions, which can not be seen as negligible as even just 1 uncommitted transaction indicates that the system is unreliable. However, when the default percentage is set to 40% and 30%, the total number of uncommitted transactions increases exponentially, as seen in figure 5.9.

The results indicate that around 40% and 30% is when the system becomes extremely

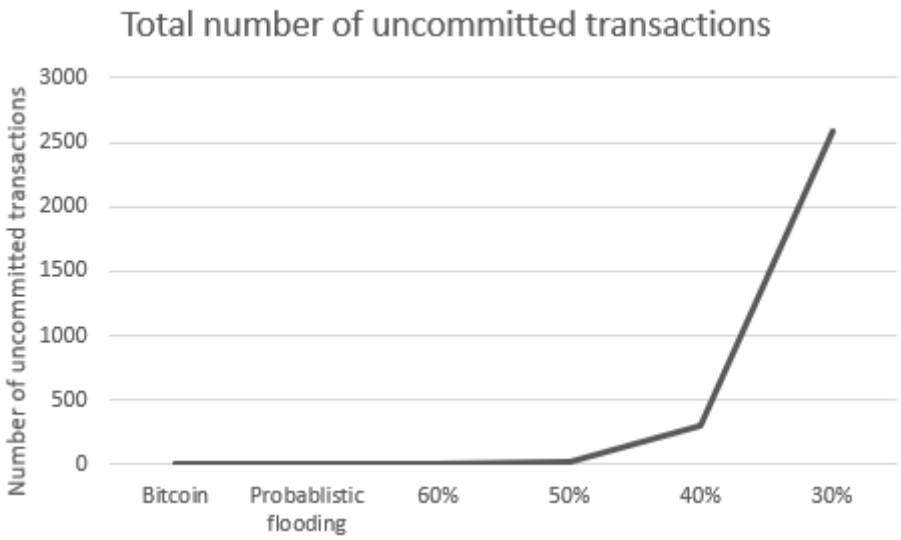

Figure 5.9: Chart indicating the exponential increase of uncommitted transactions.

unreliable, leading to an exponential increase in the number of uncommitted transactions. Although setting the percentage to a lower default percentage results in a lower total number of messages exchanged in the network during the simulation, it also leads to an unreliable system. As mentioned in section 1.1, the unreliability introduced by setting the default percentage is a huge factor as the main goal was to reduce the number of redundant messages whilst ensuring that the reliability and resilience of the system were not negatively affected. Therefore, using a default percentage of the percentages tested above could not be used in the Bitcoin system.

6 Conclusion

The main aim of this dissertation was to adjust the current flooding protocol implemented by Bitcoin to a more efficient protocol, with the aim of reducing the large number of redundant messages and duplicated messages being generated by the current flooding protocol, whilst maintaining the reliability and resilience of the Bitcoin system. As Bitcoin continues to grow in popularity and more users actively join and participate in the network, it is essential to make the system, and the protocols implemented by the system, as efficient as possible.

In this dissertation, we proposed a novel protocol that aims to reduce the number of redundant messages being generated by the current flooding protocol. The proposed protocol changes the current flooding protocol implemented by Bitcoin to a probabilistic flooding approach. The proposed probabilistic flooding approach presented in this dissertation is based on the idea that well-connected neighbours will more likely not respond to an INV message compared to a node that is less-connected, therefore the probability of sending an INV message to a less-connected node is higher than that of a well-connected node.

As we have shown in section 5, the proposed protocol is able to significantly reduce the total number of messages being exchanged on the network, whilst maintaining the reliability of the system. The number of INV messages sent per node and the total number of messages sent per node decreased by **29%** and **14%** respectively when running the probabilistic flooding protocol. During the 58 hour simulation period, the total number of messages saved when running the probabilistic flooding approach when compared to the current flooding protocol was approximately 70 million messages.

Nodes will benefit from the change in protocol to a probabilistic flooding approach as it reduces the number of redundant messages that are being generated and received by each node. The reduction in redundant messages will lead to a decrease in power consumption and CPU cycles being wasted on duplicated messages. When attempting to solve the proof-of-work to mine a block in Bitcoin, the power consumption required to attempt to mine a block is huge. The consumption of energy to mine a block,

therefore, leads to a large electricity bill. By reducing the number of redundant messages being exchanged on the network and the cycles being wasted on duplicated messages, the decrease in power consumption will lead to a lower electricity bill. As a result of the decrease in costs of running a mining node, the hope is to make mining more accessible and encouraging more nodes to join the network, making the network more stable.

The novel protocol proposed in this dissertation met the objectives that it aimed to achieve - reducing the number of redundant messages on the network whilst maintaining the reliability and resilience of the system. We have shown that the current flooding protocol implemented by Bitcoin for the dissemination of information across the network is inefficient and wasteful, and have proved that the mechanism can be improved upon whilst maintaining the reliability and integrity of the system. This leads to many possible, alternate flooding solutions to the current flooding protocol for future work.

Bibliography

[1] Sarah Sharkey. Alt-pow: An alternative proof-of-work mechanism.

[2] Bitcoin official developer website, . `https://bitcoin.org/en/`.

[3] Blockchain website for bitcoin statistics. `https://www.blockchain.com`.

[4] Satoshi Nakamoto et al. Bitcoin: A peer-to-peer electronic cash system. 2008.

[5] Andreas M. Antonopoulos. *Mastering Bitcoin*. O'Reily Media, 201r.

[6] Varun Deshpande, Hakim Badis, and Laurent George. Btcmap: Mapping bitcoin peer-to-peer network topology. In *2018 IFIP/IEEE International Conference on Performance Evaluation and Modeling in Wired and Wireless Networks (PEMWN)*, pages 1–6. IEEE, 2018.

[7] Christian Decker and Roger Wattenhofer. Information propagation in the bitcoin network. In *IEEE P2P 2013 Proceedings*, pages 1–10. IEEE, 2013.

[8] `https://github.com/bitcoin/bips/blob/master/bip-0152.mediawiki`.

[9] Mauro Conti, E Sandeep Kumar, Chhagan Lal, and Sushmita Ruj. A survey on security and privacy issues of bitcoin. *IEEE Communications Surveys & Tutorials*, 20(4):3416–3452, 2018.

[10] Vitalik Buterin et al. Ethereum white paper. *GitHub repository*, pages 22–23, 2013.

[11] Joao Esteves Marçal. Adaptive information dissemination in the bitcoin network.

[12] . `http://p2pfoundation.ning.com/forum/topics/bitcoin-open-source`.

[13] Darrel Hankerson and Alfred Menezes. *Elliptic curve cryptography*. Springer, 2011.

[14] Miraje Gentilal, Paulo Martins, and Leonel Sousa. Trustzone-backed bitcoin wallet. In *Proceedings of the Fourth Workshop on Cryptography and Security in Computing Systems*, pages 25–28. ACM, 2017.

[15] Bitcoin wikipedia, . https://en.bitcoin.it/wiki/.

[16] Joseph Bonneau, Andrew Miller, Jeremy Clark, Arvind Narayanan, Joshua A Kroll, and Edward W Felten. Sok: Research perspectives and challenges for bitcoin and cryptocurrencies. In *2015 IEEE Symposium on Security and Privacy*, pages 104–121. IEEE, 2015.

[17] Yi Liu, Xiayang Chen, Lei Zhang, Chaojing Tang, and Hongyan Kang. An intelligent strategy to gain profit for bitcoin mining pools. In *2017 10th International Symposium on Computational Intelligence and Design (ISCID)*, volume 2, pages 427–430. IEEE, 2017.

[18] Daniel Augot, Hervé Chabanne, Olivier Clémot, and William George. Transforming face-to-face identity proofing into anonymous digital identity using the bitcoin blockchain. In *2017 15th Annual Conference on Privacy, Security and Trust (PST)*, pages 25–2509. IEEE, 2017.

[19] . https://bitcoin.org/en/bitcoin-core/.

[20] Ethan Heilman, Alison Kendler, Aviv Zohar, and Sharon Goldberg. Eclipse attacks on bitcoin's peer-to-peer network. In *24th {USENIX} Security Symposium ({USENIX} Security 15)*, pages 129–144, 2015.

[21] Andrew Miller, James Litton, Andrew Pachulski, Neal Gupta, Dave Levin, Neil Spring, and Bobby Bhattacharjee. Discovering bitcoin's public topology and influential nodes. *et al*, 2015.

[22] R Bala and R Manoharan. Security enhancement in bitcoin protocol. In *2018 International Conference on Wireless Communications, Signal Processing and Networking (WiSPNET)*, pages 1–4. IEEE, 2018.

[23] Cian Burns. Bitcoin anonymity and the block chain.

[24] Ghassan Karame, Elli Androulaki, and Srdjan Capkun. Two bitcoins at the price of one? double-spending attacks on fast payments in bitcoin. *IACR Cryptology ePrint Archive*, 2012(248), 2012.

[25] Nicolas T Courtois and Lear Bahack. On subversive miner strategies and block withholding attack in bitcoin digital currency. *arXiv preprint arXiv:1402.1718*, 2014.

[26] Ittay Eyal and Emin Gün Sirer. Majority is not enough: Bitcoin mining is vulnerable. *Communications of the ACM*, 61(7):95–102, 2018.

[27] Muntadher Fadhil, Gareth Owenson, and Mo Adda. A bitcoin model for evaluation of clustering to improve propagation delay in bitcoin network. In *2016 IEEE Intl Conference on Computational Science and Engineering (CSE) and IEEE Intl Conference on Embedded and Ubiquitous Computing (EUC) and 15th Intl Symposium on Distributed Computing and Applications for Business Engineering (DCABES)*, pages 468–475. IEEE, 2016.

[28] Gareth Owenson, Mo Adda, et al. Proximity awareness approach to enhance propagation delay on the bitcoin peer-to-peer network. In *2017 IEEE 37th International Conference on Distributed Computing Systems (ICDCS)*, pages 2411–2416. IEEE, 2017.

[29] Hassan Barjini and Mohmed Othman. Smoothflood: Decreasing redundant messages and increasing search quality of service in peer-to-peer networks. In *2010 International Conference on Information Retrieval & Knowledge Management (CAMP)*, pages 138–142. IEEE, 2010.

[30] Giulia Fanti and Pramod Viswanath. Anonymity properties of the bitcoin p2p network. *arXiv preprint arXiv:1703.08761*, 2017.