

# **Evaluation of Security and Performance of Clustering in the Bitcoin Network, with the Aim of Improving the Consistency of the Blockchain**



**Muntadher Fadhil Sallal**

A thesis submitted for the degree of  
*Doctor of Philosophy*

The Networking Research Group  
School of Computing  
University of Portsmouth

December 2018



## **Declaration**

Whilst registered as a candidate for the above degree, I have not been registered for any other research award. The results and conclusions embodied in this thesis are the work of the named candidate and have not been submitted for any other academic award.

Muntadher Fadhil Sallal

December 2018



## **Acknowledgements**

First and foremost, I would like to thank Allah (God) for all what I am and all what I have. I would like to particularly thank my supervision team- Dr Gareth Owenson and Dr Mo Adda for the continuous support, encouragement and advice. Special thanks must go to my mother Rabab Ridha, grandfather(Ridha Serhan), grandmother as well as my brothers and sisters (Hayder, Azhar, Hawraa, and Zufran) for the financial support during my academic studies in the past as well as the spiritual support and encouragement for my embarking on the PhD research in the past two and half an years. In addition, I would also like to thank my best friends Ali Malik and Hayder Murad for the continuous support and encouragement during my PhD study that have facilitated significantly my involvement in the writing process for this thesis.

Last but not least, my unique gratefulness goes out to Safa Shubbar, my precious gift from God; my best friends Dawood, Ahmad Shelaka, Mohanad, Neamah and Hani ben Hussain, they have always been there when I needed them. Without whom I would have struggled to find the inspiration and motivation needed to complete this thesis.

God bless you all.



## **Dissemination**

### **Journal Articles**

Sallal, M., Owenson, G., Adda, M. (2017). Towards Bitcoin Scalability: A Study to Improve Propagation Delay in Bitcoin Peer-to-Peer Network, IEEE Access Journal.(Under Review)

Sallal, M., Owenson, G., Adda, M. (2018). Security evaluation of Clustering in the Bitcoin Peer-to-Peer Network, International Journal of Ad Hoc and Ubiquitous Computing.

### **Conference papers**

Fadhil, M., Owenson, G., Adda, M. (2016). Bitcoin network measurements for simulation validation and parametrisation. In International networking conference. Frankfurt/ Germany.

Fadhil, M., Owenson, G., Adda, M. (2016). A Bitcoin model for evaluation of clustering to improve the transaction propagation delay in Bitcoin network. In 19th IEEE International Conference on Computational Science and Engineering. Paris.

Fadhil, M., Owenson, G., Adda, M. (2017, May). Locality based approach to improve propagation delay on the Bitcoin peer-to-peer network. In Integrated Network and Service Management (IM), 2017 IFIP/IEEE Symposium on (pp. 556-559). IEEE.

Fadhil, M., Owenson, G., Adda, M. (2017, June). Proximity awareness approach to enhance propagation delay on the Bitcoin peer-to-peer network. In Distributed Computing Systems (ICDCS), 2017 IEEE 37th International Conference on (pp. 2411-2416). IEEE.

Sallal, M., Owenson, G., Adda, M. (2018, October). Optimisation Clustering Protocol For Fast Information Propagation in the Bitcoin Peer-to-Peer Network. In the International Symposium on DIStributed Computing (DISC). New Orleans/USA.

## **Presentations**

“Bitcoin Network Clustering to Support a Higher Bitcoin Network Scalability”, Computing Seminar, University of Portsmouth, 11 November 2015.

“Bitcoin Network Measurements for Simulation Validation and Parameterisation”, Computing Seminar, University of Portsmouth, 16 March 2016.

“A Bitcoin Model for Evaluation of Clustering to Improve Propagation Delay in Bitcoin Network ”, Faculty of Technology Research Conference, University of Portsmouth, 7 June 2016.

“Optimisation Clustering Protocols For Fast Information Propagation in The Bitcoin Peer-to-Peer Network ”, Graduate school, University of Portsmouth, 11 October 2016.

“Locality Based Approach to Improve Propagation Delay on the Bitcoin Peer-to-Peer Network ”, Computing Seminar, University of Portsmouth, 1 November 2017.

## **Posters**

“Bitcoin Network Clustering to Support a Higher Bitcoin Network Scalability”, Computing student conference, University of Portsmouth, 15 March 2016.

“A Bitcoin Model for Evaluation of Clustering to Improve Propagation Delay in Bitcoin Network”, Faculty of Technology Research Conference, University of Portsmouth, 7 June 2016.

“Locality Based Approach to Improve Propagation Delay on the Bitcoin Peer-to-Peer Network ”, Integrated Network and Service Management (IM), 2017 IFIP/IEEE Symposium.



## Abstract

Bitcoin is a digital currency based on a peer-to-peer network to propagate and verify transactions. Bitcoin is gaining wider adoption than any previous crypto-currency and many well-known businesses have begun accepting bitcoins as means of financial payments. However, the mechanism of peers randomly choosing logical neighbors without any knowledge about the underlying physical topology can cause a delay overhead in information propagation which makes the system vulnerable to double spend attacks due to inconsistencies in the blockchain. Aiming at alleviating the propagation delay problem, this thesis evaluates the concept of network clustering in tackling the propagation delay problem in the Bitcoin network throughout introducing a proximity-aware extensions to the current Bitcoin protocol, named Locality Based Clustering (LBC), Ping Time Based Clustering (BCBPT), Super Node Based Clustering (BCBSN), and Master Node Based Clustering (MNBC). The ultimate purpose of the proposed protocols, that are based on how clusters are formulated and nodes define their membership, is to improve the information propagation delay in the Bitcoin network. The proximity of connectivity in the Bitcoin network is increased in the LBC and BCBPT protocol by grouping Bitcoin nodes based on different criteria, physical location in LBC protocol and link latencies between nodes in the BCBPT. In the BCBSN protocol, geographical connectivity increases as well as the number of hops between nodes decreases through assigning one node to be a cluster head that is responsible for maintaining the cluster. Whereas, MNBC incorporates master node technology and proximity-awareness into the existing Bitcoin protocol with the aim of creating fully connected clusters based on physical Internet proximity. We show, through simulations, that the proposed approaches define better clustering structures that optimize the transaction propagation delay over the Bitcoin protocol. However, MNBC is more effective at reducing the transaction propagation delay compared to the BCBPT, LBC, and BCBSN.

On the other hand, this thesis evaluates the resistance of the Bitcoin network and the proposed approaches against the partitioning attack. Even though the Bitcoin network is more resistant against partition attacks than the proposed approaches, more resources need to be spent to split the network in the proposed approaches especially with a higher number of nodes.

Finally, this thesis introduces a novel methodology to measure the transaction propagation delay in the real Bitcoin network with the aim of validating any model of the Bitcoin network. Transaction propagation measurements show that the transaction propagation time is significantly affected by the number of connected nodes and the network topology which is not proximity defined. In addition, large-scale measurements of the real Bitcoin network are performed in thesis with the aim of providing an opportunity to parameterise any model of the Bitcoin network accurately. Furthermore, this thesis presents a simulation model of the Bitcoin peer-to-peer network which is an event based simulation.

# Table of contents

<b>List of figures</b>	<b>xv</b>
<b>1 Introduction</b>	<b>1</b>
1.1 Digital Currency as Alternative Currency . . . . .	2
1.1.1 "Digital" versus "Virtual" . . . . .	2
1.1.2 History of Digital Currencies . . . . .	2
1.2 Cryptocurrencies . . . . .	4
1.3 The rise of Bitcoin . . . . .	5
1.4 Overview of how Bitcoin works . . . . .	6
1.5 Distributed trust in the Bitcoin . . . . .	9
1.6 Motivation . . . . .	9
1.7 Problem Statement . . . . .	10
1.8 Contributions . . . . .	11
1.9 Research methodology . . . . .	13
1.9.1 Scientific Method . . . . .	13
1.9.2 Computer Simulation . . . . .	14
1.9.3 Key research methodology stages . . . . .	16
1.10 Thesis Outline . . . . .	17
<b>2 Literature Review</b>	<b>19</b>
2.1 Background . . . . .	19
2.1.1 The Bitcoin Network . . . . .	20
2.1.1.1 The Peer-to-peer Networks . . . . .	21
2.1.1.2 Bitcoin network structure . . . . .	22
2.1.1.3 Bitcoin network discovery . . . . .	24
2.1.1.4 DNS seed nodes in the Bitcoin network . . . . .	27
2.1.2 Bitcoin protocol . . . . .	27
2.1.2.1 Transactions . . . . .	28

2.1.2.2	Blocks . . . . .	28
2.1.2.3	Blockchain & Blockchain forks . . . . .	30
2.1.2.4	Miners . . . . .	34
2.2	The role of broadcast in the Bitcoin network . . . . .	35
2.3	Information propagation delay . . . . .	37
2.4	Related Work . . . . .	40
2.4.1	Minimize verification . . . . .	40
2.4.2	Pipelining information propagation . . . . .	43
2.4.3	Connectivity increase . . . . .	45
2.4.4	Mitigating double spending attacks . . . . .	46
2.4.5	Mitigating propagation delay outside the electronic currency field: . .	47
2.5	Conclusion . . . . .	49
<b>3</b>	<b>The Proposed Clustering Approaches</b>	<b>51</b>
3.1	Introduction . . . . .	51
3.2	Locality Based Clustering (LBC) . . . . .	51
3.2.1	Localised Cluster Generation . . . . .	53
3.2.2	Cluster Maintenance . . . . .	54
3.3	Ping Time Based Approach(BCBPT) . . . . .	55
3.3.1	Distance calculation . . . . .	55
3.3.2	Cluster Maintenance . . . . .	57
3.4	Super Node Based Approach (BCBSN) . . . . .	57
3.4.1	Super peer selection algorithm . . . . .	59
3.4.2	The peer joining algorithm . . . . .	60
3.5	Master Node Based Clustering (MNBC) . . . . .	62
3.5.1	Master Node Selection . . . . .	63
3.5.2	Cluster Maintenance . . . . .	64
3.6	Conclusion . . . . .	65
<b>4</b>	<b>Bitcoin Network Measurements and Simulation Model</b>	<b>67</b>
4.1	Introduction . . . . .	67
4.2	Bitcoin Network Simulation Model . . . . .	67
4.2.1	Bitcoin Network measurements and Parameterisation . . . . .	68
4.2.1.1	Session Length . . . . .	69
4.2.1.2	Link Latencies . . . . .	70
4.2.1.3	The size of the Bitcoin network . . . . .	73
4.2.2	Bitcoin Model Structure & Validation . . . . .	73

4.2.2.1	The Model Validation . . . . .	75
4.3	Conclusion . . . . .	79
<b>5</b>	<b>Performance and Security Evaluation</b>	<b>81</b>
5.1	Introduction . . . . .	81
5.2	Performance evaluation . . . . .	81
5.2.1	Experiments setup . . . . .	81
5.2.2	Results and discussions . . . . .	84
5.3	Security Evaluation . . . . .	88
5.3.1	Experiment setup . . . . .	89
5.3.2	Results and discussions . . . . .	89
5.4	Conclusion . . . . .	93
<b>6</b>	<b>Conclusions and Future Work</b>	<b>95</b>
6.1	Introduction . . . . .	95
6.2	Addressing the research questions . . . . .	96
6.3	Summary of conducting this PhD research . . . . .	98
6.4	Scientific and practical importance . . . . .	100
6.5	Future work . . . . .	100
6.5.1	Reduce the size of the blockchain . . . . .	101
6.5.2	Observing blockchain forks . . . . .	101
6.5.3	Evaluating the network overhead . . . . .	102
	<b>References</b>	<b>103</b>
	<b>Appendix Appendix I : Transaction propagation delay measurements in the real Bit-coin network</b>	<b>111</b>
	<b>Appendix Appendix II: the confirmation time costs for the 25, 50 and 75 percentile for 1000 transactions</b>	<b>113</b>
	<b>Appendix Appendix III: UPR16 Form</b>	<b>115</b>
	<b>Appendix Appendix IV: Bitcoin Crawler</b>	<b>117</b>
	<b>Appendix Appendix IIII: Bitcoin Simulator</b>	<b>131</b>



# List of figures

1.1	Development History of Digital Currencies . . . . .	3
1.2	Life cycle of transaction process . . . . .	8
1.3	The iterative nature of the Hypothetico-Deductive Method,(Dodig-Crnkovic, 2002)(Grimes, 1990) . . . . .	14
1.4	Steps in a discrete simulation study (Law, 2003),(Law et al., 1991), (Bar-Noy et al., 1995), (Peart, 2014) . . . . .	15
1.5	depicts a conceptual model of key research methodology stages . . . . .	17
2.1	Major components of the Bitcoin payment system . . . . .	20
2.2	Bitcoin network . . . . .	22
2.3	A Bitcoin network node with all four functions: Wallet, Miner, full Blockchain database, and Network routing . . . . .	23
2.4	Initial Handshake mechanism between Nodes A and B . . . . .	24
2.5	Dissemination of <i>Addr message</i> between two peers . . . . .	26
2.6	Bitcoin transaction . . . . .	29
2.7	Bitcoin Blockchain . . . . .	31
2.8	Double spending (race attack) . . . . .	33
2.9	Information propagation mechanism between Nodes A and B . . . . .	36
2.10	The transaction A appears in two branches A and B . . . . .	38
2.11	Disagreement between the Bitcoin network nodes about which transaction is valid . . . . .	39
2.12	Sidechains with SPV proof (Back et al., 2014) . . . . .	42
2.13	First scenario where tx arrives before GETDATA . . . . .	44
2.14	Second scenario where GETDATA arrives before tx . . . . .	44
3.1	Example of Localized clusters creation. The black circles represent the border nodes between clusters, while grey and red circles represent nodes in different clusters. . . . .	52

3.2	Example of super peer clusters creation. The Blue circles represents super peers in each cluster, while grey and red circles represent nodes in different clusters. . . . .	58
3.3	Example of master node clusters creation. The black circles represent the edge nodes between clusters, while grey and red circles represent nodes in different clusters. Blue circles represents master nodes in each cluster. . . . .	63
4.1	Session lengths of peers in the Bitcoin network . . . . .	70
4.2	Link latencies between the measurement node(located in Portsmouth/UK) and other peers In the Bitcoin network . . . . .	71
4.3	Link latencies between the measurement node(located in Los Angeles/US) and other peers In the Bitcoin network . . . . .	72
4.4	Bitcoin event based simulation representation . . . . .	74
4.5	Bitcoin simulator structure . . . . .	75
4.6	Illustration of propagation experimental setup . . . . .	76
4.7	Comparison of the distribution of $\Delta t_{c,n}$ as measured in the real Bitcoin network with simulation results . . . . .	77
4.8	Proportion of nodes that announced the transaction . . . . .	78
5.1	BCBSN simulation setup . . . . .	82
5.2	MNBC simulation setup . . . . .	83
5.3	LBC & BCBPT simulation setup . . . . .	84
5.4	Comparison of the distribution of $\Delta t_{c,n}$ measured in the simulated Bitcoin protocol with BCBPT protocol, LBC protocol, BCBSN, and MNBC Protocol simulation results. ( $d_t$ in BCBPT=25ms, $d_t$ in LBC=50km) . . . . .	85
5.5	Comparison of the distribution of $\Delta t_{c,n}$ as measured in the simulated BCBPT protocol with three thresholds ( $d_t$ =30ms, 60ms, 90ms ) . . . . .	87
5.6	Comparison of the distribution of $\Delta t_{c,n}$ as measured in the simulated LBC protocol with three thresholds ( $d_t$ =20 km, 50 km, 100 km) . . . . .	88
5.7	Number of honest peers on the minimum vertex cut . . . . .	90
5.8	Number of non-attacker peers on the minimum vertex cut during an attack with 7,000 honest peers parametrized as in the real-world network, attacker's session length $S_A = 6h$ . . . . .	91
5.9	Number of honest peers on the minimum vertex cut based on number of partitions in BCBSN, LBC, BCBPT, and MNBC . . . . .	92
6.1	Incorporating multichain protocol with clustering in the Bitcoin network . . . .	101



---

2	Box plot of transaction propagation delay distribution in the real Bitcoin network	111
3	Transactions' confirmation times . . . . .	113
4	. . . . .	115



# Chapter 1

## Introduction

Money is any item or verifiable record that can be accepted in a transaction that involves transfer of goods and service from one person to other ([Mishkin, 2007](#)) ([Durlauf et al., 2008](#)). Money has been represented by different forms over time, starting from a commodity that is a good whose physical value serves as a value of money, to fiat currency whose value serves less than the value it represents as money. In an economy, fiat money is used as a medium of exchange that includes currency bills and coins that are the most common and generally accepted as a unit of account ([Kim, 2016](#)). Therefore, trading with fiat currency requires physical interaction between parties (e.g. two people need to present in person in order to exchange money for a good). With the emergence of e-commerce, sending money can be done through money remittance agents or through banks for a fee, often can be for free, without the need for physical coins or notes.

Nowadays, the market has many innovative money payment systems which are built on several platforms such as mobile phones, internet, and digital storage cards. As these alternative payment systems have seen interest, more payment systems have continued emerging, such as PayPal, Apple Pay, Google Wallet, Transferwise, and others ([Chuen, 2015](#)).

Beyond the development in the payment systems that are based on fiat currency, a new currency generation is raised to support these payment systems with respect to allow faster, flexible, and more innovative payments that support trading goods and services ([Chuen, 2015](#)). Specifically, the adoption of new internet technologies that are supported by advances in encryption and network computing, have driven transformational changes in the global economy, including how goods and assets are exchanged. Furthermore, [Kleinrock \(1996\)](#) had a vision that the internet, advanced wireless technologies, portable and distributed computing will consider the vision of 'anytime', 'anywhere' which ensures anytime, anywhere access to computing and communications, for instance while one is in transit and/or when one reaches one's destination. This vision has come to fruition recently as remote collaboration is now possible between

organizations, and people can interact and communicate through the Internet. This development in the Internet technologies has also brought a new phenomenon, known as a digital currency, that uses the Internet technologies as a medium of money transfer and store. In the following section, background with respect to digital currency will be given.

## 1.1 Digital Currency as Alternative Currency

### 1.1.1 "Digital" versus "Virtual"

There is a misunderstanding around the terms 'digital' and 'virtual' as people often use them interchangeably, while each of these terms has a different meaning. Digital currencies can be described as a non-physical representation of traditional fiat currency that is stored and transferred electronically. Whereas virtual currencies are considered as a truly online asset that does not have a value in the physical reality, as it represents only a value in its virtual world ([Al Shehhi et al., 2014](#)). More interestingly, Euro, USD, and AED are deemed as an example of traditional currencies that can be stored electronically in order to shop online. While 'Pokécoins' in the game 'Pokémon GO', for instance, traditional currency (in digital form) requires to be converted to a virtual currency. The level of centralization is also deemed to be another key distinction between digital and virtual currencies. Clearly, digital currencies fall under the control of central banks and governments, while there is no central authority that controls virtual currencies as it can be created by any corporate such as Facebook, app makers, individual, so virtual currency lies outside of national fiscal policy. However, the main common factor between digital and virtual currencies is that both of these currencies are handled electronically ([He et al., 2016](#)).

### 1.1.2 History of Digital Currencies

There are several generations of digital currencies that have emerged and developed over the time, see Fig. 1.1 for more clarification. In the 1980, the idea of the digital money emerged and has adopted by many researchers. After a quarter of a century, the idea has become a reality and early digital currencies were introduced. These currencies, that are proposed in ([Chaum, 1983](#)), ([Law et al., 1996](#)), are quite similar to the traditional banking systems where the third party is used to settle all transactions at regular intervals. In addition, these currencies were almost controlled and targeted by governments who are concerned about the regulation of these currencies.

Later on, the first emergence sign of virtual currencies was when early digital currencies have replaced by approaches like B-money ([Dai, 1998](#)), Karma ([Vishnumurthy et al., 2003](#)),

and Bit Gold (Szabo, 2008), which are based on a cryptographic puzzle (proof of work) as a replacement for central authorities where those currencies can be created by any private developers in any corporate independently from a bank. These approaches still require a central authority to maintain the money ownership records. However, these virtual currencies are not denominated in fiat currency and have their own unit of account.

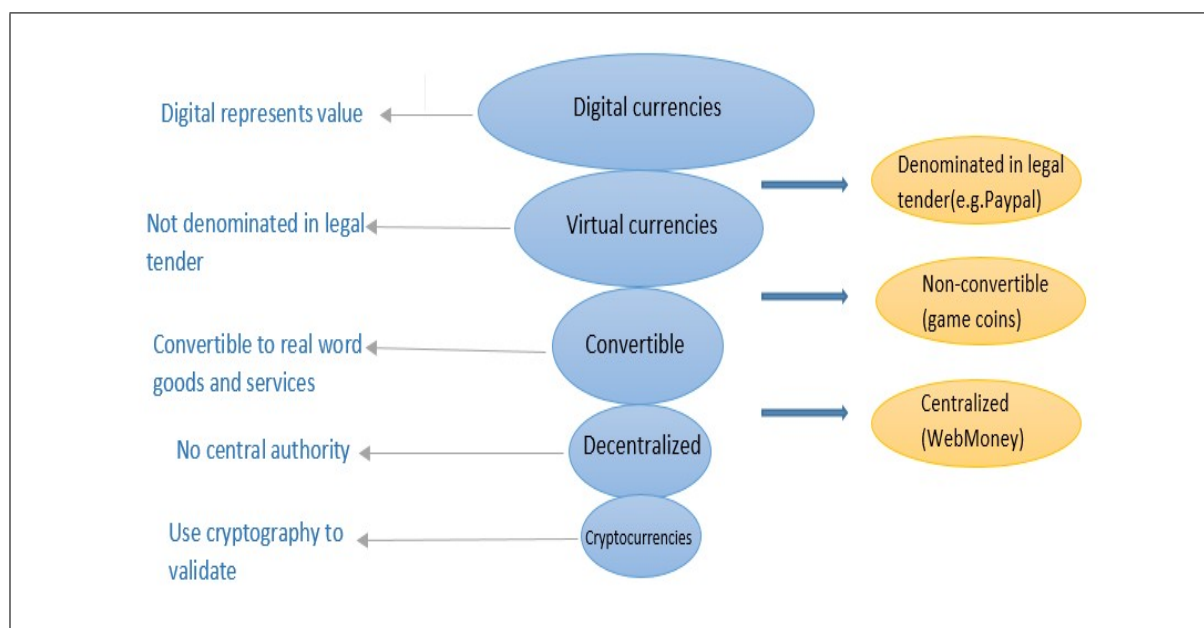


Fig. 1.1 Development History of Digital Currencies

After that, another generation of virtual currency has appeared. This generation comprises two key elements. First, the digital representation of "currency" that can be transferred between parties. Second, maintaining distributed ownership records in order to get rid of banks. Specifically, quorum systems which introduced the concept of voting, have been developed (Szabo, 1998). In this system, a correct value can be obtained by elections if the majority of peers (quorum) are honest (Malkhi & Reiter, 1998).

On the other hand, convertible virtual currencies have been developed to fill the gap of the previous virtual currencies being non-convertible currencies operated only within a self-contained virtual environment. Whereas convertible virtual currencies allow for the virtual currencies to be exchanged with fiat currencies (or other virtual currency) and used to pay for goods and services in the real world economy (He et al., 2016).

The development of virtual currencies has continued and contributed in the emergence of innovative distributed and decentralized virtual currencies. Cryptocurrencies is one of these currencies in which the governance is decentralized and they can be transacted with any outside agent. This kind of virtual currency does not fall under governments regulation as there is no

legal entity responsible for the activities. In the following section, cryptocurrencies will be discussed in details.

## 1.2 Cryptocurrencies

Cryptocurrencies are considered as a virtual payment system that generates a currency supply as well as tracks, verifies, and records transactions without involving any kind of central authority to act as an escrow. This payment system is adopted by businesses and individuals who prefer transacting over the Internet and not willing to supply 'Credit Cards' or 'Banking information'. Cryptocurrencies are based on cryptographic techniques that use digital signatures, the main reason why cryptocurrencies are named so. Despite all the positive aspects of cryptocurrencies such as transferring funds for long-distance easily without intermediaries, cryptocurrencies have negative aspects as well. Relative anonymity in cryptocurrencies could offer a great chance for illegal transactions and money laundering to be taken place ([Sharma et al., 2017](#)). However, the absence of the third party makes the regulatory agents unable to get involved in these systems which, on other hand, enables transactions to be exchanged without the fees attached. In contrast, the cost of transferring \$1,000.00 internationally via Credit card will be around 3% or \$30, and 3.9% or \$39 if the transfer will be done via Paypal ([Ahamad et al., 2013](#)). In the cryptocurrencies systems, cryptography is used to legitimize a user's claim to a value. Precisely, cryptographic digital signatures are used to prove the ownership of a digital asset through signing a digital transaction.

While the value of fiat currencies are supported by creditworthiness of governments and banks, there is no backing source for the value of cryptocurrencies. Instead, the value of cryptocurrencies is derived solely from the expectation that others would also value and use them ([Al Shehhi et al., 2014](#)). Moreover, cryptocurrencies are deemed to be "pseud-anonymous" as users who own these currencies are known by their virtual currency addresses which can not be linked to the real world identity. However, cryptocurrencies systems limit the number of currencies that might be ultimately issued.

Even though there are many types of cryptocurrencies such as Riple, Alcoins, Litecoins, etc, only one cryptocurrency grabbed a lot of attention as it made a worldwide impact. This currency is known as "*Bitcoin*" which is considered as a well-known example of cryptocurrency. In addition, Bitcoin is the first decentralised cryptocurrency that has a good reputation in the business field and large markets capitalization started using and accepting payments using this currency. In the following section, more details will be given about Bitcoin as an example of cryptocurrencies.

## 1.3 The rise of Bitcoin

In 2008, a paper was published on a cryptography mailing list by someone who used Satoshi Nakamoto as a pseudonym (Nakamoto, 2008). In this paper, a fully peer-to-peer electronic cash system, which is known as *Bitcoin*, was introduced. After the Bitcoin white paper, a proof of concept software client of Bitcoin was released in 2009 by an online community of computer scientists who studied cryptography (Bitcoin Wiki, 2008). Those scientists aimed to create an efficient and verifiable digital asset. When the Bitcoin network was launched in January 2009, majority of Bitcoin users were scientists who worked towards testing the software in order to verify Bitcoin's working parts (Ciaian et al., 2016). Over time, the use of Bitcoin increased greatly and many users have adopted Bitcoin as an official payment currency (Moore & Christin, 2013).

There are three key innovations that have been introduced in Bitcoin. These innovations can be outlined as follows: Distributed ownership ledger that records all the available digital assets in Bitcoin in a form of transactions that are verified by all users of Bitcoin. Unlike previous attempts to develop digital assets, the cryptographic payment system that is proposed by Nakamoto does not rely on a third party trust to verify the currency supply and transactions. Instead, Bitcoin developers built a decentralized network of computers that work concurrently towards achieving a common goal which is validating transactions in the Bitcoin network. The decentralized and distributed network allows every user to verify the validity of transactions based on a cryptographic protocol and the transactions history in Bitcoin that is shared over the entire Bitcoin network. The distributed transaction history, known as *blockchain*, is considered as one of the important innovations in Bitcoin which is stored locally on the computer hard drive of every user running a full version of the Bitcoin software. The blockchain is maintained through the proof of-work "mining" process. This process is achieved through running a special mining variant of the Bitcoin software that requires a great amount of computing power in order to win the right of adding information to the blockchain. Users who cover the mining task are named as *miners* (Antonopoulos, 2014).

The second key innovation in Bitcoin is *distributed currency supply* by which a unit of value on the Bitcoin network, known as *bitcoin* is released in the system. A bitcoin is represented by eight decimal places, and the smallest unit of bitcoin, known as a *satoshi*, has a value of 1/100,000,000th of a bitcoin. Regarding the source code of Bitcoin, only 21 million bitcoins will be supplied as mining rewards for miners opposite to the process that adds information to the blockchain. To date, around 70 percent of all bitcoins have been supplied, and approximately 90 percent of all bitcoins will have been supplied by 2026 (Kaplanov, 2012). Due to the use of a cryptographic proof and the distributed blockchain, verifying the authenticity and ownership

of a bitcoin as well as transferring a bitcoin's possession become possible without involving a trusted third party.

The Bitcoin protocol that is operated on a peer-to-peer network represents the third innovation in Bitcoin. Specifically, the Bitcoin protocol allows the main parts of Bitcoin to work together in a compatible way. Furthermore, the Bitcoin protocol allows payments to be handled without a fraud, which means a fully secure payment between two parties can be done without a mediator between those parties.

Unlike traditional currencies, virtual bitcoins are implied in transactions as Bitcoin maintains irreversible transactions by which bitcoins can be exchanged between users regardless of where the sender or receiver are located, same city or halfway around the world. Furthermore, Bitcoin gained a lot of media attention for being an anonymous digital currency as users in Bitcoin can hold multiple public addresses which are not linked to any personal information ([Androulaki et al., 2013](#)). Despite the wide adoption of Bitcoin, the identity of Nakamoto remains obscure and it is subject to speculation. For instance, it is not certain whether Bitcoin was created by a group of developers or the name Nakamoto is a pseudonym.

The distributed nature of Bitcoin as well as the easy trust conditions that are required among users, are the main key points behind the success of Bitcoin. Precisely, users can join Bitcoin for free and take part in the transaction verification process as well as the confirmation of transactions. In addition, Bitcoin introduces new payment strategies, such as micropayment, contract, and escrow transactions. These payment technologies were not available as traditional payments due it is high fees ([Karame et al., 2012](#)).

As Bitcoin goes beyond the scope of cash, the adoption of Bitcoin is growing into being an alternative to other fiat currencies such as dollars or pounds. In addition, the growth rates and circulation of Bitcoin do not follow any government or monetary policy, not similar to US dollars, for instance, that the Federal Reserve controls its growth rate. Therefore, Bitcoin does not consider any optimal rate of monetary growth. Instead, Bitcoin protocol is designed in a way that keeps the rate of bitcoin supply closes to zero by the year 2140, when the last bitcoin will be supplied as the limit of 21 million bitcoins will be reached ([Chuen, 2015](#)). However, a more Bitcoin adoption results in increasing the Bitcoin demand. With a constant supply of Bitcoin, increasing demand causes the Bitcoin price gradually increases over the long-term ([Brito et al., 2014](#)).

## 1.4 Overview of how Bitcoin works

A new user can get started with Bitcoin without deeply understanding the technical details. Bitcoin, from a user perspective, is just an application called Bitcoin wallet, Bitcoin core, etc,



that can be installed on either a mobile or a computer. By this application, users can get a personal wallet by which the first Bitcoin address can be generated in order to start trading with Bitcoin. Normally, there are several ways to obtain a bitcoin, such as buying them from a Bitcoin exchange or vending machines, or get them as a payment for a product or service.

As mentioned earlier, the role of a trusted third party is no longer needed in the Bitcoin system, whereas the availability of a trusted third party in digital currencies is deemed as an essential condition to launch payments. Consider a payment between *Alice* and *Bob* needs to be carried out through a normal digital payment system that requires a third party. When *Alice* sends 100\$ to *Bob*, the role of the third party starts here. The third party, e.g PayPal, that maintains a record of available balances, checks whether *Alice* has a sufficient balance that covers 100\$. The third party adds 100\$ to *Bob's* record if *Alice* has enough money in his account, otherwise the payment will be discarded by the third party. In Bitcoin, the third party is replaced by a publicly distributed record of all available balances. This record is approachable by everyone, so the responsibility of managing this ledger is shared over the entire network rather than relying on a central authority to manage this ledger. This brings the possibility of checking all accounts by any participant in Bitcoin. Specifically, Bitcoin uses a computational proof of work principle which is used by the decentralized network nodes to prevent any financial fraud through reaching a consensus on valid transactions which, on other hand, leads to build a distributed record of all available balances.

In the digital currency world, the double spending problem means that digital currencies can be spent twice. Clearly, consider a situation where digital money are represented by just a computer file, like a digital document. *Alice* can transfer 100\$ to *Bob* by simply sending a money file by e-mail. *Alice* can send a copy of the money file without deleting the original one. Within the absence of a central authority, *Alice* is still able to send another copy of the same money file to someone else. In Bitcoin, the distributed record solves the double spending problem without requiring a central authority. As mentioned before, participants of the Bitcoin network share a public ledger that includes all valid transactions that have ever been processed. Therefore, every user's computer can verify the validity of each transaction and check whether or not it has been spent before. Though, an attempt to double spend a transaction that has already been spent will be detected.

The Bitcoin user can send and receive bitcoins by using an address which is similar to a bank account. Each address consists of a public key and private key. Users of Bitcoin are distinguishable by their public key which is deemed as in opposition to their name or other identifiable information. Therefore, the public key is published and anyone can send bitcoins to it. In contrast, the private key should be kept secret as bitcoins that have been sent to a public key can only be spent by a user who possesses an appropriate private key. Users in

Bitcoin can hold multiple public addresses which are not linked to any personal information. Though, knowing a user from his public address is quite a hard task to achieve as this address is pseudonymous.

A bitcoin, that have been included in the blockchain, can be transferred between two wallets by a signed transaction. A transaction is signed by a private key that is kept in the Bitcoin wallet to provide a mathematical proof that the transaction has been sent by the wallet's owner. In addition, the private key prevents the transaction being replaced by other users once it has been issued. Once the transaction is created, it can be sent to a destination by broadcasting it in a peer-to-peer network which validates it against the sender's public key. In other words, the transaction is propagated to all of other nodes in the network if the balance of the sending address has sufficient amount of bitcoin. After incorporating the transaction into a block by a miner, the block will be propagated to all network nodes and then it will be incorporated in a public ledger, known as blockchain, by miners which usually takes 10 minutes. By including the transaction in the blockchain, a chronological order of the transaction will be enforced and all nodes of the network agree on the state of the system. However, updating a particular block is considered impossible as it requires modifying the entire previous blocks.

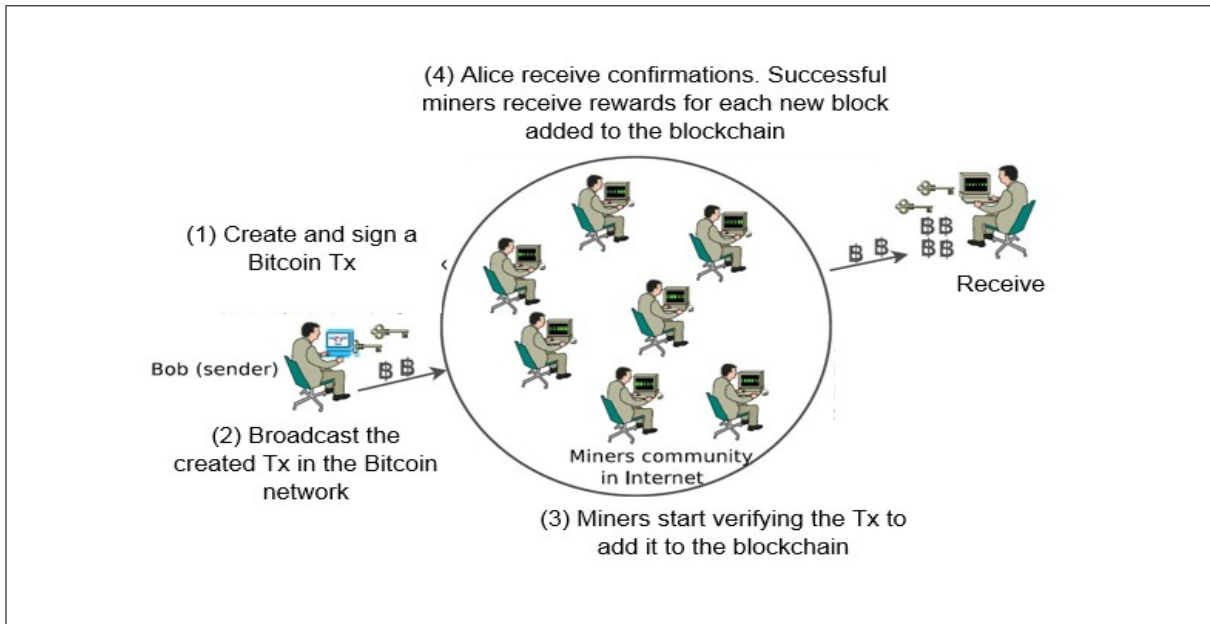


Fig. 1.2 Life cycle of transaction process

In order to give a better understanding of the overall methodology of payment in Bitcoin, consider the example in Fig. 1.2 where *Bob* wants to send 1 bitcoin to *Alice*. *Bob* is required to achieve two mandatory requirements, a Bitcoin wallet software that runs the Bitcoin's client installed in his phone/computer, and his private key as well as *Alice*'s public key. Then, *Bob* creates a transaction that implies 1 bitcoin and propagate it to the entire Bitcoin network. By

doing this, all miners in Bitcoin will be alerted in order to inform them about the new transaction. After that, miners start verifying the transaction (checking that Bob has sufficient funds) in order to add this transaction to the blockchain. Upon successful verification of the transaction, miners start racing to bundle Bob's transaction along with all the pending transactions in the Bitcoin network in order to confirm them in the blockchain. To do this, miners are required to solve a computational problem that need to spend some computational effort. Upon successful confirmation of transactions, both *Bob* and *Alice* will receive a confirmation about the successful transaction.

## 1.5 Distributed trust in the Bitcoin

As mentioned earlier, a distributed trust mechanism is achieved based on the publicly distributed ledger that is shared by the entire network nodes (Decker & Wattenhofer, 2013). This mechanism is considered as a monitoring technique by which the amount of the available bitcoins in the system will be tracked. To achieve this mechanism, two main requirements need to be fulfilled: (i) transactions verification process has to be achieved in a distributed manner to ensure the validity of transactions, and (ii) successfully processed transactions have to be quickly announced to everyone in order to guarantee the consistent state of the blockchain (Conti et al., 2017). As transactions are validated against the blockchain, reaching a consistent state of the blockchain is considered as a fundamental requirement towards achieving the distributed transactions verification process. Upon achieving the transaction verification process, a Bitcoin transaction has to be broadcasted to all nodes within the network in order to reach a consensus about which transactions are valid. Eventually, this consensus will be recorded in the blockchain. As the probability of reaching a global state of the blockchain is mainly affected by how quickly the Bitcoin information( transactions/blocks) are announced to everyone in the Bitcoin network, the main goal of the Bitcoin network is to propagate the information to the entire nodes as quickly as possible. Alas, a delay in information propagation is experienced during the transaction verification process which results in inconsistent blockchain. This makes Bitcoin vulnerable to some classes of attacks.

However, the focus of this work is on how to perform overall improvement in the information propagation delay in the Bitcoin network in a fashion that makes the blockchain more consistent.

## 1.6 Motivation

Transactions verification process must end up with a situation where all of nodes in the Bitcoin network agree to a common transactions history. However, transaction validation is

far from trivial due to the fact that the distribution of verified information(transaction/block) is delayed due to the network latency(Conti et al., 2017). Consequently, inconsistency in the replicas of the ledger that every node keeps are unavoidable. This results in a situation where the Bitcoin network nodes become uncertain about the validity of a given transaction. Furthermore, a desynchronized replicas of the ledger incentivises attackers to impose their own transaction history, possibly using the same bitcoins more than once. However, the latency of communications between nodes in the Bitcoin network is critical even though the probability of reaching an agreement about transactions history is high (Garay et al., 2015) (Miller & LaViola Jr, 2014).

As reaching a consensus in the Bitcoin network might require tens of minutes (Karame et al., 2012), a vendor accepts Bitcoin transactions and delivers products without waiting for the transactions' confirmations, a method known as fast payment. In this case, the probability of a double-spend is not negligible (Karame et al., 2012).

The main motivation for this work is to overcome the problem of achieving agreement on a common transactions log among nodes in the Bitcoin network through maintaining faster information propagation in the network. This work evaluates the impact of different network clustering approaches on improving the propagation delay in the Bitcoin network without compromising security. To date, most research investigations have focused on speeding up information propagation in the Bitcoin network by either modifying the scenario of how information are propagated in the network, or updating the Bitcoin network overlay structure. Previous attempts of updating the network topology structure have not taken into account any clustering approach. Instead, these attempts have considered either increasing the network connectivity by maintaining a mesh network topology, or relying on several coordinator nodes to support proximity of connectivity in the network without paying attention to security risks. Therefore, a gap in the research on considering the network clustering concept in speeding up information propagation in the Bitcoin network throughout presenting an entire network architecture that is based on the proximity of connectivity is evident.

## 1.7 Problem Statement

As highlighted above, the information propagation delay is a serious problem facing the consistency of the blockchain and several methods have been proposed in order to fix this problem (See chapter 2). As mentioned in (Decker & Wattenhofer, 2013), one of the causes of the propagation delay problem is the Bitcoin network topology layout where nodes are connected to each other randomly without taking into consideration any proximity criteria. This forces transactions and blocks to travel between nodes throughout long distance links.

Therefore, the crucial point that this thesis focuses on is to indicate whether or not the concept of the network clustering based proximity, where nodes is connected via short-distance links, is able to speed up information propagation in the Bitcoin network without compromising security. Furthermore, this thesis aims to identify how to evaluate the performance of different clustering approaches proposed in this thesis against the acceleration of information propagation in the Bitcoin network. Moreover, this thesis aims to indicate whether or not considering clustering approaches based on proximity in the Bitcoin network would have a security impact.

The main aim of this research is to determine ' *Can clustering in the Bitcoin network improve the information propagation delay without compromising security?*'.

## 1.8 Contributions

This section outlines the key area of contributions of this work as following:

1. The main contribution of this thesis is in examining the concept of clustering in the Bitcoin network to reduce average latencies of information delivery between peers. For any peer in the Bitcoin network, choosing other peers to connect with must be dependent on a particular proximity criteria, rather than handling random connections with other peers. This results in grouping peers according to their real-world proximity which would minimize the distance between any two nodes. In this thesis, examining the proximity based clustering concept involves proposing and evaluating four clustering approaches proposed in this thesis as follows:

A- Locality Based Clustering (LBC): The LBC protocol aims to increase the locality of connectivity in the Bitcoin network by supporting the geographical proximity based connections among nodes. Proximity of connectivity can be achieved in this protocol by referring an extra function for each node. By this function, each node is able to recommend other nodes that are close in the geographical distance to its' local neighbours. Clearly, each node runs the protocol independently by information about the discovered nodes and local neighbours. Each node can obtain this information by calculating the geographical distance between the discovered nodes and its local neighbours. Geographical groups can connect to each other by border nodes which are defined as a geographically closest pair of nodes that belong to two different clusters. In addition, a distributed mechanism is developed in this protocol which handles the entry and exit of the nodes in the network. The LBC protocol is different from the already existing approaches that are proposed in other classes of peer-to-peer network to group nodes based on geographical location. The LBC approach is achieved in a distributed manner without requiring any central point or a complete view of the network layout. Instead, all nodes

participate in the mechanism of suggesting closer nodes to their neighbours which gives no control for a certain node or centralised service over the network layout. This ensures the decentralised manner of the Bitcoin network which reflects a certain level of security awareness.

**B- Ping Time Based Clustering (BCBPT):** The key insight of the BCBPT protocol is to optimize the overlay topology by creating distinct, but connected clusters of peers with P2P latencies under a given intra-cluster threshold. Based on round trip ping latencies, nodes can detect and cut most of the inefficient and redundant logical links, and add closer nodes as its direct neighbour. Similar to the LBC protocol, the BCBPT protocol considers the distributed algorithm principle where each node can recommend closer nodes to its neighbours. The key difference between the LBC and the BCBPT protocol is that the recommendation of closer nodes is done based on the physical internet distance between two nodes in the network rather than the geographical distance.

**C- Bitcoin Clustering Based Super Node (BCBSN):** The BCBSN protocol is introduced as a way to combine the reduction of the intermediate hops between any two peers as well as increasing the proximity of connectivity in the Bitcoin network based on the geographical distance between peers. In the BCBSN protocol, each peer connects to one cluster head and each cluster head connects to other cluster heads. The BCBSN protocol is different from other super peers approaches that are proposed in different classes of the peer-to-peer network. The difference lies in the area of how to choose nodes to act as super peers. In the BCBSN protocol, super peers are elected by applying a proposed selection algorithm that forces every node intends to be as a super peer to handle a reputation protocol based on achieving certain requirements. These requirements ensure that impersonating the super peer role is challenging which, on other hand, increases the security awareness. In addition, a peer joining algorithm is proposed in this protocol which ensures that newly joined peers will be directed to the right cluster based on the geographical distance.

**D- Master Node Based Clustering (MNBC):** The MNBC incorporates master node technology and proximity-awareness into the existing Bitcoin protocol with the aim of creating fully connected clusters based on the physical Internet proximity. In the MNBC protocol, information can be exchanged between clusters via master nodes as well as normal nodes. In order to increase the security awareness, the MNBC protocol selects master nodes based on the same requirements of super peer selection in the BCBSN protocol. Physical Internet distance is used as a proximity metric in the MNBC protocol which is different from the BCBSN protocol where the proximity of connectivity is measured based on the geographical distance. Edge

nodes are proposed in the MNBC protocol to open more connection channels between clusters which is considered as another key difference between the MNBC and the BCBSN protocol.

2. As undertaking clustering in the Bitcoin network is different from clustering within other classes of the peer-to-peer network due to the strict requirements of security, this research examines whether clustering can be done safely without increasing the likelihood of certain classes of attacks, in particular, partitioning attacks. Though, the potential of partition attacks on the proposed protocols as well as the Bitcoin network is evaluated in this thesis. In addition, this thesis evaluated the performance of the proposed protocols based on the information dissemination speed in the network.

3. Measurements of the transaction propagation delay in the Bitcoin network are presented in this thesis. These measurements are collected using a novel methodology by which the transaction propagation delays are accurately measured as these delays are indicated when peers receive transactions. In addition, large scale measurements of the real Bitcoin network parameters that have a direct impact on a client behavior and information propagation in the real Bitcoin network, are performed in this thesis. These measurements are used to parameterise a model of the Bitcoin network which is developed in this thesis, while transaction propagation measurements are utilized to test whether or not the developed model is reflecting the virtual reality. To enable the evaluation of the proposed clustering protocols with respect to the performance and security point of view, several simulations are developed in this thesis.

## 1.9 Research methodology

This section introduces the research methodology of this thesis which considers the combination of two main research approaches, scientific method and computer simulation. In the following subsections, scientific method, computer simulation, and key research methodology stages will be explained.

### 1.9.1 Scientific Method

In order to find out a solution for problems or questions that raised within the science, scientific method was used (Peart, 2014). In this method, answers can be formulated in a form of theories or meta-theories. Fig.1.3 shows the Hypothetico-Deductive method where theories can pass through continuous flow of the scientific change and development (Dodig-Crnkovic, 2002). Hypothetico-Deductive method is adopted by scientist who usually re-examine theories

in order to achieve new-theories or offer a new context for old ones. From each iteration in Hypothetico-Deductive method, a hypothesis that is derived from existing theories and observations is predicted. After that, a theoretical test or practical experiment is conducted in order to be replaced within the boundary of a certain world view. One of the advantages of the scientific method is that it is unbiased, an argument raised by critics of the Hypothetical-Deductive method (Dodig-Crnkovic, 2002) (Grimes, 1990). This is due to fact that this method allows experiments to be repeated and results to be reproduced (Peart, 2014). This would offer a chance to re-validate or evaluate the obtained results. However, this work demands the hypothesis to be continuously tested by considering an iterative approach that ensures conducting incremental experiments.

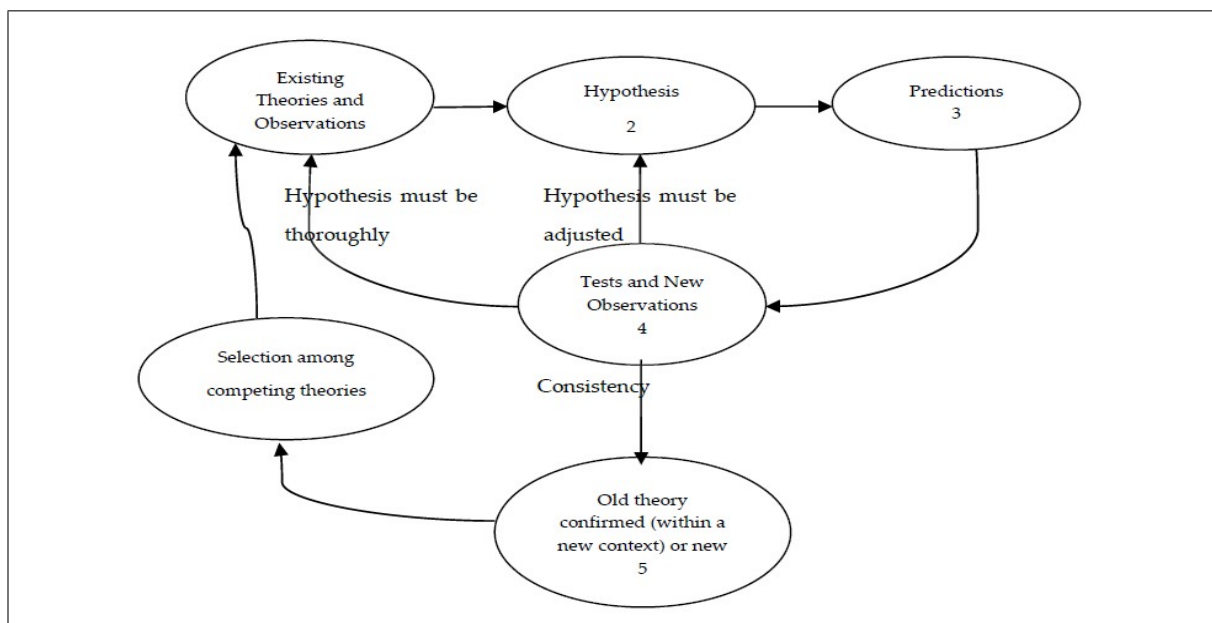


Fig. 1.3 The iterative nature of the Hypothetico-Deductive Method,(Dodig-Crnkovic, 2002)(Grimes, 1990)

## 1.9.2 Computer Simulation

Computer simulation is an approach that allows conducting controlled experiments that provide empirical data. Therefore, computer simulation offers an opportunity to observe parameters of a system phenomena within controlled conditions, enabling the evaluation of the whole system phenomena (Peart, 2014). Upon enabling a phenomena to be modelled, important aspects and features of the phenomena need to be identified and analyzed. Once a model is built, predicting consequences of considering some changes in a particular system becomes possible throughout several controlled experiments using the developed model. The reliability of experiments and validity of the obtained data are mainly based on the 'quality' of the developed model.



Therefore, building any model of the real world should be evidenced that the model is reflecting the reality. To ensure this, the developed model needs to be validated against prior models or the real world using 'a model emulating'. The model emulating is provided by the computer simulation to test the performance of the developed models through conducting experiments of the real-world (Law et al., 1991).

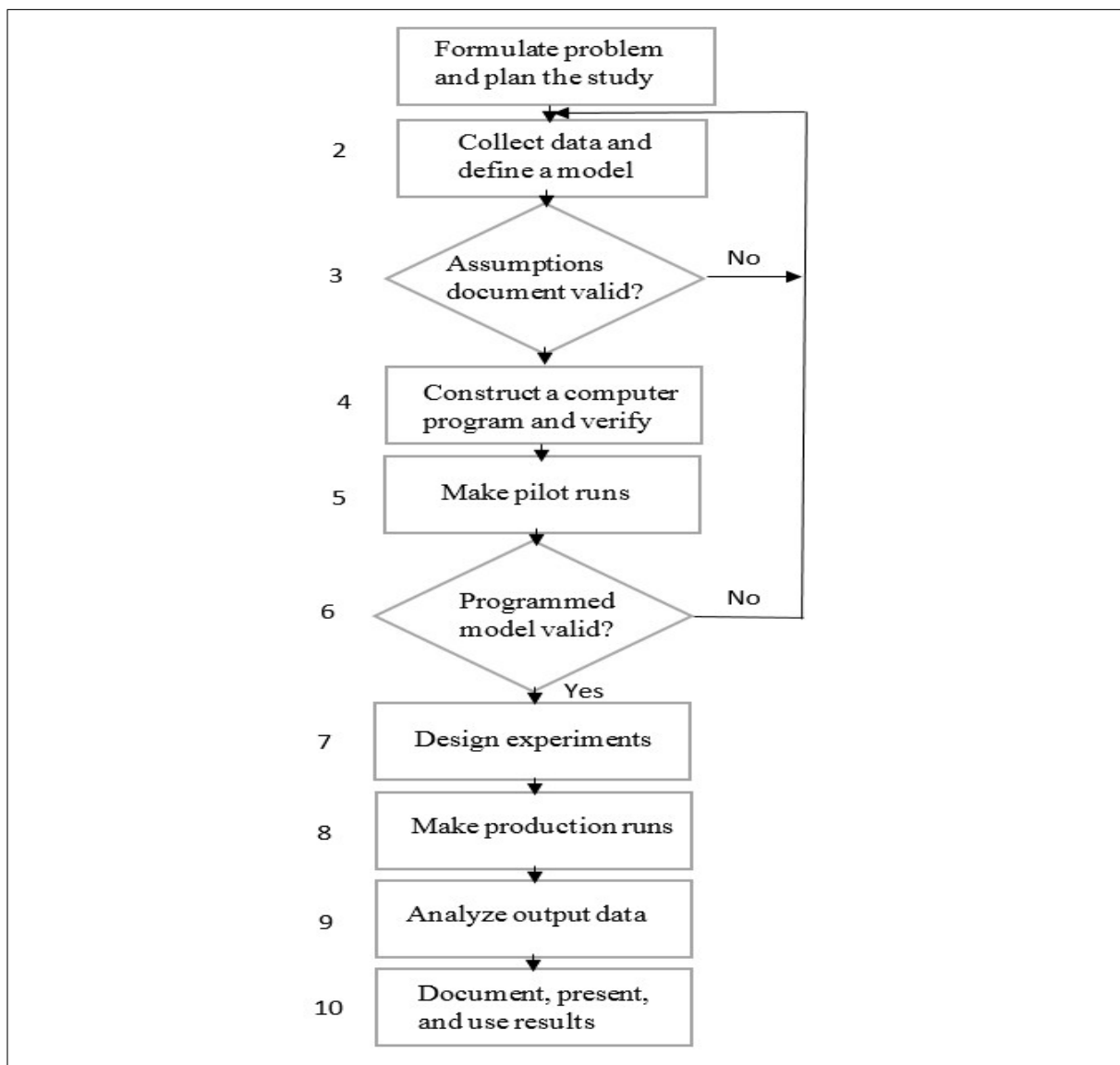


Fig. 1.4 Steps in a discrete simulation study (Law, 2003),(Law et al., 1991), (Bar-Noy et al., 1995), (Peart, 2014)

Within computer science research, computer simulation plays an important role towards supporting both theoretical and experimental methods. In addition, computer simulation is quite similar to Hypothetico-Deductive method with respect to providing an iterative research

approach by extending the investigations to study a phenomena. Moreover, simulation method can be used to model the complexity of non-linear real world systems in such a way that reflects the virtual reality (Ringland, 2010). However, there are three types of simulation methodology which are, discrete events; system dynamics; and agent-based simulations (Borshchev & Filippov, 2004).

This work uses event based simulation, where defined systems are modelled as an ordered sequence of well-defined events that comprises a specific change in the system's state. The reason behind adopting discrete event rather than other simulation approaches is that the discrete event simulation can model activities that happen in an interval of time using multiple events. This would provide a chance to utilize computational advantages that the discrete event offers over a continuous dynamics simulation. An iterative approach that is adopted in a discrete-event simulation (Law et al., 1991)(Bar-Noy et al., 1995), is illustrated in Fig.1.4. This approach starts with observing and investigating the problem as well as identifying plans along with objectives. In the next stage, the required data derived from the existing system, is collected and utilized to define a model. In the following stage, the collected data is used to form assumptions. Then, the required model is built (step 5) and verified (step 6). In step 6, if the model is valid, then the model is used to design the required experiments (step 7) which will be run (step 8) to produce results, which will be analyzed (step 9) before being documented (step 10). On the other hand, if the model is not valid (step 6), then the method iterates back to step 2 where data can be re-reviewed and corrected before conducting the the stages of the method again.

### 1.9.3 Key research methodology stages

This work follows a methodology that interleaves the hypothetico-deductive model and the discrete simulation study methodology. The key stages of the methodology that are conducted in this research are illustrated in Fig.1.5. According to the conducted research methodology, the literature that focuses on information propagation in the Bitcoin network is deeply reviewed in order to support and prove the hypothesis which considers the network clustering concept as a way to speed up information propagation. Also, the literature is analysed to figure out the main influential entities that need to be modelled. After identifying and collecting parameters that have a direct impact on a Bitcoin client behavior and information propagation in the real Bitcoin network such as number of the reachable nodes, link latencies between peers, and the peer's session lengths, a core simulation model can be built for the purpose of experiments setup. The collected data is attached to the developed model in order to simulate the reality. Before conducting any experiments, the developed model must be validated against the real-world system. This can be done by comparing the results that are produced from the developed simulator to the real system results. Specifically, the validation of the simulation model is done

based on measurements of the transaction propagation delay. Therefore, establishing the range of propagation delays within the real Bitcoin network is conducted as a fundamental stage in this methodology.

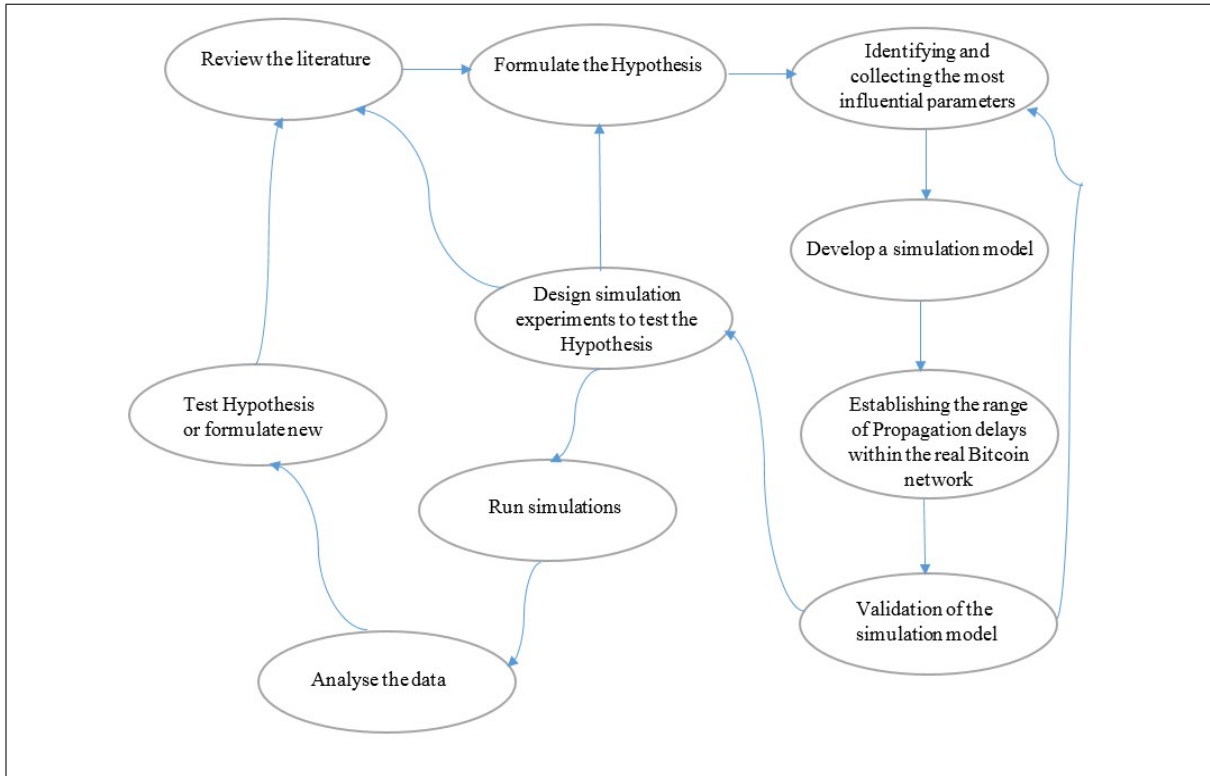


Fig. 1.5 depicts a conceptual model of key research methodology stages

Once the simulation model is validated, the experiments can be designed to test the hypothesis. To ensure that the experiments' output is not corrupted, an iterative incremental approach will be considered where every single variable will be tested at each iteration. However, simulations are conducted for a number of times to ensure reliable results. Finally, the produced results will be evaluated in order to validate the hypothesis.

## 1.10 Thesis Outline

This thesis consists of six main chapters namely, introduction, literature review, proposed clustering approaches, Bitcoin measurements and simulation model, performance and security evaluation, and conclusion. The rest of the thesis is organized as follows:

Chapter 2 gives an overview of the key components Bitcoin. In addition, this chapter critically analyses double spending attacks in the Bitcoin network and how blockchain forks offer a

chance to perform this kind of attack. In addition, Chapter 2 also discusses information propagation in the Bitcoin network and how propagation delay may result in inconsistency in the public ledger. Furthermore, Chapter 2 critically reviews some existing methods and techniques in relation to the acceleration of information propagation in the Bitcoin network as well as overcoming double spending attacks. Moreover, existing methods and techniques with respect to the mitigation of propagation delay in other peer-to-peer networks are highlighted in this chapter.

Chapter 3 introduces four clustering approaches with the aim of speeding up information propagation in the Bitcoin network. These approaches are based on improving the proximity of connectivity in the Bitcoin network.

Chapter 4 introduces large-scale measurements of the most influential parameters of the Bitcoin network. In addition, a novel methodology that measures the transaction propagation delay in the Bitcoin network is introduced in this chapter. Furthermore, a simulation model of the Bitcoin network is presented in this chapter.

Chapter 5 presents performance and security evaluations in relation to the proposed clustering approaches. Performance evaluation is based on the transaction propagation delay, while the security evaluation is based on the difficulty of performing partition attacks.

Chapter 6 describes the contributions of this thesis and how research questions are addressed. Furthermore, future work is also highlighted towards further improvement in this research area.

# Chapter 2

## Literature Review

As discussed in Chapter 1, this thesis focuses on the evaluation of clustering as a mechanism to improve the information propagation delay in the Bitcoin network without compromising security. This chapter critically discusses some existing methods and techniques in relation to the acceleration of information propagation in the Bitcoin network including minimize verification, pipelining information propagation, and connectivity increase. This chapter also discusses how the propagation delay problem was addressed in other peer-to-peer networks. Furthermore, prior theories that focus on analysing and mitigating double spending attacks will be highlighted in this chapter.

This chapter also includes the description of information propagation in the Bitcoin network and how it affects the consistency of the blockchain. Furthermore, a background about different components of Bitcoin will be given in this chapter.

### 2.1 Background

As it is mentioned earlier, Bitcoin is the name of currency that Bitcoin enables, one bitcoin (BTC) has an equivalent value in British pounds, US dollar. Depending on the context, Bitcoin is the name that is used as an indication for an abstracted protocol that has been introduced by Satoshi Nakamoto ([Nakamoto, 2008](#)). In addition, Bitcoin also refers to a reference implementation, known as *bitcoind*, which is the most common and used Bitcoin client that is written as a proof of concept implementation. As shown in the Fig.2.1, Bitcoin consists of several main components that support and handle the Bitcoin protocol. These components include Bitcoin P2P network, Bitcoin protocol, Bitcoin user, transactions and blocks, consensus blockchain, and miners who maintain the blockchain.

The role of users in Bitcoin includes sending and receiving payments throughout establishing wallets that represent the Bitcoin client ([Androulaki et al., 2013](#)). By getting the Bitcoin

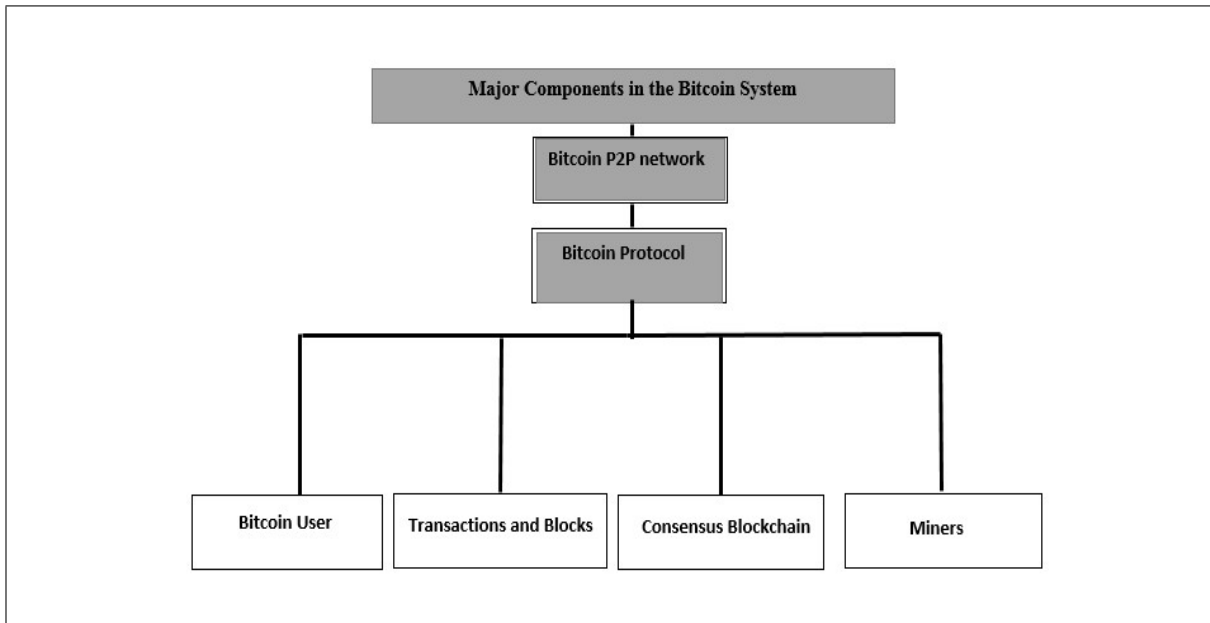


Fig. 2.1 Major components of the Bitcoin payment system

wallet installed (e.g. Bitcoin-Qt), a user becomes an actual participant in Bitcoin. On the other hand, transactions and blocks help in achieving the Bitcoin protocol. Transactions transfer coins between two destinations, whereas blocks contribute in building the ledger that considers transactions' chronological order (Heilman et al., 2015). This ledger stores the entire networks' transactions history which reflects the available bitcoins in the Bitcoin system. Though, when a user claims a bitcoin, this means that he has got a valid transaction with a balance that covers the claimed amount confirmed in the blockchain. Furthermore, maintaining the blockchain is a competitive task that is fulfilled by a group of nodes named as miners (Garay et al., 2015). These miners normally get paid for doing this task. However, Bitcoin protocol is considered as a secure environment in which all the aforementioned components interact and work with each other to achieve the main goal of Bitcoin. For more clarification purpose, major components of Bitcoin will be critically explained in the following sections.

### 2.1.1 The Bitcoin Network

The Bitcoin network is simply a peer-to-peer network by which Bitcoin users from around the globe are able to interact with each other and exchange the currency, so this contributes in making Bitcoin a global payment system (Ron & Shamir, 2013). By using the Bitcoin network, the Bitcoin protocol is able to establish the global state of the Bitcoin system which is achieved by propagating transactions and blocks through the network. In the following subsection, a brief background in relation to peer-to-peer networks will be given before moving on to discuss

the Bitcoin network structure, Bitcoin network discovery, and DNS seed node in the Bitcoin network.

### 2.1.1.1 The Peer-to-peer Networks

The term peer-to-peer network (P2P) refers to a network of computers(nodes) that are configured to allow each node to peer to other nodes. P2P networks are considered as a decentralized environment which has no controlling authority or central server. Every node in the P2P network can act as a server and a client at the same time, which means that every node can send and receive information within the network. The early Internet in which nodes on the IP network were equal, is considered as the best example of a P2P network architecture ([Herley, 2008](#)).

Basically, the P2P networks follow a dynamic reorganization of peer members where every peer can join a P2P network at anytime or at the same time where old peers leave the network ([Schollmeier, 2001](#)). Furthermore, peers can simultaneously download files and collaborate by sharing the network resources (e.g. file sharing). Another feature of P2P networks is its resistance against fault-tolerance when a resistant routing protocol is adopted. Specifically, P2P networks can still work effectively even though a peer is disconnected or goes down. In Bittorrent system, for instance, a file can be downloaded by clients which are classified as servers ([Qiu & Srikant, 2004](#)). When a certain server does not respond to a message request from a client, the client can contact other servers in the network and pick up parts of the file. This is the opposite to a client-server model with respect to fault tolerance as all network communications stop if the server goes down.

P2P networks are divided into two types, pure P2P network, and hybrid P2P networks ([Herley, 2008](#)). In pure P2P networks, all peers are equal with respect to the assigned functions. Peers can interact and collaborate with each other without any central point that acts as a coordinator. While in the hybrid P2P networks, a central organizer must be applied to coordinate the connections and interactions among peers, such as Napster network, and BitTorrent network. Regarding fault tolerance, pure P2P networks have a higher level of fault tolerance compared to hybrid network.

On the other hand, pure P2P networks consume more network resources, the main reason why pure P2P networks is less scalable than the hybrid P2P networks. For instance, searching for files or data item in file sharing systems which represents a pure P2P network, requires exploring the whole network throughout flooding the network with request messages. Therefore, every peer receives a request message should check the request against the locally known data items and forward it to its neighbours. This results in an overhead as massive duplicated request messages would be available at each peer. However, there is a completely different relay theory

that has been followed in the Bitcoin network. Rather than searching for a specific file or data items, Bitcoin network's main target is to distribute information as fast as possible to reach consensus on the blockchain (Tschorsch & Scheuermann, 2016).

### 2.1.1.2 Bitcoin network structure

Bitcoin network refers to a group of nodes that handle the Bitcoin protocol. As it is already mentioned, Bitcoin is built on a decentralised structure which is considered as one of the key features of Bitcoin. Though, there is no centralised server that the Bitcoin architecture relies on. Instead, a distributed protocol has been maintained to support the system (Donet et al., 2014).

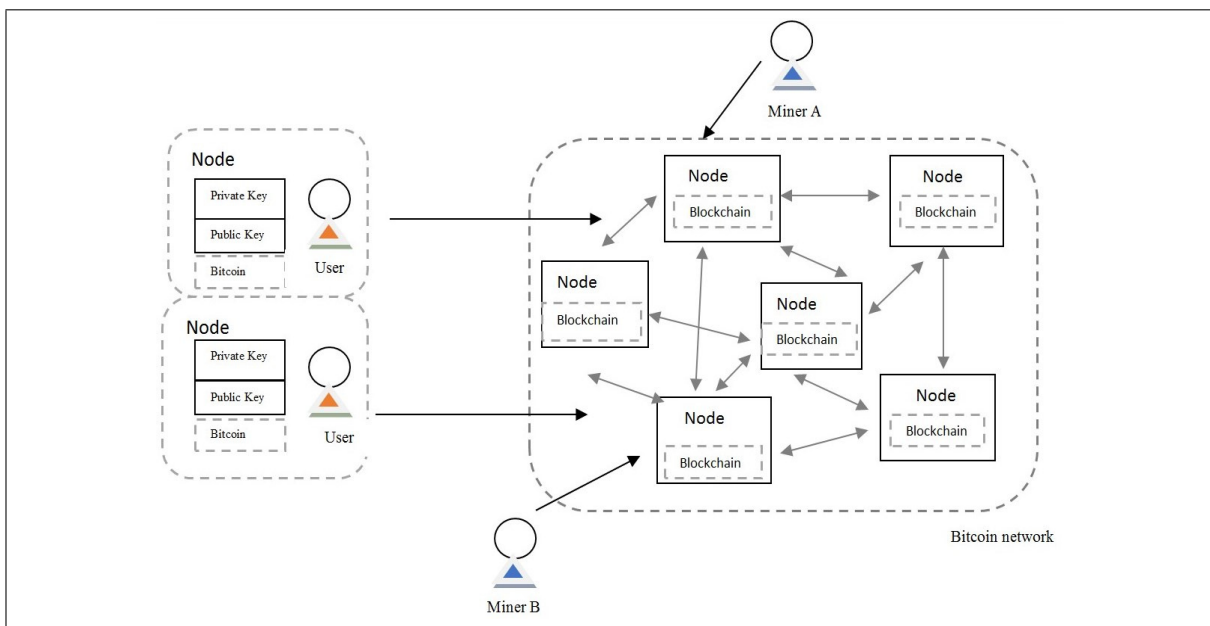


Fig. 2.2 Bitcoin network

In this network, as shown in Fig.2.2, each peer runs the Bitcoin protocol and connects with other peers over a TCP channel (Biryukov et al., 2014). As the Bitcoin network topology is not proximity defined, connecting to other peers is maintained randomly. In addition, every node should maintain a maximum of 8 outgoing connections to peers and accepts up to 117 connections (Heilman et al., 2015). Nodes can join and leave the network at any time they want. When a node re-joins, it asks other nodes for new blocks to be able to complete its local copy of the blockchain (Kondor et al., 2014). For the purpose of making denial of service impractical, just the valid information (transactions and blocks) are propagated, whereas invalid transactions and blocks are discarded. Bitcoin network nodes are classified into two groups, servers which can accept incoming connections and those which can not (clients), because they are behind NAT or firewall (Feld et al., 2014).



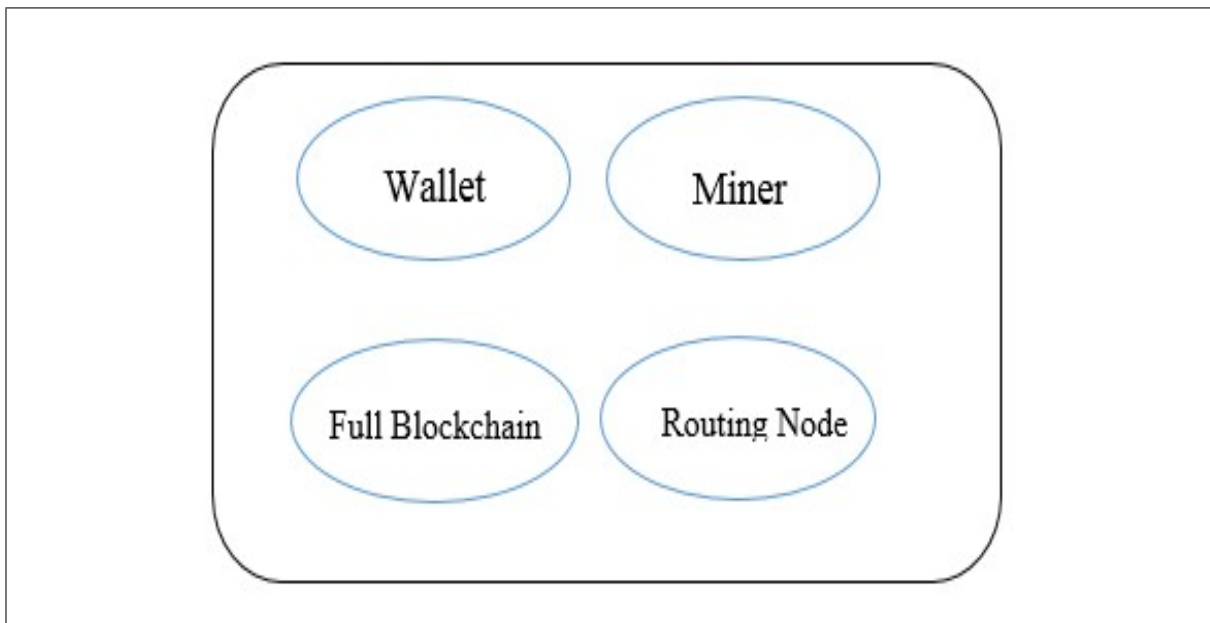


Fig. 2.3 A Bitcoin network node with all four functions: Wallet, Miner, full Block- chain database, and Network routing

As shown in Fig.2.3, the Bitcoin nodes take different roles in the network based on the functionality that those nodes support such as wallet services, routing, etc. As Bitcoin relies on distributed validation, an essential role in which transactions are validated in a distributed manner, is covered by all nodes in the network ([Heilman et al., 2015](#)). In order to participate in the Bitcoin network, all nodes have to do the routing function. This function includes validate and propagate transactions, and maintaining connections to other nodes. Nodes that maintain a full up-to-date copy of the public ledger(blockchain) are known as full nodes ([Antonopoulos, 2014](#)). Those nodes are able to verify any transaction without requiring an external reference. Whereas, SPV nodes that maintain only part rather than the full version of blockchain, can verify transactions using a method named Simplified Payment Verification (SPV). SPV is simply a protocol that follows a light transaction verification process for transactions which is sent from a user ([Kiayias et al., 2016](#)). SPV is vulnerable to some classes of attacks, such as the sybil attack and the double spending attack ([Okupski, 2014](#)).

Nodes that attempt to solve the proof of work problem by running specialized equipment, are known as mining nodes. Full nodes can be classified as mining nodes as they maintain a full version of the blockchain, whereas SPV nodes need to participate in a mining pool in order to work as a miner. The mining pool consists of a set of miners that agreed to divide the mining reward of blocks that found by a miner in the pool. Mining pool relies on a server pool that acts as a full node ([Schrijvers et al., 2016](#)). Although a full node is represented by a user wallet who

use the desktop Bitcoin client(desktop wallet), many users wallet nowadays are considered as SPV nodes which suite the resource-constrained devices such as smart phones.

### 2.1.1.3 Bitcoin network discovery

When a node  $N$  joins the Bitcoin network for first time, a discovery mechanism that does not consider any proximity criteria, is adopted to find other nodes in the network. As a first step, at least one existing Bitcoin node needs to be discovered by the node  $N$  in order to discover more nodes ([Antonopoulos, 2014](#)). After that, more connections will be established between the node  $N$  and the nodes that are discovered. Establishing connections to other nodes is done without taking into account any proximity priority as the Bitcoin network topology is not proximity defined ([Decker & Wattenhofer, 2013](#)) ([Stathakopoulou et al., 2015](#)).

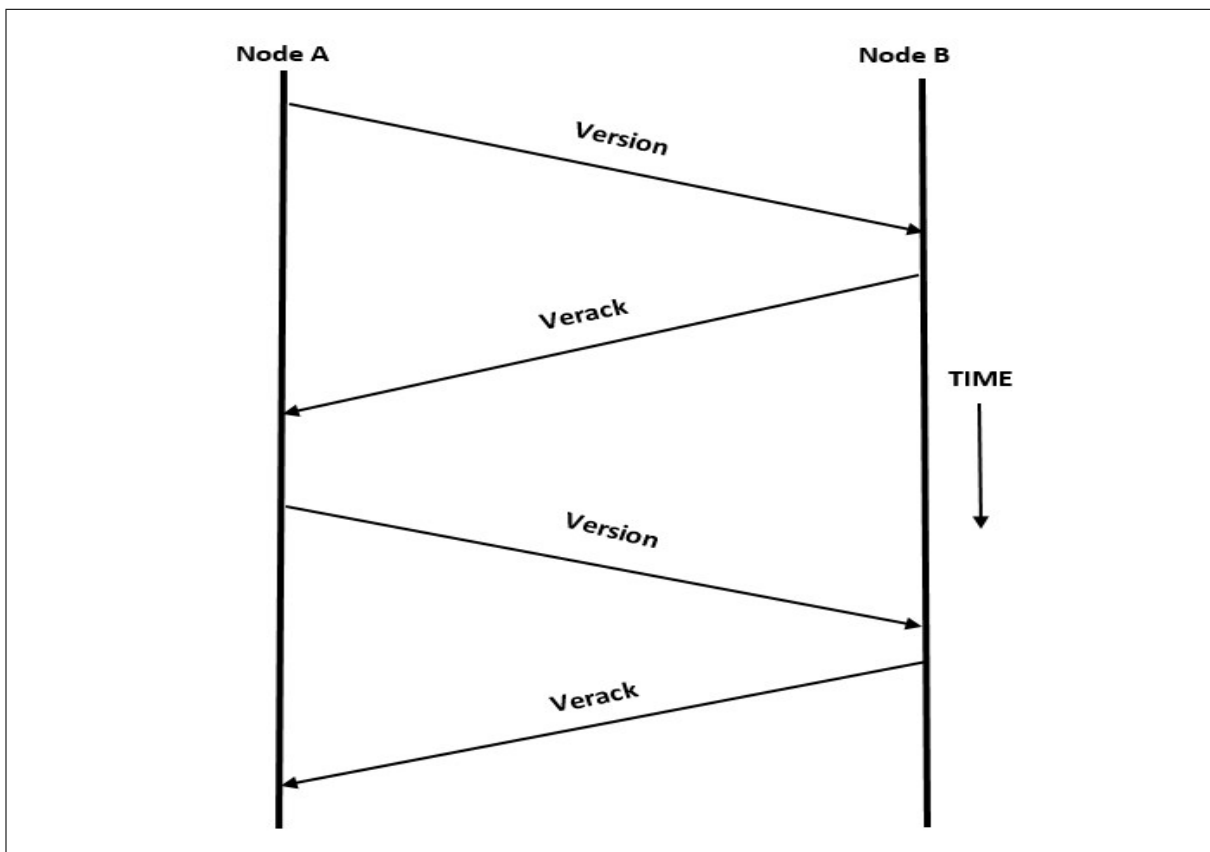


Fig. 2.4 Initial Handshake mechanism between Nodes A and B

To establish a TCP connection, as shown in Fig.2.4, a handshake with a known peer is handled by sending a *version message* which contains basic identifying information. These information include:

-Protocol-version which is a constant number that refers to the Bitcoin protocol such as 70002.

-nTime: refers to the current time.

-addrYou: indicates the IP address that belongs to the remote node.

-addrMe: indicates the IP address that belongs to the local node.

-BestHeight: points out the current block height of the node's blockchain.

-subver: The type of software that is running in this node is indicated by this field such as /Satoshi:0.9.2.1/ .

A peer responds back to the version message by sending a *verack message*. Each peer holds a list of IPs of peers that are connected to it. To stop peers misbehaving, each node handles a penalty score mechanism for each node connected to it. The score is increased when an unreliable behavior is announced. When the score reaches 100, the misbehaving IP is banned by the node that handles the penalty score. Furthermore, a transactions pool is maintained by each node which includes transactions that wait to be verified and relayed to the neighboring nodes (Biryukov et al., 2014).

On the question of how a new node discovers the first node in the network, there are some stable nodes that behave as seed nodes listed in the Bitcoin client that could suggest to the new node some other nodes in the network (Heilman et al., 2015). Specifically, bootstrapping that needs to be handled by the new node, requires at least one IP address of a Bitcoin network node which is known as *DNS seed node*. After maintaining a connection to the seed node, further introductions to other nodes will be handled. Then, more connections to other nodes will be established and the new node will disconnect from the seed node. However, connecting to other nodes would help the new joining node in discovering more nodes. This can be done through sending an *Addr message* which includes the IP address of the sender node. Precisely, the newly connected node can advertise its own IP to other nodes by sending an *Addr message* to its neighbours. This helps the new node to be found by other nodes. On the other hand, the new node can get to know other nodes by sending a *Getaddr* message to its neighbours. As illustrated in Fig.2.5, neighbours respond to the *Getaddr* message by sending a list of IP addresses of other nodes in the network (Antonopoulos, 2014).

However, *Addr message* is continuously initialized by a node that sends its IP address to each node of its connected nodes (Heilman et al., 2015). In addition, *Addr message* is also sent when a node receives an *Add message* with no more than 10 addresses. In this case, the node forwards this *Add message* to two peers of its connections. These peers are selected by taking the hash of each connected peer's IP as well as a nonce that associated with the day. Based on the lexicographical order of hash values, the node selects peers that have the first and second hash values. To stop propagating *Addr message* at a certain point, each node maintains a list

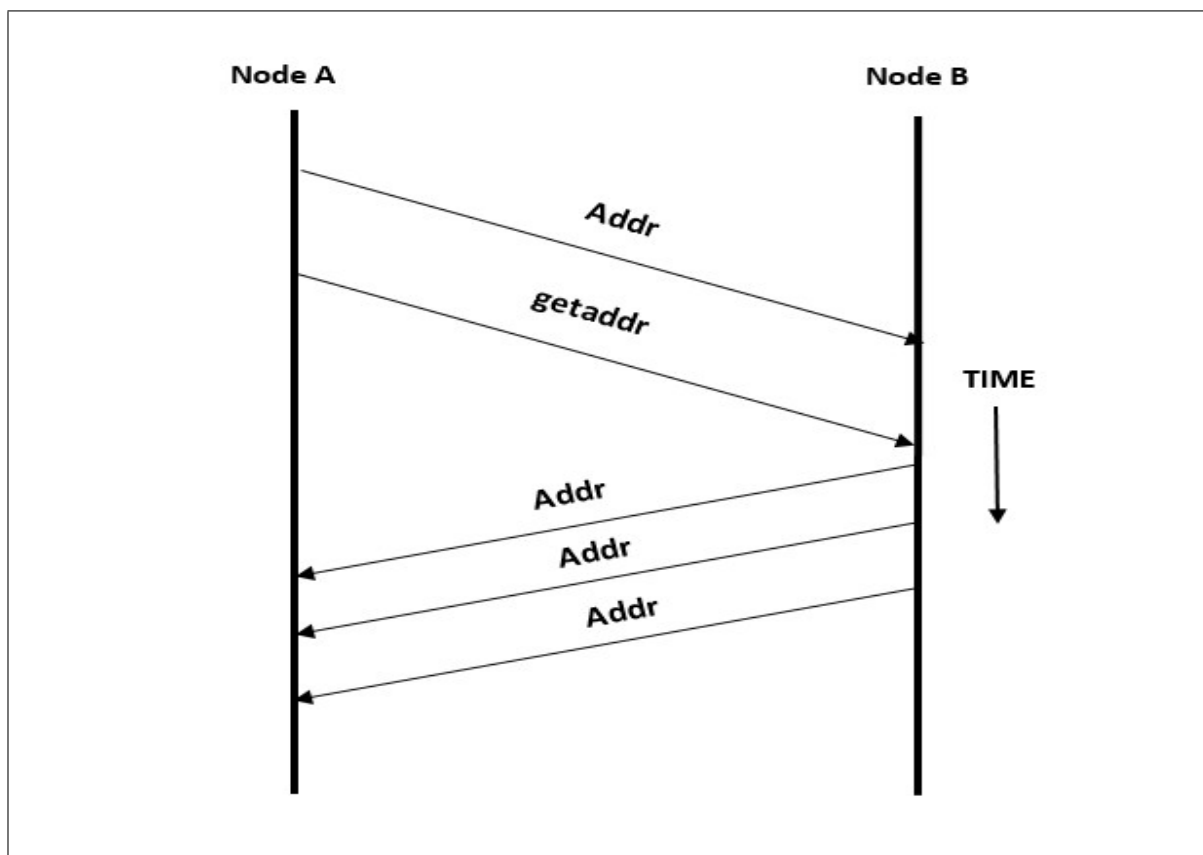


Fig. 2.5 Dissemination of *Addr* message between two peers

of known IPs that have been sent to or learnt from each peer of its connected peers, so no IP address is sent from the known list. In addition, flushing on a daily bases is done in the known IPs list.

Even though each node establishes connections to other nodes, the node should continue discovering more nodes and advertise its existence to the newly joined nodes ([Biryukov et al., 2014](#)). This is due to unreliable paths as nodes come and go in the network in a random way. Therefore, a node that connects to other nodes does not guarantee that these connections will not be lost. However, discovering other nodes continues to operate with the aim of offering diverse paths into the Bitcoin network. When a node reboots, it can re-join the network without needing to bootstrap the network again as the node can still remember most recent successful nodes connections, so the node tries to reestablish connections to those nodes by sending connection requests. While there are no responses for these requests, the node starts bootstrapping the network again. In terms of a connection has no traffic goes through for more than 90 minutes, the connection will be dropped off ([Antonopoulos, 2014](#)).

#### 2.1.1.4 DNS seed nodes in the Bitcoin network

As it is mentioned earlier, A Bitcoin DNS seeder is a server that assists nodes to discover active peers in the Bitcoin network. Therefore, the DNS seeder responds to the DNS query by initiating a message that contains a list of IPs. The maximum number of IPs that can be attached to the message is limited by constraints on DNS, around 4000 messages can be returned by a single DNS query (Heilman et al., 2015). In the Bitcoin network, there are six DNS seeds that periodically crawl the entire network in order to obtain active IP addresses. However, there are two scenarios where DNS seeders are queried by other nodes. The first scenario happens when a node that joins the network for the first time, and tries to connect to the active IPs. While in the second scenario, the DNS seeder is queried by a node that restarts and attempts to reconnect to new peers. In this case, the DNS query is initialized after 11 seconds since the node attempted to reconnect and has less than two outgoing connections (Heilman et al., 2015). However, DNS service can be considered as a form of centralization in the Bitcoin network.

### 2.1.2 Bitcoin protocol

As it is mentioned earlier, the Bitcoin's goal is to achieve a peer-to-peer electronic payment system that cut out the role of the third party, which means a fully secure payment between two parties can be done without a mediator between those parties (Moore & Christin, 2013). Due to a problem, known as *Byzantine Generals problem*, achieving payments in Bitcoin without a security issue is far from trivial (Buchman, 2016). Basically, the *Byzantine Generals problem* focuses on the scenario of how two remote entities in a network could send messages between each other while also knowing that those messages are authentic and not manipulated in some way against rules (Lamport et al., 1982) (Hsieh & Chiang, 2014). In other words, in the sense of large networks how it is guaranteed that everyone in that network is working towards the right goal. Reflecting the Byzantine Generals problem on P2P cash systems, users of these systems can easily double spend transactions, or attempt to deny service from another user or a type of transaction. To this extent, the Bitcoin protocol is proposed in Bitcoin to overcome the *Byzantine Generals problem* by using a very complex transaction verification process along with digital signatures (Pease et al., 1980). The Bitcoin protocol also uses an ongoing ledger that is a hash based proof-of-works ledger which can not be changed without re-doing that works. In other words, the Bitcoin protocol is based on a distributed public ledger which records all bitcoins in the network. Each entry in this record is considered as a transaction by which the virtual currency is transferred. In order to have a complete overview over the main aspects of the Bitcoin protocol, several components of Bitcoin that have a direct impact on the Bitcoin protocol will be described in details in the following subsections.

### 2.1.2.1 Transactions

Transactions are considered as one of the main entities that the Bitcoin protocol relies on. For bitcoins value to be transferred from one or more source to one or many destinations, transactions take the responsibility of achieving this goal. Specifically, transactions are created by a Bitcoin user who intends to send a specific amount of bitcoins to one or more destination accounts (Decker, 2016). A distention account can spend the transferred bitcoins by giving an authorization to another user, and so on. Each transaction includes input which represents debit against a Bitcoin account, and output which represents credit added to a Bitcoin account (Antonopoulos, 2014). To combine or split bitcoins, as shown in Fig.2.6, a Bitcoin transaction can handle multiple inputs and outputs. Outputs represent the transferred bitcoins whereas inputs reference funds from other previous transactions. The sum of all outputs should be equal to the sum of all inputs. In some cases the amount of bitcoin in the output might be less than the amount of bitcoin in the input as there is a bitcoin fee which is dedicated from the output as a reward for a miner who confirmed the transaction in the public ledger. The new owner of the transferred bitcoins is determined by a transaction output which creates a new output/outputs when it is referenced as an input in a future transaction. Valid transactions should be signed with a private key which is associated with a spending account. This signature is an Elliptic Curve Digital Signature Algorithm (ECDSA) which is a digital signature scheme based on public key cryptosystem ECC that is considered as a proof of ownership which can be independently verified by other users (Wang, 2014). The balance of an account is the sum of all unspent outputs of the account which can be tracked by the public ledger. A transaction output consists of a numerical value of bitcoin as well as a condition to spend that output which known as “encumbrance”. Outputs are fundamental information that manage the bitcoin spending in such a way that prevents any accounts clashes or double spend the same amount (Biryukov et al., 2014). Precisely, only previously unspent outputs can be considered in the input of a follow-up transaction. As unspent outputs are tracked in the ledger, outputs are verified against the public ledger which needs to be consistent.

Each transaction is known by a hash of its serialised representation. For a transaction to be valid, several criteria have to be fulfilled. Firstly, an output is spent only once. Secondly, the sum of the newly allocated outputs should be less or equal to the sum of the claimed outputs (Heilman et al., 2015).

### 2.1.2.2 Blocks

An agreement has to be achieved among nodes in the Bitcoin network regarding a common order over transactions with the aim of keeping the ledger replicas consistent. In distributed

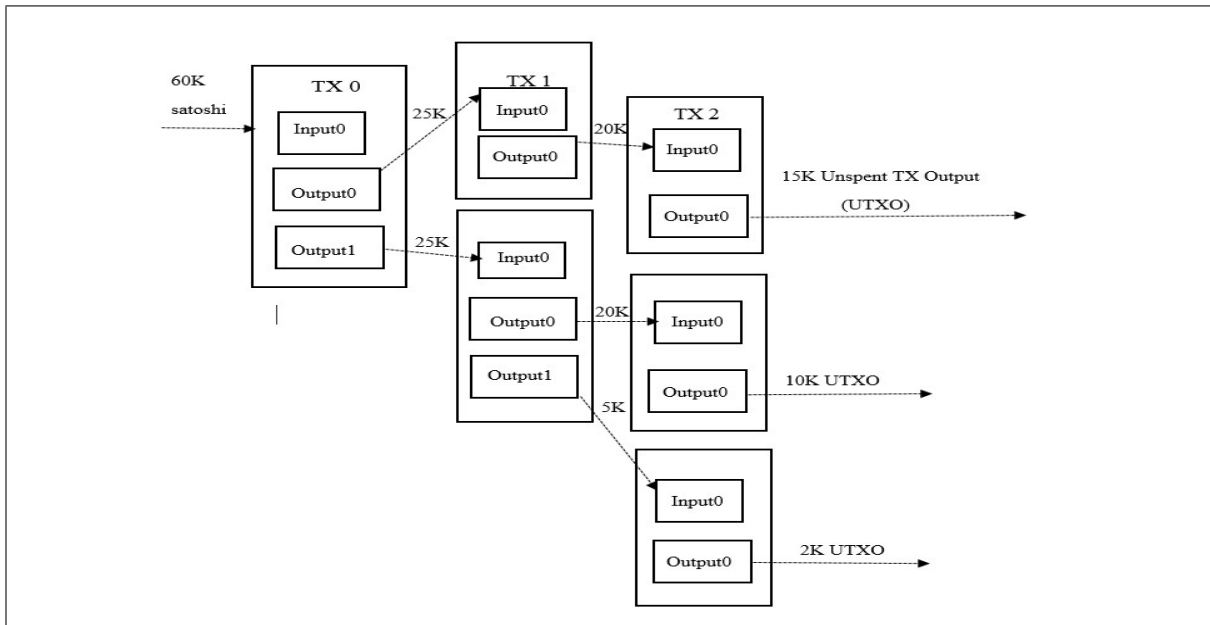


Fig. 2.6 Bitcoin transaction

systems, reaching a consensus on a common order of transactions is not trivial. Therefore, valid transactions should be committed and synchronized by broadcasting a block that forms part of the ledger, in order to keep the chronological order of the valid transactions across nodes (Ober et al., 2013).

Once a node commits a set of transactions  $T_k$ ,  $T_k$  is included in a block  $k$  which is created by the same node. Then, the block  $k$  is disseminated to the entire nodes in the network. On receiving the block  $k$  by each node, it checks the validity of the block  $k$  in order to add it to the blockchain. In addition, nodes verify all transactions inside the block  $k$ . This includes checking if the node that created the block  $k$  sent himself the correct reward (currently 25 BTC). In case of the block  $k$  passed through the validity checks at a particular node, the node rolls back the tentatively committed transactions since the last block and consider applying transactions from the received block  $k$ . In this sense, an agreement among all nodes on the validity of all transactions that are included in the block  $k$  is achieved. In other words, transactions that were committed and included in a block are considered to be confirmed and not possible to be re-applied (Decker, 2016). On the other hand, transactions that conflict with other transactions which are included in a block are discarded as they are deemed to be invalid (Koshy et al., 2014).

The view of the changes achieved by the block creator that implies acknowledging a group of transactions in the block, can be effectively reflected and imposed since the previous block. However, these changes are limited and transactions cannot be forged by the block creator unless the underlying public/private key cryptosystem is not secure (Tschorsch & Scheuermann,

2016). The block creator can only be able to control the arrived transactions' order and which transactions should be included in its block. In order to determine which node may impose its view and acknowledge a group of transactions in a block, nodes spend a computational effort attempting to find a solution to a proof-of-work (Natoli & Gramoli, 2016). The solution is based on achieving a byte string of meaningless data, called nonce, which is combined with the block header. This combination results in SHA-256 hash of the block that have a certain number of leading zero-bits, or target (Barber et al., 2012). Target is a 256-bit number (extremely large) that all Bitcoin clients share. As hashes are one-way functions, finding the right nonce is not trivial. Calculating the block hash for all possible nonces is considered the only possible way to get a good nonce. However, finding out the input of the solution is a difficult task to achieve. On the other hand, verifying the solution is a straightforward task to fulfill. Based on the nonce that is part of the block, nodes that receive the solution can easily check whether or not the block creator solved the proof-of-work. In order to identify the target, consensus by all nodes should be achieved which allows an average of one result every 10 minutes in the entire network. Furthermore, the target is adjusted every 2016 blocks, approximately every 14 days (Conti et al., 2017). The block is accepted by the network if the hash of a block's header is lower than or equal to the current target. A lower target, a more difficult to generate a block.

As an incentive, a reward in a form of newly minted bitcoins is given for a node that finds a solution for the proof of work problem, known as a miner (Babaioff et al., 2012) (Sompolinsky & Zohar, 2013). This reward can be assigned by including a transaction, which has no inputs but may specify outputs for a predetermined number of bitcoins, into a block. The rewarding transaction is considered valid when it is included in a block as well as the sum of outputs is smaller or equal to the sum of the transaction's input.

### 2.1.2.3 Blockchain & Blockchain forks

Creating blocks that contain transactions does not guide Bitcoin towards achieving a general consensus, unless these blocks are chained together, creating a chronological order over blocks and transactions (Decker, 2016). To this purpose, all valid transactions are grouped into blocks forming a directed tree. Apart from first block, which is known as genesis block, as it is illustrated in Fig.2.7, every block in this tree is linked with previous blocks by including a unique hash of the previous block in its header. In this case, the previous block  $B$  that is referenced by a block  $K$ , is classified as the parent of the block  $K$ . The root of the tree is the genesis block which is considered as an ancestor of all blocks. The longest path from the genesis block to any block in the tree is defined as blockchain (Decker & Wattenhofer, 2013). The height of a block in the blockchain is defined as the distance between the block and genesis block. The blockchain head refers to the block that is furthest away from the genesis block. In



addition, a summary of all transactions in a block is maintained and included in the blockchain. This summary is attached to every block in the blockchain in a form of a binary hash tree, known as *Merkle tree* (Karame et al., 2015) (Conti et al., 2017).

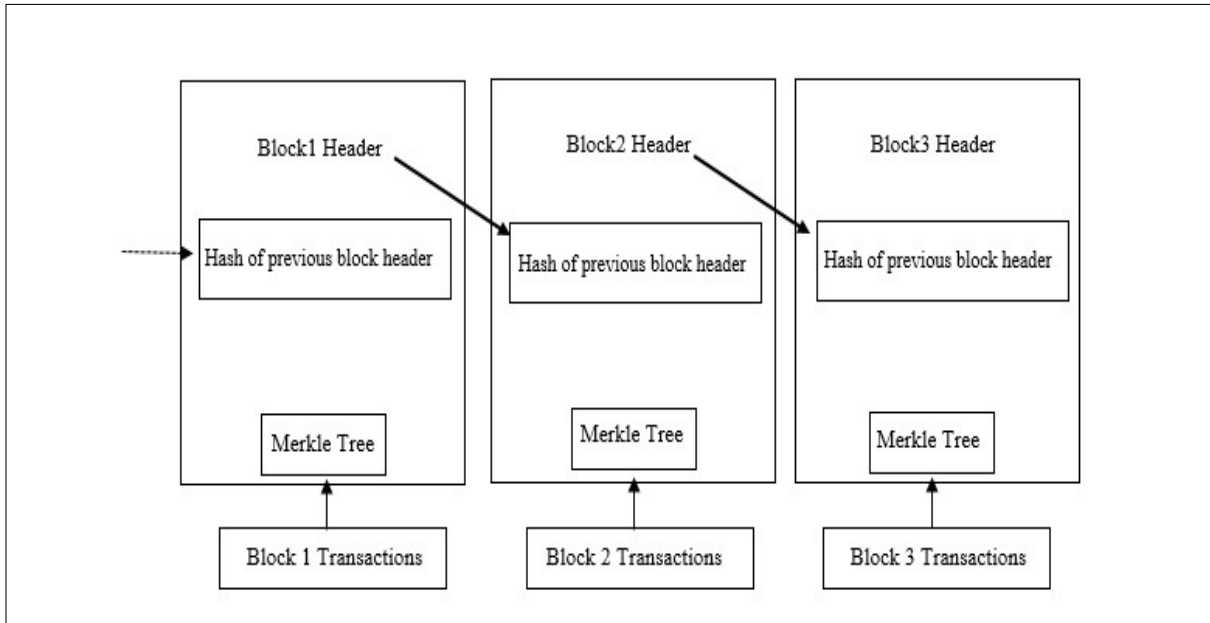


Fig. 2.7 Bitcoin Blockchain

Including the hash of the previous block in the block header ensures that the parent block has to be found before the child block. As a result, the chronological order of transactions is fulfilled, transactions in a higher block height have been found and verified after transactions that are included in a lower block height. However, this chaining supports the security of Bitcoin as linking each block to the previous block guarantees that every change in the parent block requires changes in the child block (Rosenfeld, 2014). Specifically, when any modification needs to be taken place in the parent block, the parent's hash changes. As the parent's hash is included in the child's block header, a change in the child's hash is required which, on the other hand, requires a change in the grandchild's hash and so on. This ensures that updating a block requires forcing a recalculation of all blocks following it. Consequently, massive computation effort is required for such a recalculation especially when a block has many subsequent blocks as it is computationally difficult. Therefore, blockchain is resistant to modifications from the most resourceful attackers (Miers et al., 2013). However, finding a block that can appear in the blockchain guarantees a reward with a new minted bitcoin. This encourages miners to find a block that can build on the current blockchain header due to the fact that building on an earlier block requires creating a longer branch than the current longest branch (Okupski, 2014). A branch is a path in the blockchain which starts from a leaf block to the genesis block (Androulaki et al., 2013).

However, the wide adoption of Bitcoin causes an enormous increase in the number of transactions which, on the other hand, imposes a constant growing in the size of the blockchain (Croman et al., 2016). This results in a non-negligible size of both blockchain and blocks. Nowadays, tens of gigabytes is the approximate size of blockchain. As a consequences, the transactions validation process requires more effort.

Due to the information validation process (transactions/blocks) that is fulfilled in a distributed way, *blockchain forks* might occur. The delay overhead in information propagation is considered as the main causes factor of blockchain forks (Decker & Wattenhofer, 2013).

Since a block can only have one parent block, it might have multiple children reference the same parent as a previous block (Antonopoulos, 2014). This scenario, known as blockchain forks, can happen when two different blocks have been found simultaneously at the same time by miners (Sompolinsky & Zohar, 2013). In this case, the blockchain becomes inconsistent due to the argument between nodes on which block is the current blockhead. Specifically, suppose  $A$  and  $B$  are two blocks represent the blockchain head. A conflict will be raised due to the fact that both  $A$  and  $B$  include a reward transaction. This conflict indicates that the blockchain is no longer consistent.

On receiving a new block  $b$  by a node  $N$  whose blockchain head  $a$  at height  $t$ , the node  $N$ , adds  $b$  to the blockchain and considers it as a new blockchain head with height  $t' > t$ . The new block  $b$  will be either merged with the same branch as  $a$  or with another branch. Specifically, an action has to be done by the node  $N$  while a new blockchain head  $b$  is found. This action is based on whether or not the block  $b$  is on the same branch as the block  $a$ . Precisely, when the block  $a$  is not an ancestor of the new blockchain head  $b$  which means that the block  $b$  is in a different branch, the length of the branch matters. If the block  $b$  is on a branch which is longer than the branch of  $a$ , it becomes the new blockchain head. Though, a common ancestor will be shared by both blocks. This helps the node  $N$  in updating its blockchain head through reverting all changes down till the common ancestor and changes will be applied on the branch of  $b$ . In case of both blocks are on the same branch, all the intermediate blocks in the branch will be retrieved and updated incrementally (Rosenfeld, 2014).

A Blockchain fork can be worse when every partition of the network has its own blockchain head which results in new blocks that are built based on partitions' respective blockchain heads. Eventually, forks should be resolved and only the block that is built on the longest branch will win, whereas, the rest of blocks, known as orphan blocks, will be discarded (Karame et al., 2015). All partitions should switch to the longest branch in terms of not adopting it. Furthermore, building blocks is continued on the longest 'locally' known fork that owns the highest computational power. A fork becomes the longest fork if its miners broadcast validations before others. This ensures that the fork 'overtakes' the current longest fork.

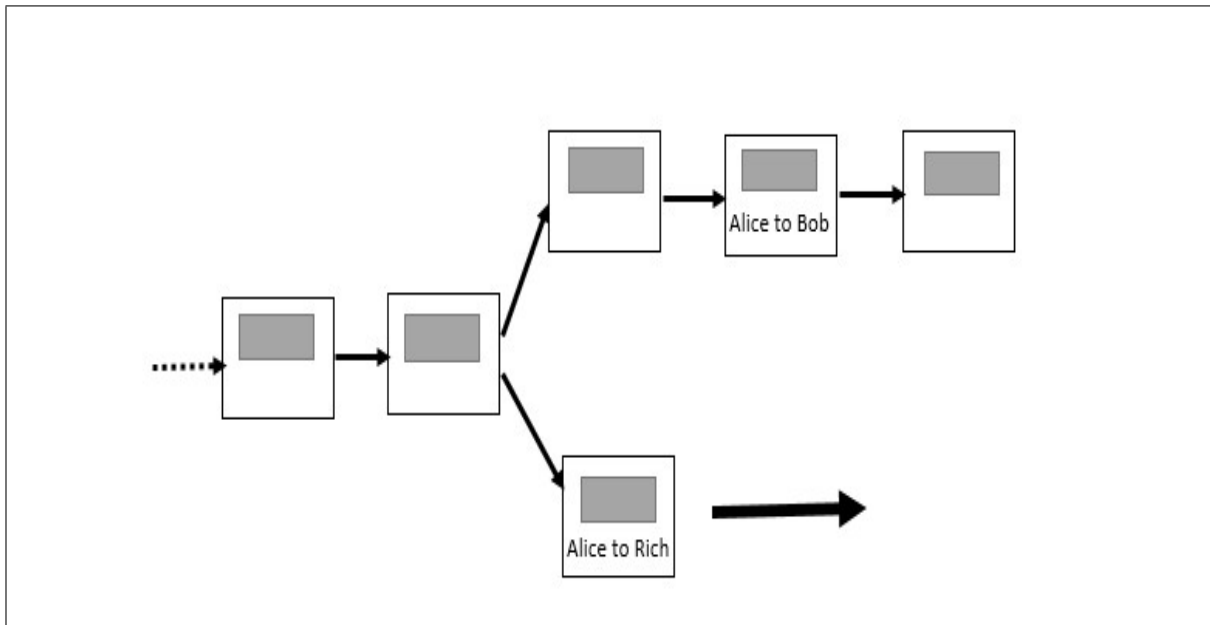


Fig. 2.8 Double spending (race attack)

Consider the example in Fig.2.8 in which Alice tries to double spend the same coins with Bob and Rich. At the same time, Alice has submitted a valid transaction to subset miners twice. Therefore, the same transaction appears in two different forks in the blockchain. The winner transaction is the Alice's transaction to Bob which appears in the longest fork. While the Alice's transaction to Rich is nullified as it tries to double spend the same coin. Rich will be informed that the received transaction is no longer valid once the fork is resolved.

Nevertheless, double spending attacks still have a potential with the presence of blockchain forks (Karame et al., 2015) (Rosenfeld, 2014). Suppose that Alice buys a product from Rich, Alice might be able to secretly mine a branch which includes, the transaction that returns the payment to himself, while disseminating the Rich's transaction. Alice will not broadcast this branch until the Rich's transaction gets confirmed. In this scenario, Rich is going to be confident about the transaction and then he will consider delivering products. Furthermore, Alice has to be sure that the secret branch is longer than the public branch, so if necessary, continue extending the secret branch. Finally, Alice broadcasts the secret branch when he confirms that the secret branch is longer than the public one. By doing this, Rich's transaction is no longer valid. Typically, this is computationally expensive, requiring an attacker to control 50% of the computing available in the network. The kind of attack known as double spending attacks with N-confirmations (Rosenfeld, 2014).

On the other hand, double spending attacks can easily happen within another scenario, known as double spending attacks with 0-confirmations (Karame et al., 2012) (Bamert et al., 2013). In this attack, blockchain accidental forks do not play any role, as the blockchain is not

checked at all. In particular, 0-confirmation attacks mostly happen in fast payments when a vendor accepts Bitcoin transactions and delivers products without waiting for the transactions' confirmations. Specifically, double spending attacks take place when two transactions ( $T_K$  and  $T_R$ ) are created by an attacker. These transactions have the same input (same source of Bitcoin) and different outputs (different recipients, suppose we have two transactions,  $T_K$  will go to the majority of peers and  $T_R$  will go to the vendor). A double spending attack is considered to be successful when the majority of peers accept  $T_K$  while the vendor accepts just  $T_R$ . This means  $T_K$  is confirmed before  $T_R$ . The acceptability of  $T_K$  by subsequent blocks as an original transaction would prevent the vendor redeeming the  $T_R$  because it is considered as an invalid transaction as it is trying to spend money which has already been spent.

#### 2.1.2.4 Miners

Mining is a novel mechanism in Bitcoin by which the general consensus regarding the set of committed transactions is achieved. In this mechanism, some nodes known as miners try to solve proofs of work through finding inputs to hashes that yield digests with many leading zeros (Eyal & Sirer, 2014). A solution for proofs of work can only be achieved through brute force. The time that is taken by miners to solve the proofs of work is based on the difficulty of the puzzle (target), 10 minutes on average (Conti et al., 2017). Finding a solution for the proof of work problem means that the hash of previous puzzle as well as a group of transactions are included in a block which will be propagated to the rest of the Bitcoin network. Honest miners accept the block and start working to append it.

However, miners main tasks are described as follows. Firstly, miners collect pending transactions in order to form blocks, instead of verifying individual transactions. Secondly, miners validate a block by calculating the hash of the block through trying several nonce values until getting a hash value that is equal to or lower than the given target. As the target is readjusted after every 2,016 blocks, the author in (Kraft, 2016) proposed an equation to calculate the target value of the Bitcoin system. This equation can be written as:

$$T = T_{prev} * \frac{T_{actual}}{2016 * 10min} \quad (2.1)$$

Where  $T_{prev}$  is the old *target* value, and  $T_{actual}$  is the time period the the Bitcoin network took to generate the last 2,016 blocks.

The third task of miners is to propagate blocks along with the calculated hash to the rest of miners and append it to its blockchain. Other miners validate the received blocks through verifying the calculated hash value against the target value. Those miners add the received block to their blockchain if it passes the verification process. Once a block is successfully

added to the blockchain, the miner who solved the poof of work problem before others wins a reward. Therefore, miners in Bitcoin system are racing with each other to win the reward, a more computational power a miner maintains, a more chances to win the race (Croman et al., 2016).

A reward that is given for miners who solve the proof of works problem is represented in two forms. A small fee of a transaction committed in a block is considered as one of the reward forms which is calculated as the difference between the input and output of the transaction. While the other form of the reward is a set of newly generated coins. To claim the reward by a miner, a special transaction that has no input, known as coinbase, is included (Tschorsch & Scheuermann, 2016). However, some miners choose to collaborate with each other in order to produce a combined CPU power that increases the chances of being able to extend the blockchain. This collaboration is translated through establishing ‘mining pool’ where miners work together and each of them is rewarded a part of the received reward. On the other hand, mining pools is critical in a case where majority of miners join one pool as this pool might be able to achieve the majority of the mining power that could possibly control the network through preventing the rest of the network from participating in maintaining the blockchain.

## 2.2 The role of broadcast in the Bitcoin network

As it is mentioned earlier, Bitcoin follows a distributed trust mechanism which relies on a distributed validation and tracking of transactions. Based on this mechanism, a Bitcoin transaction has to be broadcasted to all nodes within the network to reach a consensus about which transactions are valid. The consensus is recorded in a publicly distributed ledger which is shared by the entire network (Spagnuolo et al., 2014). As the distributed validation that is achieved by the Bitcoin protocol is based on a replicated ledger which is collectively implemented by a network of volunteers, achieving a consistent sequential log of transactions through updating and synchronizing the ledger replicas is considered as the main goal of the Bitcoin protocol. For this purpose, two main types of information need to be propagated in the network: transactions and blocks.

Information(transactions/blocks) validation process in the Bitcoin network is achieved through a gossip-like protocol to broadcast information throughout the network (Bamert et al., 2013). Transactions are disseminated through the network using a protocol, which includes propagating two types of messages, an INV message and a GETDATA message (Decker & Wattenhofer, 2013). In order to avoid sending a transaction to a node that already receives it from other nodes, the transaction is not forwarded immediately. Instead, as shown in Fig.2.9, the transaction availability is announced first to nodes once the transaction has been verified.

When a node receives a transaction from one of its neighbours, it sends an INV message containing the hash of the transaction to all of its peers. On receiving an INV message, a node checks whether the transaction has been received before. If it has not been seen before, the node will request the transaction by sending a GETDATA message. Responding to the received GETDATA message, a node sends the transaction's data. Valid received transactions will be collected and included in a block by a node that generates the block. A block availability will be announced to other nodes, as explained in Fig. 2.9, following the same mechanism of transactions availability announcement (Decker, 2016).

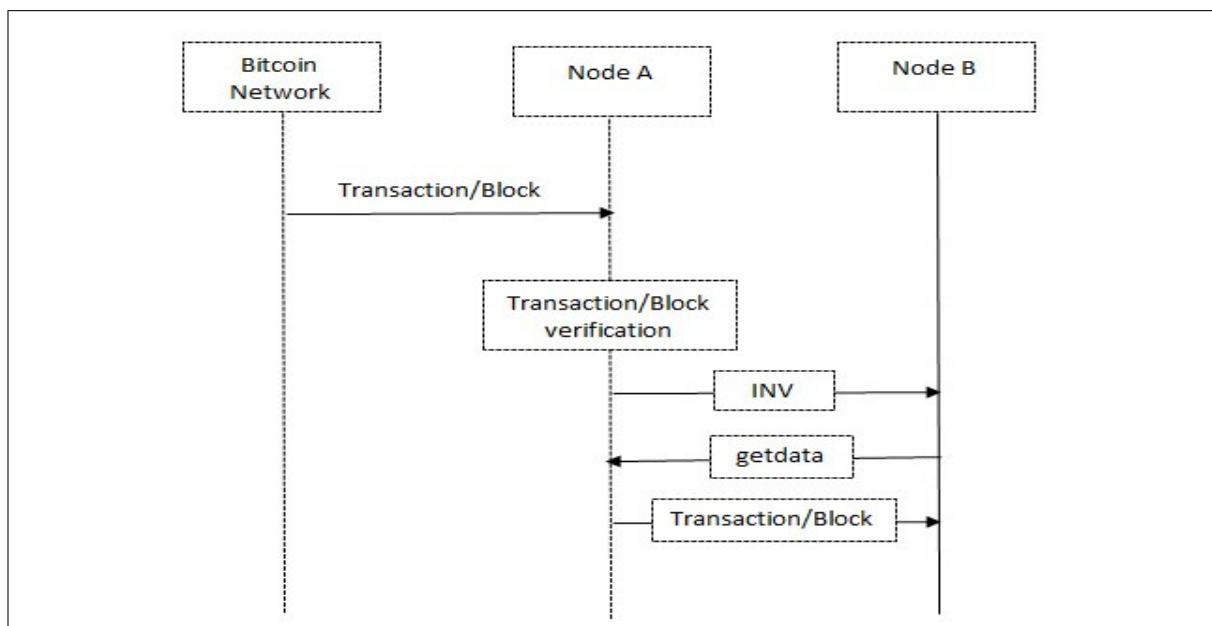


Fig. 2.9 Information propagation mechanism between Nodes A and B

An INV message is not propagated to all of the connected peers at the same time, instead, every 100ms, an INV message is sent to a random selected peer of all connected peers (Miller & Jansen, 2015). This means the number of connected nodes has a direct impact on the time that is required for forwarding the INV message.

In order to establish a quick agreement on valid transactions, transactions need to be submitted for timely inclusion in the directed transaction graph. Alas, when a delay in the agreement is experienced or transactions have prevented from entering the directed transaction graph, attackers has an opportunity to ban users from services. However, an attacker might be able to revert an agreed-upon graph which, on other hand, cause an attacker being able to steal funds by double spending (Karame et al., 2015). This issue will be discussed in the following section.

## 2.3 Information propagation delay

Based on the aforementioned scenario of information dissemination in the Bitcoin network, nodes synchronize their replica of the public ledger and form the entire consensus of the Bitcoin network. However, reaching a consistent state in the network is mainly based on how quickly information(transactions/blocks) is spread throughout the network. As the agreement between nodes regarding a common transactions history is affected by the delay in the distribution of a verified information(transaction/block), speeding up information propagation can contribute in tackling the risks of an inconsistency state in the Bitcoin network (Decker, 2016). On the other hand, if a peer in the Bitcoin network is able to disseminate a data item quicker than other peers in the network, the peer could gain inappropriate profit through applying a scenario of double spending attacks (Conti et al., 2017).

As the Bitcoin network topology is not proximity defined, connecting to other peers is maintained randomly without considering any proximity criteria. In other words, long-distance links are not taken into consideration when the Bitcoin physical network topology is built. This increases non-compulsory hops that the information passes through. In addition, as it is mentioned in (Decker & Wattenhofer, 2013), the sheer distance between the origin of a transaction or block and other nodes is deemed as the most significant problem in the Bitcoin network. As a result, transaction verification process is slower (Stathakopoulou et al., 2015). Hence, the potential of double spending attacks, that are more difficult to discover in slow network, increases due to the conflict between nodes regarding the transactions history.

Turning now to clarify how the information propagation delay imposes Bitcoin being vulnerable to some classes of attack. Centralized digital currencies requires a central authority to be in place with the aim of preventing double spending attacks to be taken place (Ahmad et al., 2013). In contrast, Bitcoin relies on protocol rules which play a key role in avoiding double spending attacks. These rules state that only previously unspent transaction outputs might be considered in the input of a follow-up transaction (Nakamoto, 2008). As unspent transaction output is verified against the common transactions history, public ledger has to be consistent. Due to a situation where not all nodes agree on the same blockchain header, inconsistency in the blockchain are unavoidable and spending the same bitcoin twice has a potential. The main cause of this situation is the delay overhead in the transaction verification process where all transactions must be verified by all nodes in Bitcoin to achieve an agreement regarding a common transactions history. Therefore, validating a transaction against an unsynchronised public ledger would reduce the chances of getting a high probability agreement over the Bitcoin network nodes. This results in a conflict regarding the validity of a particular transaction which may facilitate rewriting the transaction history by an attacker. Specifically, uncertainty



regarding the validity of a given transaction reduces the chance of achieving agreement of all nodes on the same blockchain header which, on the other hand, would cause blockchain forks.

As is mentioned earlier, forks are created when two blocks are possible to be created simultaneously, each one as a possible addition to the same sub-chain. According to (Bitcoin Wiki, 2008) (Sompolinsky & Zohar, 2013), as it is illustrated in Fig.2.10, a transaction can appear in two different branches of the blockchain. Though, as discussed in (Karame et al., 2015), in the special case where Bitcoin is subject to blockchain forks, attackers might be able to impose their own transactions history, possibly to reverse transactions they sent so as to successfully perform double spending attacks (Rosenfeld, 2014). Specifically, attackers can secretly mine a branch which includes, the transaction that returns the payment to themselves, while disseminating the merchant's transaction. However, blockchain forks are caused by the delay overhead of information propagation, a fact that has been mentioned in (Decker & Wattenhofer, 2013).

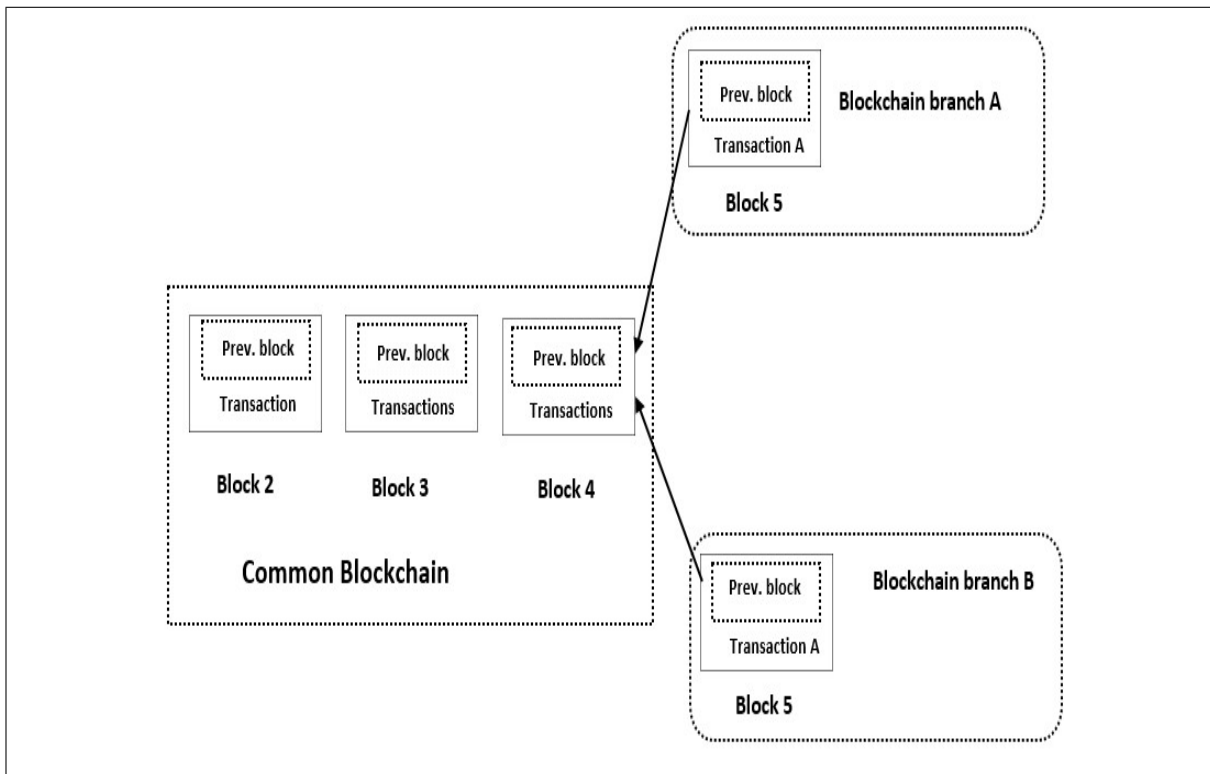


Fig. 2.10 The transaction A appears in two branches A and B

Consider the example in Fig.2.11 where Alice attempts to double spend 10 bitcoins to both Bob and Charlie. Alice creates two identical transactions with the same amount of bitcoins and propagates them in the network at the same time. Those transactions confuse the network nodes and a Bitcoin fork might appear when those transactions are distributed in the network inappropriately due to the propagation delay of transactions and /or blocks. As its shown in



Fig.2.11, both of Alices' transactions are accepted by a different group of nodes which confirms the disagreement between those groups about which transaction is valid. Therefore, speeding up information propagation in the Bitcoin network would give a better distribution of these information which reduces the possibility of the blockchain fork occurrence.

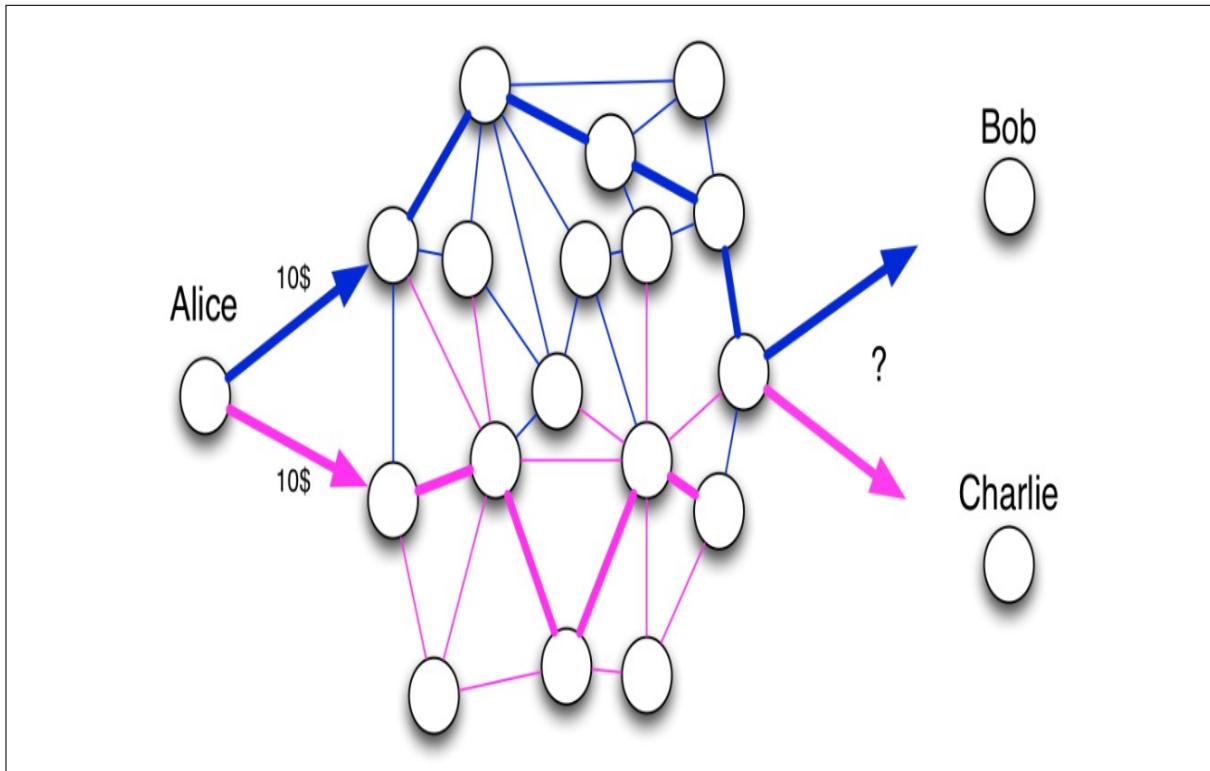


Fig. 2.11 Disagreement between the Bitcoin network nodes about which transaction is valid

Miller & Jansen (2015) research has shown that the number of nodes in the Bitcoin network and the structure of the overlay network have a great impact on the transaction propagation time. Their results showed that the overlay topology of the Bitcoin network, which is not geographically localised, offers inefficient transaction propagation time.

Based on the aforementioned issues which indicate the importance of speeding up the information dissemination, the ability to speed up information propagation in the Bitcoin network is considered as the main challenge that must be faced in Bitcoin. Therefore, accelerating information dissemination is considered as the main intended impact of this work which would contribute to the direction of tackling the problem of the agreement on a common transaction history among nodes in the Bitcoin network.

However, more recent attention in the Bitcoin field has focused on the problem of the delay overhead in information propagation which is linked to the problem of reaching a consensus in the Bitcoin network. The first attempt to overcome the information propagation delay in Bitcoin

network was proposed in (Decker & Wattenhofer, 2015) which suggested an offline payment known as a micropayment. This payment that consists of a separated type of transactions, passes through special micropayment channels (Rosenberg, 2010). Though, transactions are propagated outside the Bitcoin network, in a network of payments provider. However, this solution does not handle the decentralized concept of Bitcoin. In the following section, related work in speeding up information propagation in the Bitcoin network as well as other peer to peer networks, and in modelling approaches to avoid double spending attacks will be critically reviewed.

## 2.4 Related Work

In this section, prior theories and approaches that impose some changes in the Bitcoin network with respect to nodes' behaviour with the aim of applying some modifications in the information propagation protocol, and overlay Bitcoin network topology, will be critically discussed. The main objective of these theories and approaches was to overcome information propagation overhead which, on the other hand, assist in mitigating double spending attacks throughout reaching to a consistent transactions log. In addition, previous theories that are proposed with the aim of mitigating the information propagation delay in other classes of peer-to-peer network will be critically discussed. Moreover, existing methods to tackle double spending attacks in the Bitcoin network will be highlighted in this section.

### 2.4.1 Minimize verification

There have been several investigations that aim to reduce the information propagation time throughout minimizing the time of information(transactions/blocks) verification. As it is mentioned earlier, when a node receives a transaction/block, it verifies whether it is valid or not. If the transaction/block is valid, the node forwards it to its neighbours. In contrast, invalid transactions/blocks are discarded. The idea of reducing the information verification time, in particular, block verification time has been adopted in (Decker & Wattenhofer, 2013) as a way to speed up information propagation. Basically, the verification process is divided into two parts:

1. Initial difficulty check
2. A transaction validation

The first part involves comparing the block header that includes the proof of work hash against the current difficulty target. In addition, this part includes validating the received block against the recent block in order to make sure that the received block is not a duplicated block

as well as being referenced by the recent block which is considered as its predecessor. This guarantees the received block has not been submitted before.

In order to minimize the block verification time, the *minimize verification* protocol that has been proposed in (Decker & Wattenhofer, 2013) has stated some changes in the Bitcoin nodes' behaviour that make every node fulfills only the first part of the block verification process. Specifically, when a node receives a block, it checks the proof of work difficulty and forwards the block to its neighbours, rather than suspends the relay until the validation of all transactions in the block is completed. This would minimize the block propagation delay in the Bitcoin network.

However, the change in the nodes' behaviour mentioned above is more likely to have a security risk on Bitcoin as discarding transactions validation would give a great chance to an attacker to flood the network with invalid transactions which, on the other hand, results in a distributed denial of service attack. In addition, the change of nodes' behavior does not take into account the transaction propagation delay which means that transactions would still pass through the original information broadcasting scenario. As a result, the change does not have a large impact on the overall information propagation delay.

As is mentioned earlier, blockchain plays an important role in information validation where transactions/blocks are validated against the blockchain. Though, the size of the blockchain matters in the process of mitigating the delay of information propagation. In this regards, some research focus on scaling up the blockchain as a way to reduce the information verification time. In this context, Back et al. (2014) proposed in their white paper a new technology, known as sidechain, which overcomes the limitations of the size of the blockchain via creating independent pegged sidechains. The sidechain allows secure transferability for bitcoins or different assets between multiple blockchains. As shown in Fig.2.12, SPV proof states that a user who is willing to use his coin in other chains, has to prove that his coin has been successfully locked in its own chain, and send this proof to other chains to express to users on those chains that the coin is valid.

Sidechain with the SPV proof seems to have more adoption than other scaling up technologies due to the convenient transfer of the coins between chains. This implies that a user with the sidechain technology does not need to make coins or assets compatible when dealing with a different chains. However, the major flaw that could affect the adoption of the sidechain is that a user has to wait for approximately two days in order to confirm that the coin or asset is locked into a valid transaction. So it offers inconvenient time compared to other technologies. Furthermore, the sidechain approach does not give any type of security for the sidechains. In other words, these chains do not have a huge hash rate, so it is quite easy to abuse those chains

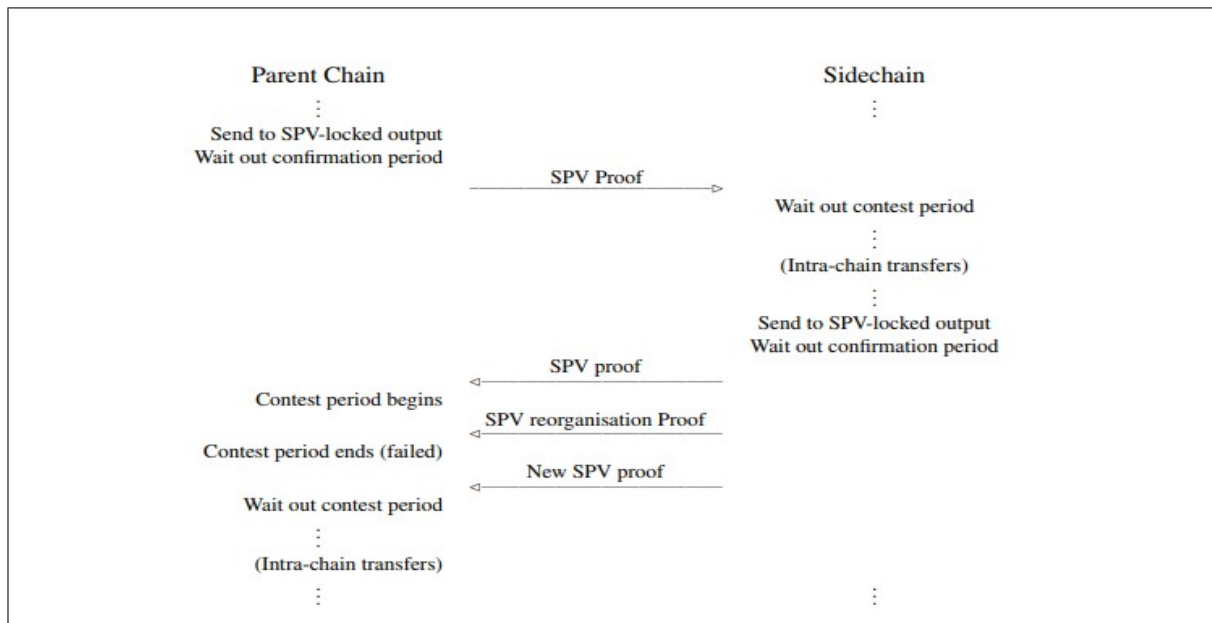


Fig. 2.12 Sidechains with SPV proof (Back et al., 2014)

by applying 51% attack where an attacker is being able to generate the longest chain in the long run (Bastiaan, 2015).

In the same direction, another theory has been proposed in (Pazmiño & da Silva Rodrigues, 2015) which focuses on the blockchain as a main factor of reducing the transaction verification time. As transactions are validated against the blockchain that currently contains a history of all transactions and it still grows in the size with each new transaction, it has been claimed that reducing transactions history at each node plays an important role towards achieving an optimal transaction verification time (Pazmiño & da Silva Rodrigues, 2015). Precisely, Pazmiño & Rodrigues have proposed a new algorithm, known as *BASELINE*, in which the blockchain is divided at each node in the Bitcoin network into several parts  $n$ . These parts are distributed at each node on several local computers. As all parts represent the same user, public/private keys are same for all parts. On the other hand, each part has a different portion of the public ledger. Their evaluation experiment has shown that the verification time could be enhanced by 71.42% if the blockchain is divided at a given node on five computers. This means that an improvement in the information propagation delay could be achieved when a number of division at each node is greater. However, the proposed *BASELINE* algorithm is less likely to be adopted as a realistic solution for the propagation delay problem due to the expensive requirements where every node in the network should maintain several local computers.

In the same context where researchers focused on speeding up information propagation in conjunction with minimizing the blockchain size, Sompolinsky & Zohar (2013) proposed a new approach that improves the scalability of the blockchain by performing more security

for off-chain blocks through miners. More precisely, miners would have the responsibility of keeping track and protecting the soft forks that are linked to the main blockchain. In principle, this approach considers miners as a trusted third party and gives them more control over the Bitcoin network. Therefore, this approach stands against the decentralisation concept of Bitcoin, resulting in minimizing the security awareness. Furthermore, these soft forks are subject to 51% attack as these forks maintain less hash rate.

On the other hand, the Blinkchain approach is introduced in (Basescu et al.) that focuses on minimising the transaction verification time with the aim of decreasing the consensus latency. The Blinkchain approach is based on splitting the blockchain into localised shards, one blockchain per geographical location. Each blockchain is associated with a number of nearby validators. This reduces the transactions history at each blockchain which leads to speed up the transaction verification process. However, this approach reduces the resistance of blockchain to 51% attacks as these blockchains offer less hash rate. Furthermore, this approach doesn't offer any interoperability technique that allows shard blockchains to interact with each other.

On the basis of above descriptions about limitations of minimize verification protocol, sidechain technology, BASELINE algorithm, security of soft forks approach, and Blinkchain approach, it motivates the development of a new approach/approaches to overcome these limitations with respect to overall information propagation delay improvement as well as security efficiency. The new proposed approaches is further introduced in chapter 3.

### 2.4.2 Pipelining information propagation

As introduced in Stathakopoulou et al. (2015), faster information propagation can be achieved by pipelining information dissemination in order to minimize the round-trip times between nodes and their neighbours. Specifically, this solution claims that incoming INV message which includes a list of hashes of the available transactions, can be immediately forwarded instead of waiting to receive transactions. Therefore, nodes can ask for a transaction even though it has not arrived yet. On receiving the transaction, it will be forwarded immediately for nodes that have asked for it, considering that a GETDATA message has already been received from those nodes. By doing this, the idle time in which nodes are normally waiting for the GETDATA message to arrive, would be utilized. Suppose  $a$ ,  $b$ ,  $c$  are nodes connecting to each other and an INV message of a transaction Tx arrives to the node  $a$ . As shown in Fig.2.13 and Fig.2.14, two scenarios will be followed based on the availability of the transaction Tx. In the first scenario, Fig.2.13, the node needs to keep the transaction as it has arrived before receiving any GETDATA message. While in the second scenario, Fig.2.14, the node needs to keep the GETDATA message as it has arrived before receiving the transaction TX.

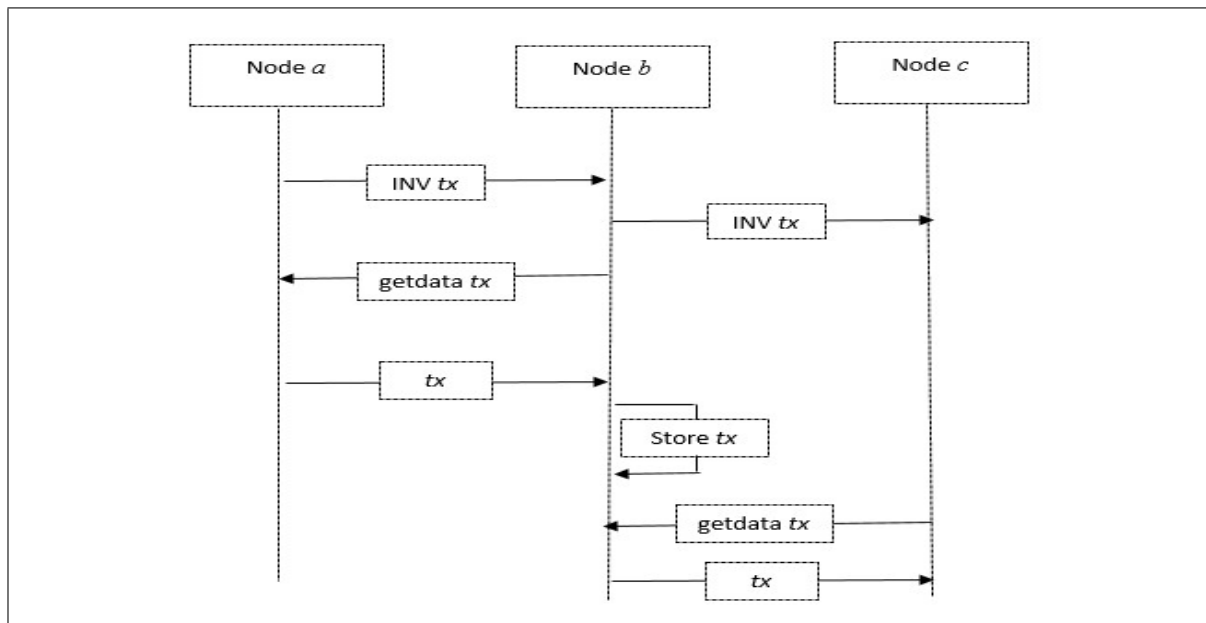


Fig. 2.13 First scenario where tx arrives before GETDATA

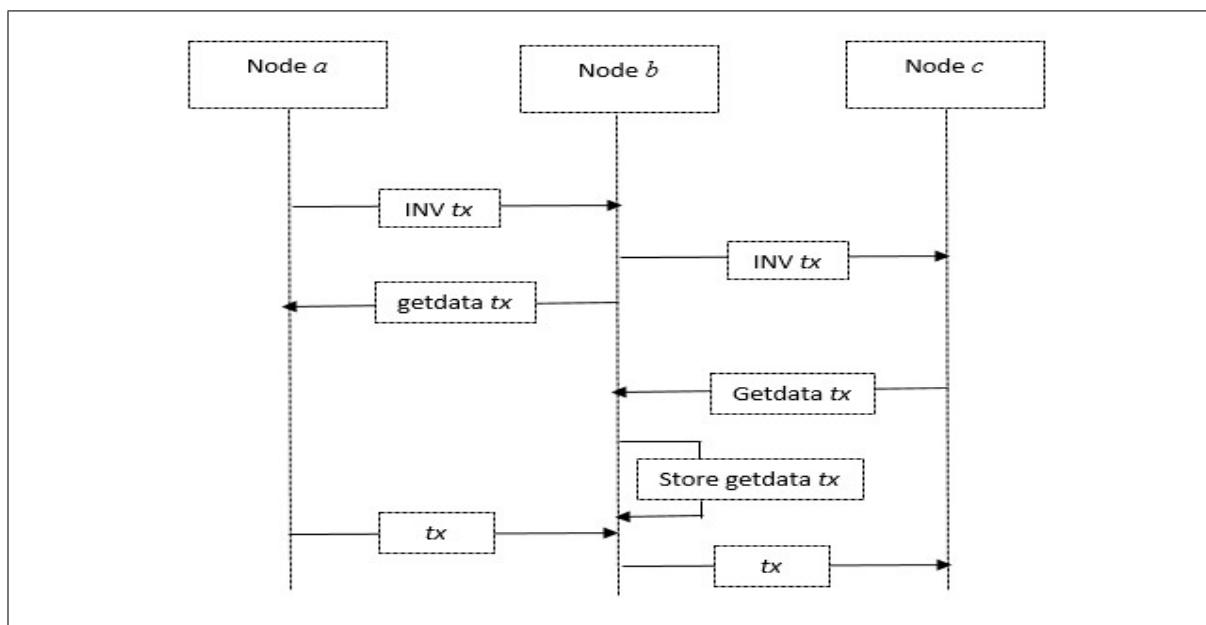


Fig. 2.14 Second scenario where GETDATA arrives before tx

The key problem with the pipelining propagation protocol is that the Bitcoin network global state might become inconsistent when nodes request a transaction that is not available. As a result, inconsistent network leads to increase the chances of performing successful double spending attacks. Furthermore, the pipelining propagation protocol requires unlimited memory at every node with the aim of keeping either transactions until a GETDATA message arrives,

or GETDATA messages until transactions arrive. Moreover, it is believed that this theory can reduce the information propagation delay with a very low rate as transactions still need to pass through random and unlocalized connections to visit most of the Bitcoin network nodes.

On the basis of above descriptions about limitations of the pipelining propagation protocol, it motivates the creation of a new representation of the network topology that performs less number of network hops that information pass through as well as supports the proximity of connectivity between nodes. This would result in speeding up information propagation which, on the other hand, make the network more consistent. Therefore, chances of performing double spending attacks would be less. The new representation is further described in Chapter 3.

### 2.4.3 Connectivity increase

As it is mentioned earlier, the sheer distance between the initiator of a block or transaction and nodes is considered as the most causative factor of the information propagation delay in the Bitcoin network. Studies in (Decker & Wattenhofer, 2013) claimed that increasing the network connectivity throughout minimizing the distance between any two nodes, by creating a star sub-graph topology that is used as a central communication hub, has a large effect on the reduction of the information propagation delay. Specifically, a new network topology is proposed in (Decker & Wattenhofer, 2013) in which each node maintains a connection pool that is able to keep up to 4000 open connections. As a result, the node mostly connects to every single advertised address. Therefore, information would visit less number of hops which reflects faster information propagation. However, the Bitcoin protocol allows nodes to maintain up to 8 outgoing connections in order to prevent controlling the network by malicious nodes (Biryukov & Pustogarov, 2015). Though, the proposed network topology raises severe security risk due to the fact that nodes are permitted to maintain many connections to other nodes. Therefore, malicious nodes might be able to control the network and easily disturb, abuse the network.

On the basis of maximising the proximity of connectivity, another change in the Bitcoin network topology has been proposed in (Stathakopoulou et al., 2015). This change increases the geographical connectivity in the Bitcoin network through several coordinator nodes, distributed strategically around the globe, known as CDN Bitcoin client. These CDN clients are able to search and suggest Bitcoin network nodes to each other based on the geographical location. Specifically, the CDN client calculates the geographical distance between the discovered node and other CDN clients. By doing this, the CDN client are able to recommend geographically closest nodes to other CDN client. Compared to the protocol that is proposed in (Decker & Wattenhofer, 2013), CDN clients are allowed to maintain many (up to 100) outgoing connections to the nodes that are geographically close.



The main downside of this solution is that, any node can be a CDN client which makes Bitcoin more vulnerable to some classes of attack. Specifically, malicious nodes can easily impersonate the role of CDN clients and maintain connections to many nodes in the network. This results in malicious nodes being able to control a big portions of the network nodes. As a result, DDOS attacks and partition attacks would be possible to launch. Another concern is raised with the CDN clients protocol is that it is relatively centralized as any CDN client can be used as a coordinator node without meeting any requirements or achieving an agreement over the network nodes. Furthermore, the idea of recommending closer nodes to other nodes will not have a high impact on the overall network connectivity if it is implemented by limited nodes that are not well connected.

The above descriptions indicate that the limitations of the connectivity increase protocols in the above mentioned aspects could result in significant security implications as well as a low level impact on information propagation. This thus motivates the development of another connectivity increase approaches that are further introduced in Chapter 3. In the following section, prior attempts to analyse and mitigate double spending attacks in the Bitcoin network with respect to 0-confirmations and N-confirmations will be critically discussed.

#### 2.4.4 Mitigating double spending attacks

Some research has been done towards analysing and mitigating double spending attacks with respect to both scenarios, 0-confirmations and N-confirmations. Regarding N-confirmation double spending attacks, measurements of the probability of double spending attacks based on measurements in the real Bitcoin network have been provided in [Decker & Wattenhofer \(2013\)](#) through developing an analytical model of Bitcoin. Furthermore, a strong correlation between the size of a message and the propagation delay has been observed. As an adversarial fork of the blockchain still causes a possibility of double spending, some research have admitted that reducing the possibility of accidental forks would help in double spending attacks avoidance ([Karame et al., 2015](#)) ([Sompolinsky & Zohar, 2013](#)).

In the context of 0-confirmation, a model which considers some modifications in the transaction dissemination protocol has been presented in ([Karame et al., 2015](#)) ([Bamert et al., 2013](#)). The main intuition behind these modifications is to mitigate double spending attacks in fast payments. In ([Karame et al., 2015](#)), a new model was proposed which allows the vendor receives  $T_K$  (conflicting transaction) and  $T_v$  (honest transaction that sent to the vendor) almost at the same time. This would help the vendor to discover double spending attacks at the right time before delivering products. Specifically, the core idea of this model is that when a transaction is received by a node, it adds the transaction to its pool and forwards it to the other nodes if the transaction has not been seen before. Otherwise, it directly forwards the transaction to other



neighbours without adding it to its pool. This scenario allows the conflicting transaction  $T_k$  to be received by the vendor before delivering products. Though, the vendor would immediately detect the attempt of the double spending attack when the conflicting transaction  $T_k$  is received. The most serious disadvantage of this method is that a large volume of nonessential traffic would flood the network which results in inefficient performance of the Bitcoin network.

As a realistic solution, a prototype system which is applied in vending machines was proposed in (Bamert et al., 2013). This system has performed a fast payment with 0.088 as a probability of double spending attacks through setting a server that observes transactions. This server gives a signal, which indicates that a transaction has been confirmed to the blockchain, when the transaction is propagated and reached over 40 nodes. Unfortunately, this solution is limited because an attacker's transaction could still be propagated to the majority of nodes. That disproves the claim of considering a transaction is approved if it is received by 40 nodes.

#### 2.4.5 Mitigating propagation delay outside the electronic currency field:

To date, several studies on the reduction of propagation delay in some of P2P applications through overcoming the random assignment of connections, have been addressed. These studies have not dealt with security influences due to the nature of these applications which are not related to the currency field. Whereas applying any approach that aims to overcome the propagation delay problem in the Bitcoin network should be done without compromising security.

Schollmeier & Kunzmann (2003) proposed a new conceptual architecture, known as GnuViz architecture, which adds location awareness to the Gnutella network in order to overcome the problem of the random assignment of connections. This architecture relies on a crawler that explores the network structure in order to identify connections between nodes. Therefore, the crawler uses PING and PONG messages to create a record that includes the IP addresses of nodes and connections between these nodes. After building this record, IP addresses of the explored nodes are mapped to their geographical location using NetGeo (Ma, 2006). The main limitation of GnuViz architecture is that it does not simulate the reality as the mapping can only be done offline.

Similarly, there is a basic approach that has been adopted to reduce propagation delay in the BitTorrent network. Bindal et al. (2006) have proposed an approach, which aims to improve the traffic locality in the BitTorrent network throughout applying biased neighbour selection method. In this method, nodes connect with the majority of its neighbors within the same ISP. That means each node will be forced to connect to the geographically closest nodes. Specifically, nodes know about the other close nodes by getting locality information from a tracker. Therefore, when a node contacts the tracker in order to discover nodes in the same ISP,

the tracker hands back 35 internal nodes (within the same ISP) as well as a number of external nodes. After that, the node connects to those 35 nodes and form a cluster. The tracker notifies the node with more nodes if there are less than 35 internal nodes. The tracker follows several mechanisms in order to have a significant knowledge about ISP locality of nodes. One of these mechanisms is to use a packet inspection in which packets are sent to discover the traffic of the network. Whereas the other mechanism is to use internet topology maps or AS(autonomous system) mappings to identify the ISP.

The key weakness with the biased selection protocol is that the protocol does not follow a distributed algorithm that runs by each node independently. Instead, nodes rely on trackers to get information about other nodes in the same ISP locality. Therefore, any failure within the tracker or delay in the tracker's responses would affect handling the protocol. On the other hand, there is a traffic overhead when trackers use packets inspection to get information about the nodes ISP locality. Though, this motivated the development of a localised based clustering protocol that follows the distributed algorithm principle where each node runs the protocol independently by information about discovered nodes and local neighbours.

Regarding Mobile Adhoc network, most of the previous studies have focused on presenting an extension of best effort routing algorithms as a method to overcome the propagation delay problem. [Lee et al. \(2002\)](#) have proposed a new routing protocol that is on-demand. In this approach, a route is discovered only when a node needs to send data. Specifically, when a node needs to discover a new short route, global search (flooding) procedure is followed. For the purpose of discovering a route, two packets are propagated, route request (RREQ) and route reply (RREP). RREQ is initiated by a node that wants to discover a route. After that, the initiator node broadcasts the packet to all of its neighbours. When a node receives the packet, it rebroadcasts it further and records details about the request as well as a backward pointer to the initiator node. On receiving the RREQ by a destination node, a RREP is built and sent to the initiator of the RREQ by using the backward pointer. The RREP is sent at each hop by uni-cast and a forward pointer to the destination node is recorded at each hop via the previous hop of the RREP. Based on the backward and forward pointer, initiator node can establish a short path to the destination and build a mesh of nodes. However, this approach can be effective in small networks only as routes in large networks might experience a quite significant delay.

However, a number of authors have attempted to implement a hierarchy organization in ad hoc networks which results in localised information dissemination. As a result, cluster based protocols have been proposed in ([Marshall & Erciyes, 2005](#)) ([Erciyes & Marshall, 2004](#)). These protocols group nodes in a locality together into a cluster, and assign one node to be a cluster head that is responsible for maintaining the cluster. Cluster head is responsible for routing information between nodes on an inter-cluster basis until the cluster containing the node is

found. Electing a cluster head is done based on some criteria such as choosing a node with the lowest ID and a dynamic allocation based upon remaining battery life (Gavalas et al., 2006).

However, the clustering approach mentioned above does not take into account whether the a cluster head and nodes within a cluster are close in the physical network or geographical location. Due to this, the information propagation delay is not significantly affected. Moreover, aforementioned clustering approach can not meet the security requirements of Bitcoin in case of considering it to be applied in Bitcoin network due to the election mechanisms of the cluster head. These mechanisms ensure the performance of the cluster head without considering rules that make impersonating the role of cluster head challenging. Therefore, this motivated the development of clustering approaches that consider the nodes proximity within the geographical and physical network as well as meets the security requirements of Bitcoin network throughout challenging rules for occupying the cluster's head role as well as a distributed mechanism for cluster heads election.

## 2.5 Conclusion

This chapter highlights the main components of Bitcoin system. In addition, the essential parts of Bitcoin system that the Bitcoin protocol relies on, have been indicated and widely explained. This chapter also focuses on Bitcoin from a networking perspective. Furthermore, the main structure of transactions and blocks as well as the role of transactions and blocks in achieving the Bitcoin protocol were described in details. Moreover, how the blockchain is built as well as the strengths and limitations of the blockchain were analysed. In this chapter, we focused on one of the blockchain limitations, in particular , blockchain forks that have a significant security risk on Bitcoin. This risk is raised due to the fact that the blockchain forks give a great chance to double spending attacks to be launched. Therefore, we analysed double spending attacks in Bitcoin system. However, information propagation delay is considered as the main cause factor of blockchain forks.

In this chapter, information propagation delay problem in the Bitcoin network and how it plays an important role in increasing the possibility of blockchain forks, were critically discussed. This chapter also reviews some existing methods and techniques that attempt to overcome the information propagation problem in the Bitcoin network as well as other peer-to-peer networks. On the other hand, the strengths and limitations of the methods and techniques are highlighted in this chapter. These thus motivate the development of more advanced methods and techniques which are further introduced in the following chapter.



# Chapter 3

## The Proposed Clustering Approaches

### 3.1 Introduction

Chapter 2 reviewed several methods of speeding up information propagation in the Bitcoin network. A list of limitations of these methods are highlighted and lead to the development of new approaches for accelerating information propagation. As it is mentioned earlier, information propagation delay can be responsible for the Bitcoin network being inconsistent. The sheer distance between the origin of a transaction or block and other nodes is deemed as the most influence problem in Bitcoin network ([Decker & Wattenhofer, 2013](#)). Ignoring this important problem may result in a considerable increase in communication costs and delays. As a result, transaction verification is slower ([Stathakopoulou et al., 2015](#)). Hence, the potential of double spending attacks, that are more difficult to discover in slow networks, increases due to the conflict between nodes regarding the history of transactions. Taking that into account, grouping peers according to their real-world proximity would minimize the distance between any two nodes which results in minimising average latencies for information delivery between peers.

This chapter will outline the proposal for a form of proximity of connectivity in the Bitcoin network which is considered as a mechanism to tackle the problem of information propagation delay in the Bitcoin network. The proposal includes introducing four clustering approaches namely LBC, BCBPT, BCBSN, and MNBC. Therefore, the first contribution mentioned in [Section 1.8](#) is fulfilled in this Chapter.

### 3.2 Locality Based Clustering (LBC)

As it is mentioned in Chapter 2, prior approaches that focus on increasing the proximity of connectivity in the Bitcoin network shows significant security implications due to their nature

that stands against the decentralization principle of Bitcoin as well as maximising the chances of controlling the network throughout allowing each node to maintain more than 8 outgoing connections. In addition, previous approaches are implemented by limited nodes which are not well connected, resulting in a low level impact on information propagation. Therefore, a location based clustering protocol, named Locality Based Clustering (LBC), that overcomes security and performance limitations of previous approaches with the aim of maximising the proximity of connectivity in the Bitcoin network without compromising security.

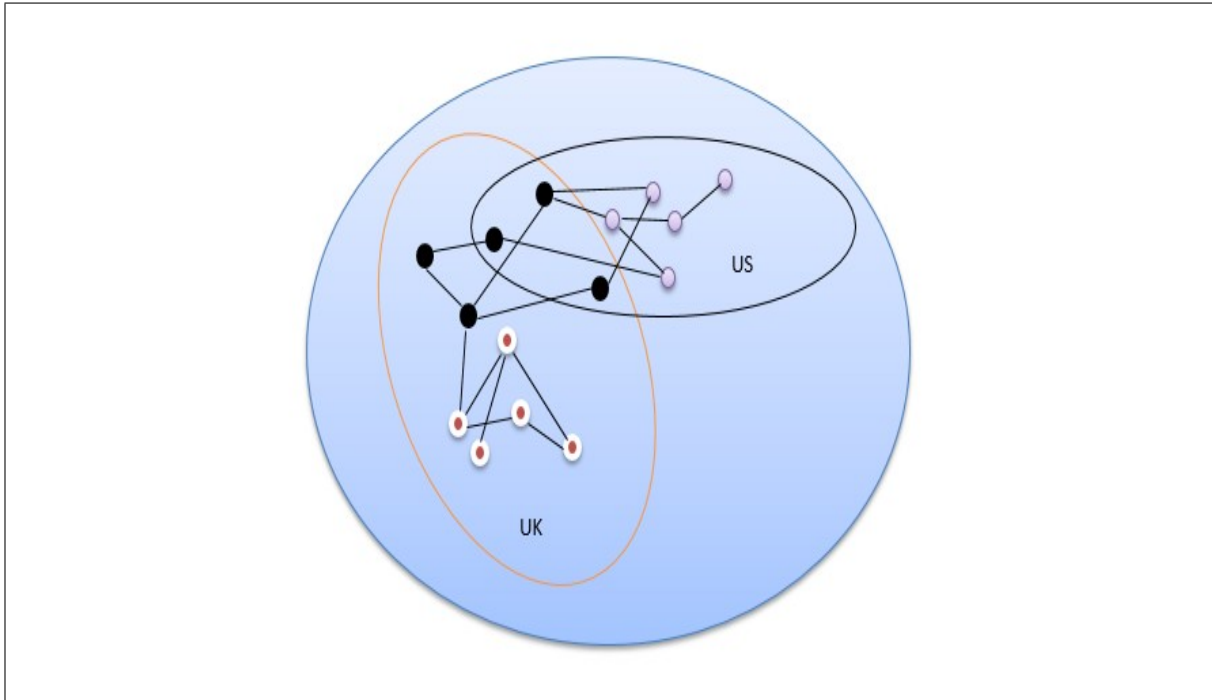


Fig. 3.1 Example of Localized clusters creation. The black circles represent the border nodes between clusters, while grey and red circles represent nodes in different clusters.

Aiming to overcome the above limitations, LBC protocol is implemented in a distributed manner where all nodes contribute in achieving a proximity based network layout. This prevents any node having control over the network as there is no node that would have a full knowledge of the entire network topology. In order to evaluate the impact of maximising the geographical proximity based connections on information propagation in the Bitcoin network, LBC protocol groups Bitcoin peers based on the geographical locality of their IP prefixes. This contributes in minimising the network latency between peers which results in improving the information propagation delay in the Bitcoin network. In LBC protocol, peers' IPs are used as the basis for defining a locality area inside the Bitcoin network. However, LBC protocol is a measurement based and can dynamically change the network layout and connect the geographically closer peers. Therefore, LBC aims to convert the Bitcoin network topology from normal randomised

neighbour (connected nodes) selection to location based neighbor selection. Peers in LBC are self-clustering based on locality, thus every peer must know whether other peers are geographically close to it. This means every peer in the Bitcoin network chooses its neighbour mostly from those within the same geographical location and forms a cluster. Within each cluster, as it is shown in Fig.3.1, peers are highly connected via short-distance links. Giving the visibility into the available information from the outside cluster as well as avoiding the network partitions, clusters are fully connected by their border nodes. Border nodes between two clusters represent the two closest nodes belonging to the two clusters.

### 3.2.1 Localised Cluster Generation

In the LBC protocol, each node runs the protocol independently by information about discovered nodes and local neighbours. In this phase, nodes in the network are grouped into clusters such that the nodes in the same location belong to the same cluster. This can be done by referring an extra function to each node in the Bitcoin network. By this function, each node is responsible for recommending proximity nodes to its neighbours. The proximity is defined based on the physical geographical location. In other words, the proximity relies on a distance threshold which identifies the number of clusters and the size of a cluster. Specifically, when a node discovers new Bitcoin nodes, it calculates the distance between its neighbours and the Bitcoin nodes that it has discovered. Two nodes  $N_i$  and  $N_j$  are considered close to each other if:

$D_{i,j} < D_{th}$  where  $D_{i,j}$  is the distance between  $N_i$  and  $N_j$ ,  $D_{th}$  is the distance threshold. LBC protocol uses the haversine formula to measure the geographical distance (Veness, 2011). Specifically, the haversine formula is used to calculate the real-world distance between two nodes on the Earth's surface specified in longitude and latitude. Therefore, we retrieve from the IP address of the node the latitude and longitude by using MaxMind GeoLite City database (Maxmind, 2013). Haversine is defined as:

$$a = \sin^2\left(\frac{\Delta\phi}{2}\right) + \cos(\phi_1) \cos(\phi_2) \cdot \sin^2\left(\frac{\Delta\lambda}{2}\right) \quad (3.1)$$

where  $\phi$  is latitude,  $\lambda$  is longitude,  $R$  is earth's radius (mean radius = 6,371km (Anderson et al., 2008)). The distance  $d$  in meters is then calculated as:

$$d = 2R \cdot \text{atan2}(\sqrt{a}, \sqrt{1-a}) \quad (3.2)$$

When a node  $N$  discovers another node  $K$  that is close to the  $K$ 's neighbour  $M$ , the node  $N$  sends the discovered node  $K$  to its neighbour node  $M$  as a recommended node to connect

with. On receiving the IP address of the recommended node  $K$ , the node  $M$  should connect to it and try to find out whether or not the recommended node is also close to its neighbours. This scenario is repeated at each node that receives recommended nodes from its neighbours.

As mentioned before, clusters are fully connected by their border nodes. Therefore, border nodes will be selected between every pair of clusters. Border nodes will be selected to be the closest pair of nodes that belong to two clusters. This ensures an efficient information dissemination between clusters as many transmission channels will be available for information to be exchanged among clusters. Furthermore, increasing the number of border nodes between clusters results in maximising the level of difficulty to partition the network (e.g. partitioning attack). More clearly, let  $S = \{s_1, s_2, \dots, s_m\}$  and  $R = \{r_1, r_2, \dots, r_n\}$  represent two clusters, and let  $[s_b, r_b]$  denote their border nodes, where  $s_b \in S$  and  $r_b \in R$ , then for all other pairs of clusters (such that  $s_i \neq s_b, r_j \neq r_b, s_i \in S, r_j \in R$ ),  $distance(s_i, r_j) \geq distance(s_b, r_b)$ . Note that  $distance(x, y)$  represents the geographical distance between the two nodes  $x$  and  $y$  in the network.

### 3.2.2 Cluster Maintenance

The Bitcoin network structure exhibit some degree of churn where peer nodes enter and exit the network at arbitrary points in time. The existing clusters of the nodes in the network are influenced by the dynamics in the Bitcoin network structure. Therefore, a mechanism that handles the node dynamics is required to avoid re-clustering the entire network on each node entry and exit. Re-clustering is impractical when the entry and exit of the nodes are frequent as the clusters never stabilize (Ramaswamy et al., 2005). Though, a mechanism is designed to handle the entry and exit of the nodes in the Bitcoin network.

When a node  $Z$  wants to join the Bitcoin network, it learns about the available Bitcoin nodes from a list of DNS services. Upon receiving a query from the node  $Z$ , DNS services probe the node  $Z$  in order to determine its geographical location through calculating the distance based on the same methodology that is mentioned in Section 3.5.1. Based on the probe results, DNS services check the network and return any known peers close to the node  $Z$ . If none were found, random peers will be returned. If the DNS service is close to the node  $Z$ , it returns all peers that are close to itself. Then, the node  $Z$  measures the distance to each discovered node to get its location ordering based on a distance threshold. The  $Z$ 's location ordering would help the node  $Z$  to be assigned to a specific cluster. After that, the node  $Z$  sends a *JOINrequest* destined for the closest node  $C$  of the discovered nodes. Once the node  $Z$  connects to the node  $C$ , it receives a list of IP's of nodes that belong to the same cluster of the node  $C$  in order to allow the node  $Z$  connects to the nodes that belong to  $C$ 's cluster only. When the node  $Z$  wants to leave the network, no further action is required.



From a security point of view, DNS nodes cannot impose significant security issues, where a newly joined node is forced to connect to attacker nodes, even though they decide to act as malicious nodes. The reason behind that is the newly joined nodes normally learn one peer from Bitcoin DNS nodes, and then nodes can use the normal discovery mechanism of the Bitcoin network to find more nodes to connect with.

### 3.3 Ping Time Based Approach(BCBPT)

In fact, nodes that are geographically close might actually be quite far from each other in the physical internet and vice versa. For instance, hosts directly connected by an optic fiber are most likely very “close” when the proximity only takes into account the link latency between network nodes, even if they are physically placed far away from each other. This actual, physical internet distance may lead to different results, leading to different conclusions too. As proximity can be measured using different criteria, such as the physical location and the link latency among peers (Miers et al., 2010), we propose a proximity based latency awareness protocol with the aim of evaluating the security and performance impact of the proximity of connectivity based ping latencies on the Bitcoin network. Based on round trip ping latencies, nodes can detect and cut most of the inefficient and redundant logical links, and add closer nodes as its direct neighbour. Therefore, peers within each cluster are highly connected via short link latencies. This offers faster information propagation, resulting in a better distribution of Bitcoin information over the network which helps in reaching a consistent state in the Bitcoin network. To maximize the security awareness with respect to network partitions as well as ensure an efficient information distribution between clusters, clusters are fully connected using border nodes. Border nodes are selected using the same strategy of border nodes selection in LBC protocol that is mentioned in section 3.2 with only one difference that the  $distance(x,y)$  represents the distance between the two nodes  $x$  and  $y$  measured by link latencies.

#### 3.3.1 Distance calculation

The distributed algorithm principle is followed in BCBPT where each node runs the protocol independently by information about the proximity of local neighbours and discovered nodes. In this phase, it is necessary to maintain clusters of nodes with less ping latencies. By doing this, proximity in the physical internet would be enhanced, resulting in improving the information propagation delay in the Bitcoin network.

In BCBPT, each node is responsible for gathering proximity knowledge regarding the discovered nodes. When a node discovers new Bitcoin nodes, it calculates the physical Internet

distance between itself and the Bitcoin nodes that it has discovered. The calculation of the physical Internet distance relies on the round-trip latency between two nodes. As it is mentioned earlier, nodes discover other nodes in the Bitcoin network using either the Bitcoin network discovery mechanism or the Bitcoin DNS service. Two nodes  $N_i$  and  $N_j$  are considered close in the physical Internet if

$D_{i,j} < D_{th}$  where  $D_{th}$  is the distance between  $N_i$  and  $N_j$  measured by the round-trip latency,  $D_{th}$  is the latency threshold. We introduce a utility function that could calculate the distance between two nodes in the Bitcoin network measured by latency. This function would dramatically change the behavior of the overlay and help enriching nodes with proximity knowledge. The new utility function is shown in 3.3:

$$D_{i,j} = \frac{M_{ping}}{rate(r)} + 2P + q' \quad (3.3)$$

where  $i$  and  $j$  are two nodes in the network,  $M_{ping}$  is the length of the ping message (Bytes). The term  $rate(r)$  represents the rate of transmission which is the total amount of data that can be sent from one place to another in a given period of time (around  $\approx 100$  KB/hour), while  $P$  refers to the propagation speed which is the amount of time it takes for one particular signal to get from one point to another.  $P$  is multiplied by 2 because of the roundtrip time. The propagation speed is calculated as:

$$P = \frac{D_M}{S} \quad (3.4)$$

The term  $D_M$  denotes the distance between two nodes  $i$  and  $j$ .  $D_M$  can be calculated using the geographical distance calculation methodology that has been used in section 3.2.  $S$  is the speed of the signal which is equal to  $3 \times 10^8 m/s$  when dealing with Wi-Fi internet, while it is equal to  $2/3 \times 3 \times 10^8 m/s$  in terms of copper cable.  $q'$  represents the queuing time(average). Queuing time can be calculated as:

$$q' = \frac{M_{ping}}{rate(r) - \lambda} \times M_{ping} \quad (3.5)$$

where  $\lambda$  represents the arrival rate (How many pings are arriving to the node  $j$ ).

As distance measurements are subject to the network congestion and therefore dynamic, within some variance, multiple messages between pairs of nodes, repeatedly are sent over the time in order to determine variance. In terms of a discovered node being close to another node, the node establishes a connection with the discovered node by sending a version message as a handshake. In contrast, these two nodes would have a very little chance to get directly connected and stay in the same cluster if they are so far away from each other. Therefore,

clusters in the overlay network become more proximity-aware and nodes make a better decision in terms of limiting the cost of communication.

### 3.3.2 Cluster Maintenance

Regarding the cluster creation and maintenance protocol, a node  $N$ , while joining the network for first time, learns about the available Bitcoin nodes from a list of DNS services. However, the node discovery service should also make a ranking on which node to select as the initial DNS seed service might return sub optimal peers. Therefore, DNS service nodes should recommend available nodes to the node  $N$  based on the proximity in the physical geographical location as the geographic distance in the Internet, in many cases, is a good indication of the topological distance. DNS service follows the geographical distance calculation methodology that has been used in LBC protocol, Section 3.2, in order to recommend the closest available nodes to the node  $N$ . The node  $N$  calculates the distance to each discovered node in order to get its proximity ordering based on a link latency threshold. This ordering would help the node  $N$  to be directed to a specific cluster. After that, the node  $N$  tries to connect to a node  $K$  which is the closest node in the nodes list that is supplied by the DNS service. However, the role of the DNS service stops once the node  $N$  connects to the node  $k$ . Periodically, the node  $N$  discovers other nodes in the network using the Bitcoin network nodes discovery mechanism (Heilman et al., 2015). Then, the node  $N$  finds out whether the discovered nodes are physically close by following the distance calculation mechanism that has been mentioned in Section 3.3.1. When the node  $N$  wants to leave the network, in this case no further action is required.

Similar to LBC protocol, Bitcoin DNS service cannot rise a serious security risk as the newly joined nodes normally use the Bitcoin network discovery mechanism after connecting to at least one node supplied by the DNS service.

## 3.4 Super Node Based Approach (BCBSN)

As the number of hops between peers is considered as one of the factors that the proximity between nodes in the peer-to-peer networks is based on (Miers et al., 2010), such an approach that utilize the idea of super peers technology can contribute in minimizing the intermediate hops between peers. As the Bitcoin network is a financial instrument which needs to be resilient against active attacks, the super peer approach introduced in this work enhances, in different ways, previous super peer solutions (Mizrak et al., 2003) (Yang & Garcia-Molina, 2003) with respect to security awareness. First, it does not require any network node to have full knowledge of the entire network topology. It is therefore adequate for supporting the

decentralized concept of Bitcoin. Second, super peers are selected based on achieving several conditions in a distributed manner. Therefore, impersonating a super peer role by a malicious node is a challenging task to achieve. In such a super-peer approach, the design of the overlay network is composed of several clusters of peers. It selects a peer to be a super-peer and this super-peer becomes a cluster head that propagates network information to other super peers in different clusters. In addition, super-peers can be given an extra functions to be handled in the network, such as grouping peers based on a specific criteria. If we can group peers according to their geographical proximity, we can further speed up information broadcasting in the network. Thus, a hierarchical Bitcoin overlay network that clusters nearby peers might achieve faster information propagation than the original Bitcoin system. In this work, the concept of super peers is applied on the Bitcoin network in order to gain the capability of optimizing proximity of connectivity between peers taking into account their geographical location.

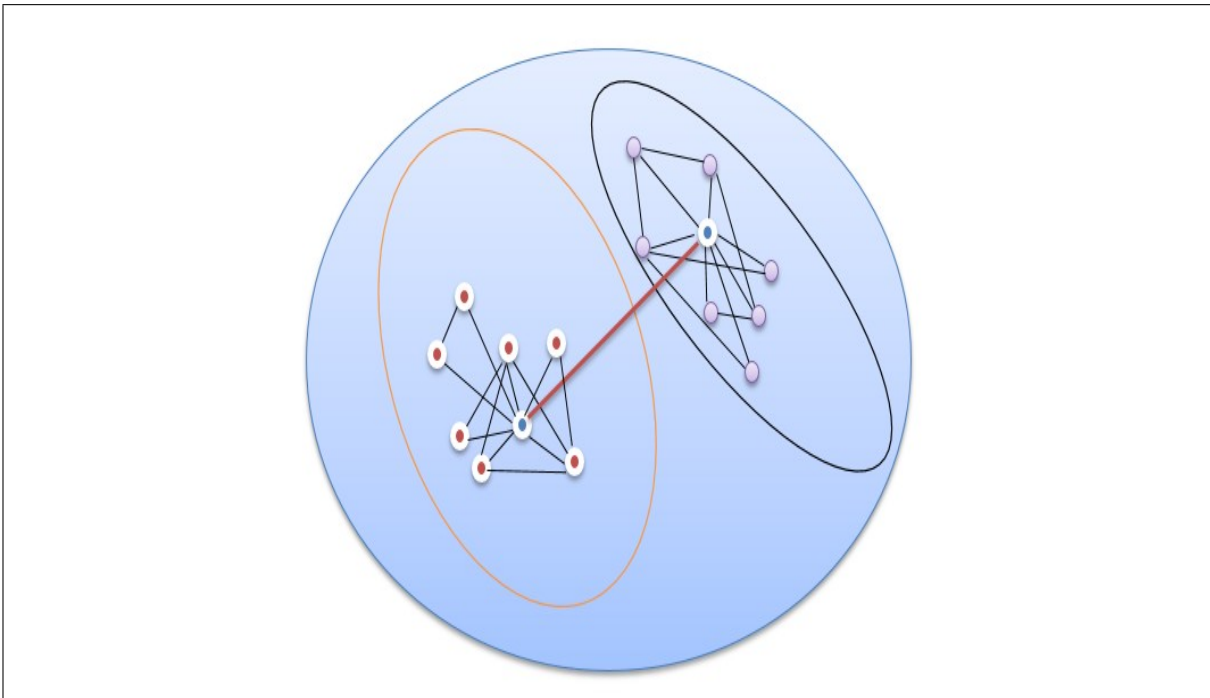


Fig. 3.2 Example of super peer clusters creation. The Blue circles represents super peers in each cluster, while grey and red circles represent nodes in different clusters.

A cluster generation protocol is introduced, named as Bitcoin Clustering Based Super Node (BCBSN), as a way to combine the reduction of the intermediate hops between any two peers as well as increasing the geographical connectivity between peers. BCBSN protocol aims to generate a set of geographically diverse clusters in the Bitcoin network by exploiting super peers technology. Within each cluster, BCBSN protocol assigns one node to be a super peer that is responsible for maintaining the cluster as well as broadcasting information in the

Bitcoin network. However, this is the first work to address the clustering based super-peer technology in the Bitcoin network. It is claimed that applying BCBSN protocol, where each peer connects to one super peer and each super peer connects to other super peers, would reduce the non-compulsory hops that the transaction passes through, so that, on the other hand, this may minimize the propagation delay. Specifically, as it is shown in Fig. 3.2, several nodes are selected to be as a coordinator of clusters (super peer). Each super peer connects to the geographically closest nodes and forms a cluster. All super peers in the network are fully connected and known to each other. BCBSN protocol constructs super peer clusters using two proposed algorithms: the peer joining algorithm and super-peer selection algorithm. These algorithms allow peers to be destined to a cluster where all peers are close in the geographical location.

### 3.4.1 Super peer selection algorithm

Due to security requirements, super peer selection algorithm in BCBSN is differing on the criterion for the selection of the super nodes. For instance, in (Lin & Gerla, 1997), super peer selection is based on the unique identifier ID of each node. In other words, the node which has a lowest ID is selected as a super peer. Whereas, super peer position has several conditions need to be fulfilled in BCBSN protocol. Precisely, super peer selection is based on handling a reputation protocol. This protocol is achieved by referring a weight defined by a positive real number for each node. The weight is calculated based on how much bitcoins are spent by each node and how long each node has been online. A node with a bigger weight is selected as a super peer. The main advantage of this approach is that, impersonation of a super peer by a malicious node would be challenging as well as the super peer's role would be occupied by those nodes that are better suited for that role. According to the rewarding mechanism that has been proposed in (Babaioff et al., 2012), it has been proven that applying such a rewarding schema would incentivize information propagation in the Bitcoin network. Therefore, a reward is given in the BCBSN protocol for a super peer when it propagates a valid transaction and behaves honestly. This is considered as an incentive mechanism for nodes to win the super peer's role. In the case of a super peer goes offline, each cluster selects a backup peer which copies the entire cluster state information periodically from the super peer. The backup peer is selected based on the same mechanism and criteria of super peer selection. However, BCBSN protocol requires some changes on the real Bitcoin network propagation protocol. Consider a number of nodes grouped by locality into two clusters, A and B. Each cluster has elected a node to act as a super peer. A node  $n$  within cluster A wishing to initiate a transaction and broadcast it to its connected nodes. On receiving the transaction at a super peer S in cluster A,

it is re-broadcasted to other super peer  $S_1$  in cluster B as well as  $S$ 's connected nodes within cluster A.  $S_1$  also rebroadcasts the transaction to its connected nodes within cluster B.

As one of the parameters of calculating a weight for each node is to identify how long a node has been online, the super peer selection algorithm handles a penalty score mechanism by which the stability of nodes will be measured. In this mechanism, each node maintains a penalty score for every node connected to it. Every time a node goes offline, its penalty score is increased by 1 by its connected nodes. After that, a super peer should receive the updated score from those nodes that increased the score. Once the super peer's record is updated, the super peer sends the updated score to all of its connections.

Regarding super peer selection, two types of messages are propagated, a *SupINV* and an *AcceptINV*. Once a node  $k$  decides to be a super peer, it invites its connected nodes by sending a *SupINV* which includes the node's ID and weight. On receiving of *SupINV* message at the node  $M$ , as illustrated in Algorithm 1, the node  $M$  accepts  $k$ 's invitation if it finds the node  $k$  to be geographically closer and has a bigger weight than the super peer that  $M$  is connected to. The node  $M$  decides whether or not  $k$  is geographically close to it by calculating the geographical distance using the same methodology that have been followed in LBC protocol. Section 3.2. The node  $M$  accepts  $k$ 's invitation by sending an *AcceptINV*. The node  $M$  should keep forwarding the *SupINV* to all its connected nodes which in turn will propagate the *SupINV* further.

---

**Algorithm 1:** SupINV handler function

---

```

Let  $k$  as :nearest superpeer() with Bigger weight
Let  $sp$  as :current superpeer
1 if  $sp \neq k$  then
2    $sp = k$ 
3   connectTo ( $k$ )
4   Forward (SupINV)
5 else
6   Forward (SupINV)
7 end

```

---

### 3.4.2 The peer joining algorithm

Turning now to the second phase of the BCBSN protocol which is the cluster maintenance protocol where the entry and exit of nodes in the Bitcoin network are handled. However, notations are delivered to be used in this phase before presenting the peer joining algorithm.

Let  $M = \{k_0, k_1, \dots, k_{i-1}\}$  be a set of peers in the Bitcoin network, where  $i$  is the number of total peers. Let  $P = \{sp_0, sp_1, \dots, sp_{j-1}\}$  be a set of super peers, where  $j$  is the number of super peers and  $P \subseteq M$ . Let  $sp_l \{sp_l, b_0, b_1, \dots, b_{n-1}\}$ , where,  $l$  is ranged over  $(0, 1, \dots, j-1)$  and similarly in the case of  $n$  which reflects the set of peers in the network. Therefore, we have  $sp_l \subseteq M$  and  $M = sp_0 \cup sp_1 \cup \dots \cup sp_{j-1}$ . When a node  $z$  wants to join the Bitcoin network, it first learns about the available super nodes by contacting an arbitrary node  $k$  which already have been learnt from the DNS service. The node  $k$  responds with a list of the super peers it knows about in the network. According to the peer joining algorithm 2, the node  $z$  selects a super peer  $sp_s$ , such that  $\forall q \in P, \text{distance}(z, sp_s) \leq \text{distance}(z, q)$ . Then, the node  $z$  sends a *JoiningRequest* message to the selected super peer. Note that  $\text{distance}(x, y)$  represents the geographical distance between the two nodes  $x$  and  $y$  in the network which can be calculated by following the method that has been used in LBC protocol, in Section 3.2. The super peer  $sp_s$  responds with an *Acceptance message* which includes a list of nodes addresses that the node  $sp_s$  connects with in order to allow the node  $z$  connects to the nodes that belong to  $sp_s$  cluster only. When the node  $z$  wants to leave the network, it sends a disconnect message to its super peer, which requires no reply. Then, the node  $sp_s$  should update its nodes list automatically. Once the node  $z$  joins the Bitcoin network, it sends metadata over its connections to its super peer, at the same time the super peer adds the node  $z$  to its index.

---

**Algorithm 2:** Peer joining algorithm

---

```

Let  $P$  as : Super peers set
Let  $z$  as : new peer to join the network
1 while  $P \neq 0$  do
2    $d \leftarrow \text{distance}(z, sp_i)$  where  $\forall sp_i \in P$ 
3    $d_1 \leftarrow \text{distance}(z, sp_j)$  where  $\forall sp_j \in P$ 
4   if  $d < d_1$  then
5      $z \leftarrow \text{connectTo } sp_i$ 
6   else
7      $z \leftarrow \text{connectTo } sp_j$ 
8   end
9 end

```

---



### 3.5 Master Node Based Clustering (MNBC)

Master Node Based Clustering protocol (MNBC) extends the BCBSN protocol that was proposed in Section 3.4, with the aim of addressing security and performance limitations of BCBSN protocol. As it is mentioned in section 3.4, the BCBSN protocol aims to generate a set of geographically diverse clusters in the Bitcoin network by exploiting super peers technology. Within each cluster, the BCBSN protocol assigns one node to be a super peer that is responsible for maintaining the cluster and broadcasting information in the Bitcoin network. In the BCBSN protocol, clusters are fully connected via super peers only. Due to this, the information flow between clusters in the BCBSN protocol is only fulfilled through super peers. Furthermore, super peers in the BCBSN protocol group peers based on their geographical location in order to increase the proximity of connectivity in the network. However, as it is mentioned in section 3.3, long-link distance might be applied between any two peers even though they are in the same geographical location. From a security point of view, the level of security awareness in the BCBSN protocol can be improved if more nodes between clusters are maintained. This reduces the probability of the network partition occurrence.

The limitations of BCBSN protocol mentioned above have motivated the development of a new protocol that overcomes the lack of connection channels between clusters as well as considers the physical internet distance rather than the geographical location. Specifically, the new protocol, named as Master Node Based Clustering(MNBC), relies on several nodes, known as master nodes, to achieve fully connected clusters based on the physical Internet proximity, where information can be exchanged between clusters via master nodes as well as normal nodes. The idea of the MNBC protocol is inspired by the *Master node technology* that was originally adopted in Darkcoin (Duffield et al., 2014). However, master nodes in Darkcoin were responsible only for propagating the network information to the majority of nodes without taking into account whether or not those nodes are close. Also selecting master nodes in Darkcoin does not require conditions to be fulfilled with the aim of ensuring security. Whereas master nodes in the MNBC protocol connect to other nodes based on a proximity criteria. Furthermore, master nodes in the MNBC protocol are selected through applying a selection phase that requires several conditions need to be fulfilled in order to cover the role of master nodes.

As it is shown in Fig.3.3, clusters in MNBC are fully connected via master nodes. Giving the possibility of a better improvement in information propagation and security awareness, clusters are also connected by several nodes, known as edge nodes, that represent the closest nodes belonging to different clusters. Master nodes are normal Bitcoin full nodes that can offer a level of additional functions as follows. Creating a set of clusters in the Bitcoin network where each cluster includes nodes that are close to each other in the physical network. Furthermore,



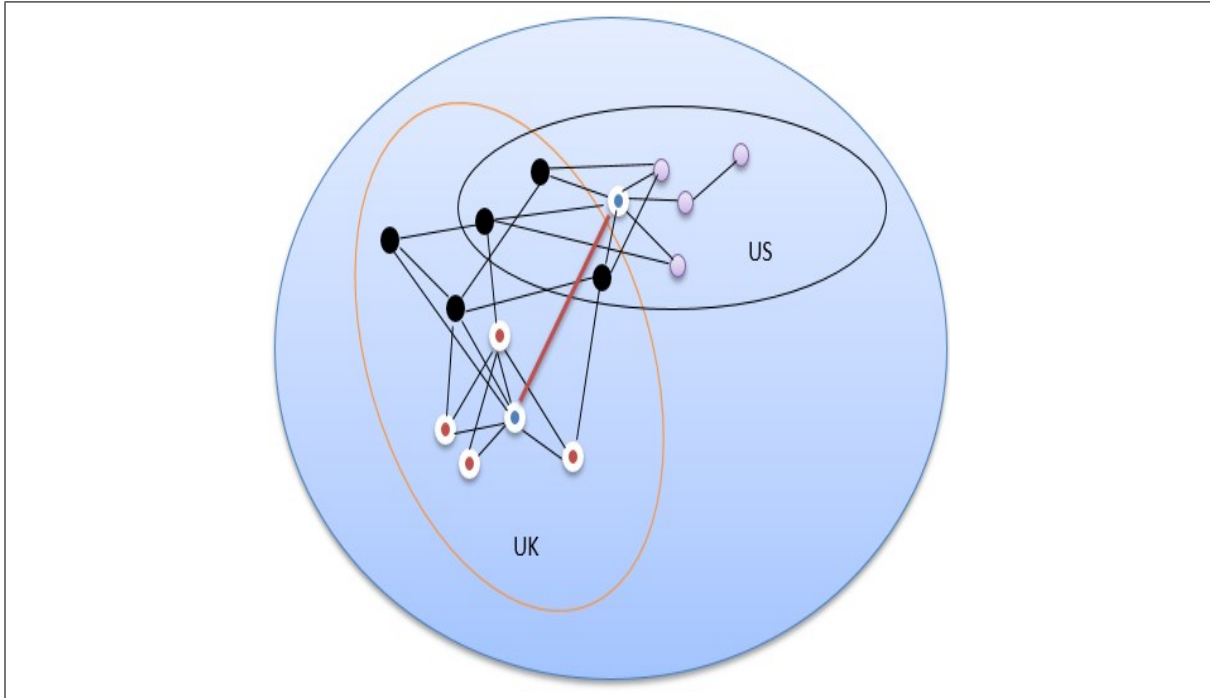


Fig. 3.3 Example of master node clusters creation. The black circles represent the edge nodes between clusters, while grey and red circles represent nodes in different clusters. Blue circles represents master nodes in each cluster.

supporting a propagation scenario, by which messages are propagated to a list of all known master nodes across the network as well as nodes that belong to the master node's cluster. In addition, information can be propagated to outside a cluster by edge nodes that are connected to other nodes in different clusters.

### 3.5.1 Master Node Selection

Master node selection follows the same selection criteria of super peers that are mentioned in BCBSN protocol. Specifically, master node role requires gaining a score which is calculated based on how much each node burns bitcoins and how long a node has been online. The main advantage of this approach is that, impersonation of a master node by a malicious node would be challenging. Therefore, this score helps in electing master nodes that are better suited for that role. To incentives nodes to compete towards winning the master node's role, as it has proven in (Babaioff et al., 2012), a reward is given for a master node when it propagates a valid transaction and behaves honestly. When a particular node achieves the best score over other nodes in the network, as it is illustrated in Algorithm 3, the node would be elected as a master node.

When a particular peer wants to occupy the role of master nodes, the peer invites other peers that connect to it by propagating two types of messages a *masterINV* and an *AcceptINV*. Consider a node  $M$  decides to be a master node and a peer  $P$  receives a *masterINV* from  $M$ , a modified version of the super peer selection algorithm that is mentioned in section 3.4.1 is used. The modification includes considering link latencies in the calculation of the distance between peers and master nodes, rather than relying on the geographical location. Specifically, the node  $P$  accepts  $M$ 's invitation if it finds the node  $M$  to be closer in the physical internet and has a bigger weight than the master node that the node  $P$  is connected to. The node  $P$  decides whether or not the node  $M$  is close in the physical internet by calculating the internet distance using the same methodology that has been followed in in BCBPT protocol, Section 3.3. The node  $P$  accepts  $M$ 's invitation by sending an *AcceptINV*. The node  $P$  should keep forwarding the *masterINV* to all its connected nodes which in turn will propagate the *masterINV* further.

---

**Algorithm 3:** Master node score calculation algorithm

---

**Let  $M$  as :** Master nodes set in the network  
**Let  $z$  as :** Best master node score to achieve

```

1 while  $M \neq 0$  do
2   for master node in  $M$  do
3      $n \leftarrow \text{masternode.CalculateScore}()$ 
4     if  $n > z$  then
5        $z = n$ 
6       winning – node  $\leftarrow \text{masternode}$ 
7     end
8   end
9 end

```

---

### 3.5.2 Cluster Maintenance

Regarding new peers joining the network, the peer joining algorithm that is mentioned in section 3.4.2 is modified in order to be used in the MNBC protocol. The modification lies in the area of calculating the distance between a master node and the newly joined peer. Let  $R\{n_0, n_1, \dots, n_{i-1}\}$  be a set of peers in the Bitcoin network, where  $i$  is the number of total peers. Let  $M\{mp_0, mp_1, \dots, mp_{j-1}\}$  be a set of master nodes, where  $j$  is the number of master nodes and  $M \subseteq R$ . Let  $mp_l\{mp_l, b_0, b_1, \dots, b_{k-1}\}$ , ( $l = 0, 1, \dots, j-1$ ) and  $k$  is the number of peers in the cluster,  $mp_l$  be a set of peers in the  $l$ th cluster. Therefore, we have  $mp_l \subseteq R$  and  $R = mp_0 \cup mp_1 \cup \dots \cup mp_{j-1}$ . When a node  $z$  wants to join the Bitcoin network, it first learns about the available master nodes by contacting an arbitrary node  $T$  which already have been

learnt from DNS service. The node  $T$  responds with a list of the master nodes it knows about in the network. The node  $z$  selects a master node  $mp_i$  such that  $\forall mp_j \in M, distance(z, mp_i) \leq distance(z, mp_j)$ . Then, the node  $z$  sends a *JoiningRequest* message to the selected master node. Note that the distance is also calculated based on the link latency through following the same methodology that has been adopted in the BCBPT protocol, Section 3.3.1. The master node  $mp_i$  responds with an *Acceptancemessage* which includes a list of nodes addresses that the node  $mp_i$  connects with in order to allow the node  $z$  to connect to the nodes that belong to  $mp_i$ 's cluster. When the node  $z$  wants to leave the network, it sends a disconnect message to its master node, which requires no reply. Then, the master node  $mp_i$  should update its nodes list automatically.

As it is mentioned before, clusters are fully connected by their edge nodes and master nodes with the aim of improving the security and performance of the MNBC protocol. Therefore, edge nodes will be selected between every pair of clusters. Edge nodes is selected to be the closest pair of nodes in the physical internet that belong to two clusters. Edge nodes are selected using the same strategy of border nodes selection that is mentioned in the LBC protocol, Section 3.2, with only one difference that the  $distance(x, y)$  represents the distance between the two nodes  $x$  and  $y$  measured by link latency.

### 3.6 Conclusion

This chapter introduces four approaches, namely LBC, BCBPT, BCBSN, and MNBC, in relation to the proximity based clustering in the Bitcoin network. These approaches consider increasing the proximity of connectivity in the Bitcoin network as a mechanism to speed up information propagation with the aim of tackling the problem of inconsistencies in the Bitcoin network that is caused by the network latency. These approaches are discussed based on how clusters are formulated and nodes define their membership. LBC increases the proximity of connectivity in the Bitcoin network by supporting geographical proximity based connections among nodes. Whereas, BCBPT optimizes the proximity of the overlay topology by creating distinct, but connected clusters of peers with P2P latencies under a given intra-cluster threshold. On the other hand, BCBSN approach combines the reduction of the intermediate hops between any two peers, using super peer technology, as well as increases the proximity of connectivity in the Bitcoin network based on the geographical distance between peers. While MNBC approach incorporates master node technology and proximity-awareness into the existing Bitcoin protocol with the aim of creating fully connected clusters based on physical Internet proximity. This chapter also highlights typical advantages of the proposed approaches based on improving the proximity of connectivity in the Bitcoin network without compromising security. All of these

approaches are evaluated based on the security and performance in experimental studies in Chapter 5. The evaluation of these approaches is carried out using a simulation model which is developed in this thesis. In the following Chapter, the development and validation of the simulation model as well as measurements of the Bitcoin network are presented.

## Chapter 4

# Bitcoin Network Measurements and Simulation Model

### 4.1 Introduction

In Chapter 3, the several approaches, namely BCBPT, LBC, BCBSN, and MNBC, are proposed to tackle the problem of the agreement on the common transactions history in the Bitcoin network through speeding up information propagation. To evaluate the security and performance of these clustering approaches, major changes are required to the Bitcoin protocol which would have to be accepted by the Bitcoin community. Therefore, this chapter introduces a Bitcoin simulation model which is an event-based simulation framework dedicated to the simulation of the Bitcoin network. In addition, measurements of the transaction propagation delay are presented in this chapter to validate the developed model against the real Bitcoin network. Moreover, large scale measurements of the real Bitcoin network are performed in this chapter in order to enable a precise parameterisation of the presented simulation model. Therefore, the third contribution mentioned in Section 1.8 is fulfilled in this Chapter.

### 4.2 Bitcoin Network Simulation Model

In this section, the simulation model as well as its parameterisation and validation are discussed. However, previous models of Bitcoin have been built to perform studies of the Bitcoin network based on measuring transaction propagation delay as well as evaluating the feasibility of some attacks in the Bitcoin network. A new shadow plug-in model was presented by [Miller & Jansen \(2015\)](#) which has been implemented to run the Bitcoin reference client software for experimental purposes. This model uses a simulation environment to execute the Tor application

that supports Bitcoin using a plugin. However, this simulator offers no support for gathering statistics. In addition, it does not allow full scale experiments as each Bitcoin client has to execute all the expensive blockchain interactions and cryptographic operations of the Bitcoin network. A new Bitcoin model was proposed by [Neudecker et al. \(2015\)](#) in which the core segments of Bitcoin client (*bitcoind*) was transformed to a simulation model. In this model, all the computationally expensive cryptographic operations were abstracted in the client code. Despite this, the difficulty of modeling the complex peer interactions in the proposed clustering approaches makes this model unsuitable choice for achieving our intended purposes due to fact that this model cannot be modified.

The aforementioned limitations of the previously developed simulators encouraged the development of a new model of Bitcoin that meets the requirements of this research. Therefore, a new model of Bitcoin is developed in this work. The developed simulation model is an event-based simulation which is created based on the Bitcoin protocol specification and measurements of the real Bitcoin network. The main purpose of our developed model is to evaluate the proposed clustering protocols as a mechanism to perform faster transaction propagation without compromising security. As the simulation model should behave as closely to the real Bitcoin network as possible, integration of the Bitcoin protocol based on the Bitcoin client behavior(*bitcoind*) as well as measurements of conditions in the real Bitcoin network are modelled. As information propagation delays are the main aspect that this thesis concentrate on, an accurate estimation of the link latencies between Bitcoin network peers is required. Therefore, measurements of the link latencies between peers in the real Bitcoin network are fixed in the designed simulation model. Additionally, the distribution of peers' session lengths in the real Bitcoin network is performed and attached to our model as session length is considered as an essential network parameter that determines the network topology and the stability of the network. The expensive cryptographic operations are abstracted in the proposed model as this research focuses on information propagation delays in the Bitcoin network. This allows full scale experiments of the Bitcoin network. In the following subsection, measurements of important parameters of the Bitcoin network are performed. In addition, the Bitcoin simulation model is presented along with its validation.

#### **4.2.1 Bitcoin Network measurements and Parameterisation**

Real Bitcoin network parameters are measured in this section. These measurements are important to parameterise the presented model accurately. Distribution of the most influential parameters that have a direct impact on the Bitcoin client behavior and information propagation in the real Bitcoin network, will be presented in the following subsections. These parameters have a direct impact on information propagation in the Bitcoin network which is considered as

the main focus in this research. These parameters include peer's session lengths, link latencies between peers, and the number of the reachable nodes.

#### 4.2.1.1 Session Length

In P2P networks, including the Bitcoin network, thousands of peers arrive and departure independently which causes a collective effect known as *churn* (Xu, 2013) (Delgado-Segura et al., 2018). This dynamics of peers participation needs to be taken into consideration in the development of any model of P2P networks. The main reason behind that is to take into account the main effect of the session length on the network stability and overall structure (Stutzbach et al., 2008), the resiliency of the overlay (Leonard et al., 2005), and how the key design parameters are selected (Li et al., 2005). Furthermore, adopting an accurate model of churn results in accurate conclusions about P2P networks (Stutzbach & Rejaie, 2006). However, characterizing the churn of any P2P system requires unbiased information regarding the arrival and departure of peers (session lengths). Therefore, accurate measurements of session lengths in the Bitcoin network are performed in this work.

To start with measurements of peers' session lengths, a Bitcoin client is implemented in this thesis. This client is used to crawl the entire Bitcoin network through establishing connections to all reachable peers in the network. Periodically, the client attempts to discover Bitcoin network peers with the aim of maintaining connections to majority of them. This is done by sending an *Addr message* to the client's neighbours. By getting a list of IPs from neighbours, the client starts connecting to each IP in the received list of IPs. As crawlers require time to capture a complete snapshot that accurately reflects the topological properties and dynamics of unstructured P2P networks (Stutzbach et al., 2008), the developed client crawls the Bitcoin network over a week. During this week, snapshots of IP addresses of reachable peers were published every 3 hours to avoid a situation where the captured snapshots become more distorted due to a gap between consecutive snapshots. By using the data which was gathered by running the developed crawler for one week, points in time in which peers left or joined the network were indicated.

The captured snapshots require to be continuous(in time) over a reasonable period of time to avoid a challenge appears in the data collection for studying churn. More clearly, an incidence that might happen during snapshots gathering, such as losing the network connectivity or the observation software crashes, results in a gap in the overall gathering time. During this gap, an important data will be missing. To overcome the challenge of data missing, measurements are composed by series of snapshots maintained by the crawler, each snapshot includes the start time of the crawl. Therefore, it has been possible to identify whether or not some data got missing through examining the series of times in which snapshots started to be captured.

By doing this, it has been discovered that significant gaps in the collected data have not been experienced.

The distributions of session length in the real Bitcoin network are shown in Fig.4.1. Even though the distributions of session length reveal a considerable churn in the data, 1,400 peers did not leave the network during the observation time. Taking these distributions into account, the stability of the network fluctuates. This might lead to change the topology substantially.

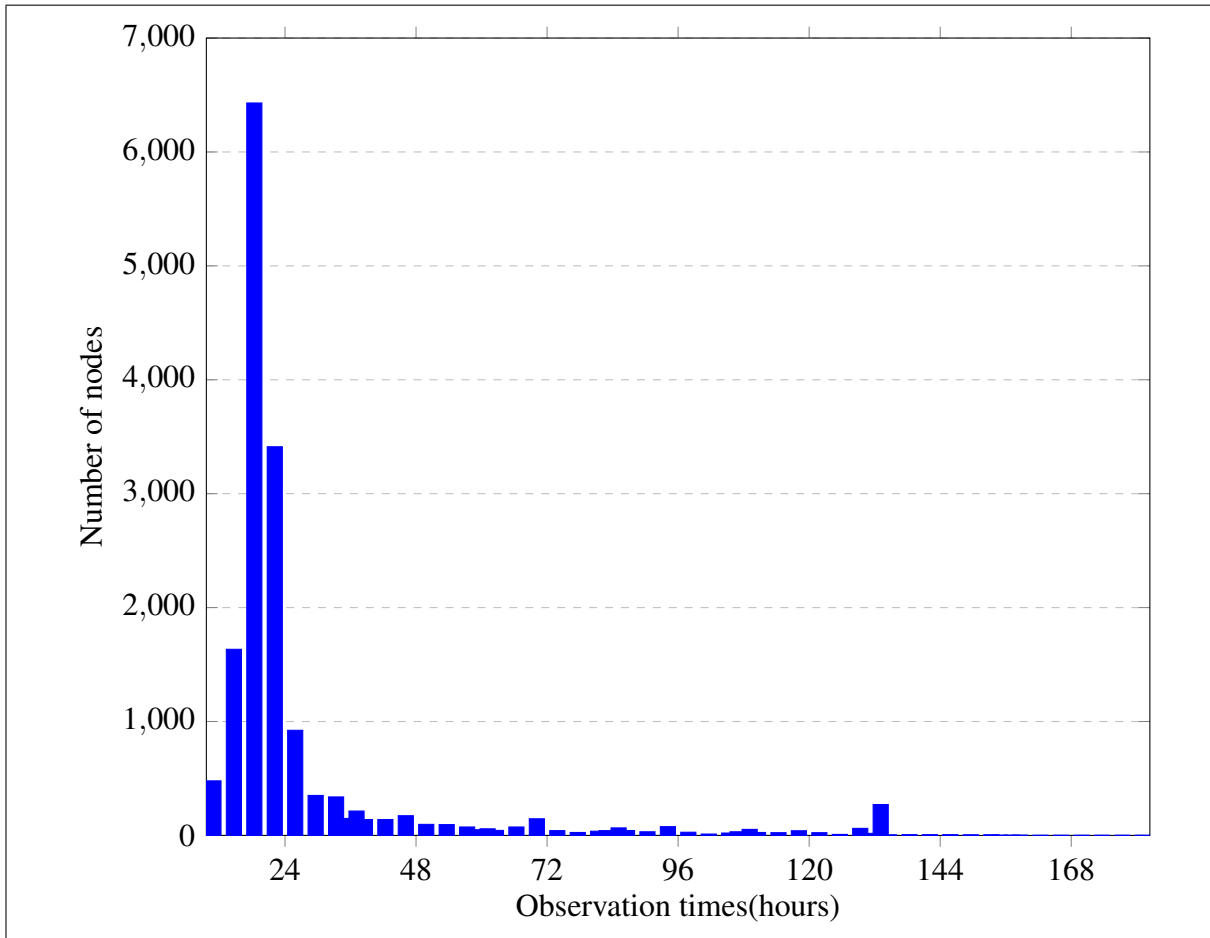


Fig. 4.1 Session lengths of peers in the Bitcoin network

#### 4.2.1.2 Link Latencies

Measurements of the network latency between peers in the Internet play a significant role in the development of any P2P network model as these measurements control the accuracy of conclusions produced by network models (Neudecker et al., 2015). Thereby, the quality of the developed model relies on the Bitcoin network latency information that require acquisition of a large scale measurements to be provided at each node. On top of that, the aim of our research



that lies in the area of information propagation in the Bitcoin network, makes the measurements of link latencies between peers as a considerable requirement to be performed in the developed model.

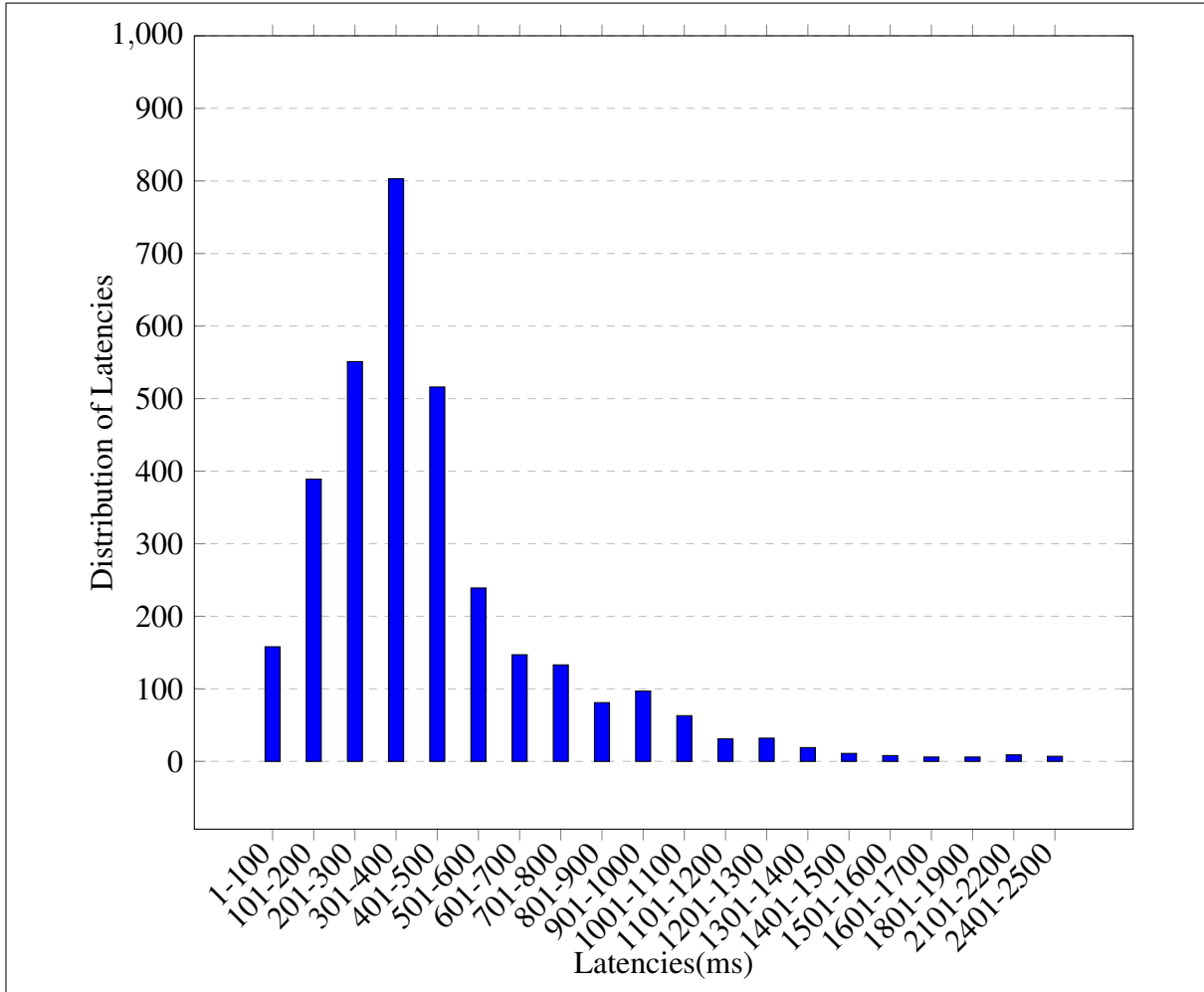


Fig. 4.2 Link latencies between the measurement node (located in Portsmouth/UK) and other peers in the Bitcoin network

In this work, measurements of link latencies between peers were collected by setting up a Bitcoin client that crawls the entire Bitcoin network. Specifically, the developed client utilizes a list of IP addresses that can be obtained by following the Bitcoin network discovery mechanism mentioned in Chapter 2, to connect to the majority of peers in the network. Also, the client considers the advantage of ping/pong messages to measure the round trip latency between the discovered peers and developed client. More precisely, the client attempts to maintain connections to several peers. After that, the client begins an iterative process of sending ping messages to each peer of the connected peers. The link latency between the client and a particular connected peer is calculated when the client hears back from the peer (receiving a

pong message). Specifically, the link latency is measured by calculating the time difference between sending a ping message to the peer, and receiving a pong message by the client. In order to maintain large scale and distributed measurements, the client periodically scan the network and apply the same scenario of measuring the link latencies.

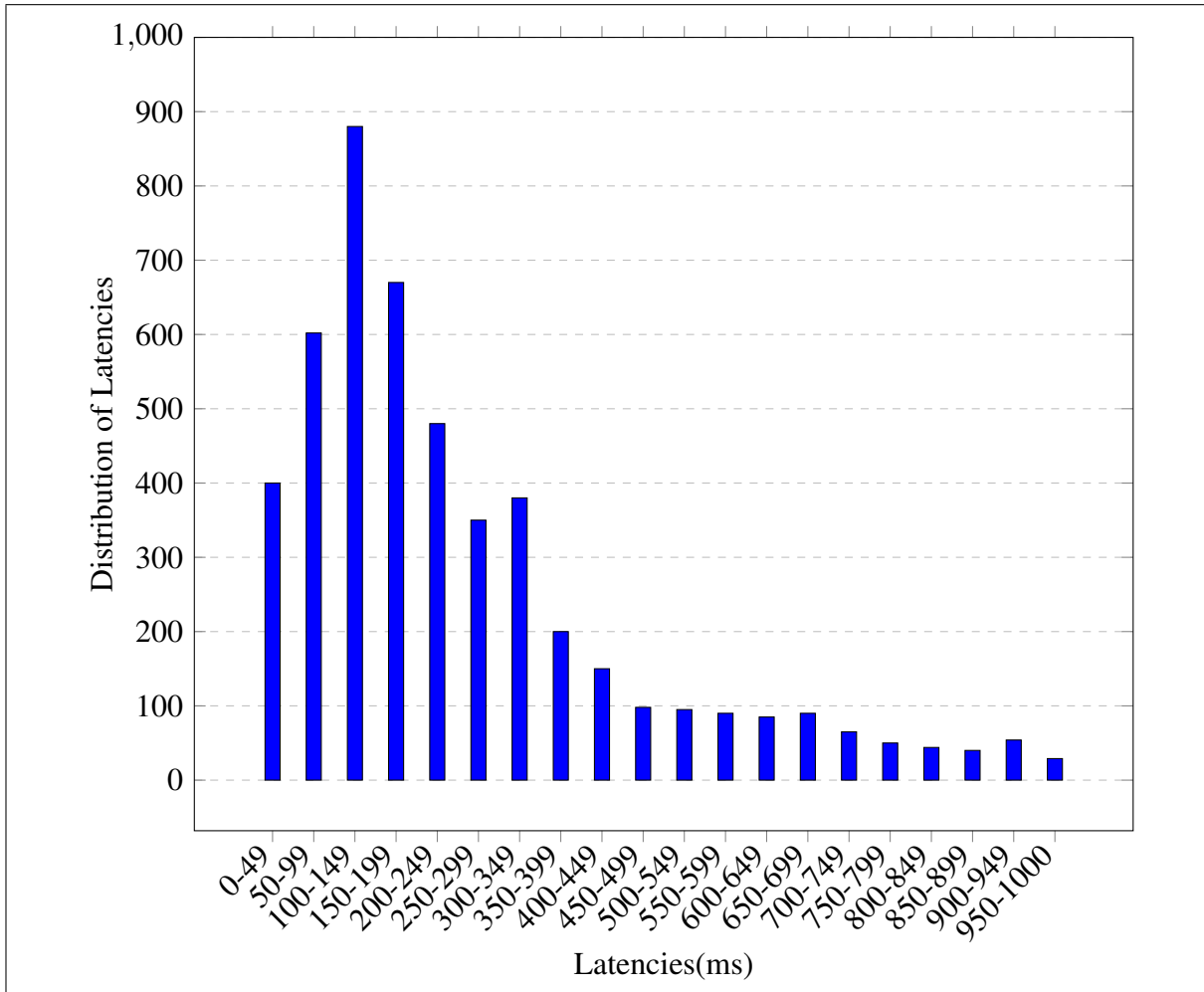


Fig. 4.3 Link latencies between the measurement node(located in Los Angeles/US) and other peers In the Bitcoin network

The distribution of latencies between the developed client that was located in Portsmouth/UK, and peers in the real Bitcoin network is shown in Fig.4.2. These distributions were collected by running the developed crawler which was connected to a round 7000 network peers and observing a total of 27,000 ping/pong messages. The distribution of latencies reveals that around 75% of the collected latencies are below 800 milliseconds, while 25% of distributions are over 1000 milliseconds and reach up to 2500 milliseconds. It should be taken into account that these measured distributions indicate the latency between the developed crawler and other

peers in the network. However, the obtained link latencies reflect an empirical distribution that is close to the normal distributions.

Although the link latency between two peers relies on the location of the host from which the latency is measured, similar distribution of latencies over the entire peers might be obtained from two different given hosts, each host in a different location. To prove that, the developed crawler was run in a different location. Fig.4.3 shows the distribution of the round-trip latencies between peers that are collected by running the developed crawler in Los Angeles/US. It can be seen that the distributions in Fig.4.3 are relatively similar (not identical) to the previous distributions in Fig.4.2. However, attaching the obtained link latencies distribution to the developed simulation model would give an accurate estimate of the time delay that is taken by a transaction to reach different peers in the network.

#### 4.2.1.3 The size of the Bitcoin network

As the developed model simulates the information propagation in the Bitcoin network, the size of the network matters due to the fact that the number of nodes has a direct impact on the range of propagation delays. Therefore, attaching an accurate measurement of the number of nodes in the network to the developed model assists in drawing appropriate conclusions from it.

The size of the Bitcoin network was measured in this work by using the same developed crawler in the section 4.2.1.1. The crawler was able to measure the size of the network by discovering the available IPs in the network and try to connect to them. Presently, the size of the Bitcoin network is around 8,000 nodes as the crawler learned 313676 IPs but was only able to connect to 7,834 peers.

### 4.2.2 Bitcoin Model Structure & Validation

After gathering measurements of the most influential parameters in the Bitcoin network, these measurements are used to accurately parameterise the presented Bitcoin model. The presented model is a lightweight, event-based simulation which is abstracted from cryptography aspects of Bitcoin. Instead, it focuses on the Bitcoin overlay network and transaction round-trip time delay. The simulation model is developed in Java for object oriented structure and modularity. Based on the concept of discrete event simulation, the behavior of the Bitcoin client is modelled as an ordered sequence of well-defined events. These events, which take place at discrete points in simulation time, comprise a specific change in the system's state. Typically, the programmer can abstract many physical level details while modelling objects using discrete event modelling (Kesaraju & Ciarallo, 2012).

In the developed discrete event simulator, two notions of time will be taken into account, simulation time and run time. Simulation time reflects the virtual time or logical time in the simulation world, whereas the run time refers to the time of processor that is consumed by a particular thread. However, simulation time has a direct impact on how the simulation events are organised and accurate results are gained. Specifically, when an event  $E_1$  is executed by a thread A, as shown in Fig.4.4,  $E_1$  should schedule another event  $E_{1,Return}$  which represents a successful return from  $E_1$ . The  $E_{1,Return}$  must be scheduled at a specific point in the simulation time which is calculated after adding an appropriate delay. This delay is collected from the time distributions that have been attached to the model and measured in Section 4.2.1.2. During the time between  $E_1$  and  $E_{1,Return}$ , the simulator can execute any number of events of the same or another client.

The simulator centers around a priority queue that includes all events which are ranked based on its expected time of schedule (ETS) (See Fig. 4.5). ETS is calculated and referred to each event based on the time distributions which are measured in the real Bitcoin network and attached to the simulator. Based on ETS, the foremost event will be scheduled and removed from the queue. An individual node behavior such as joining or leaving the network, create transactions, forward transaction, is implemented by inheriting from given generic java classes.

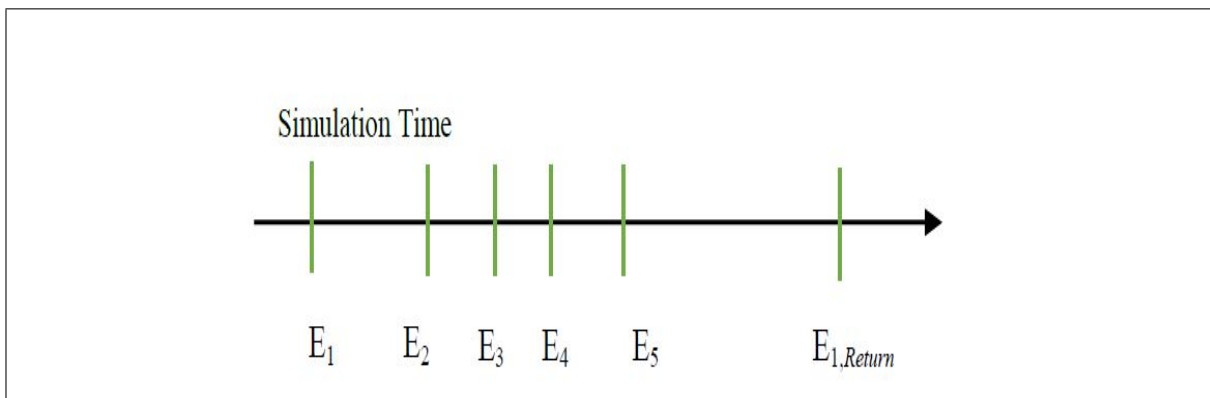


Fig. 4.4 Bitcoin event based simulation representation

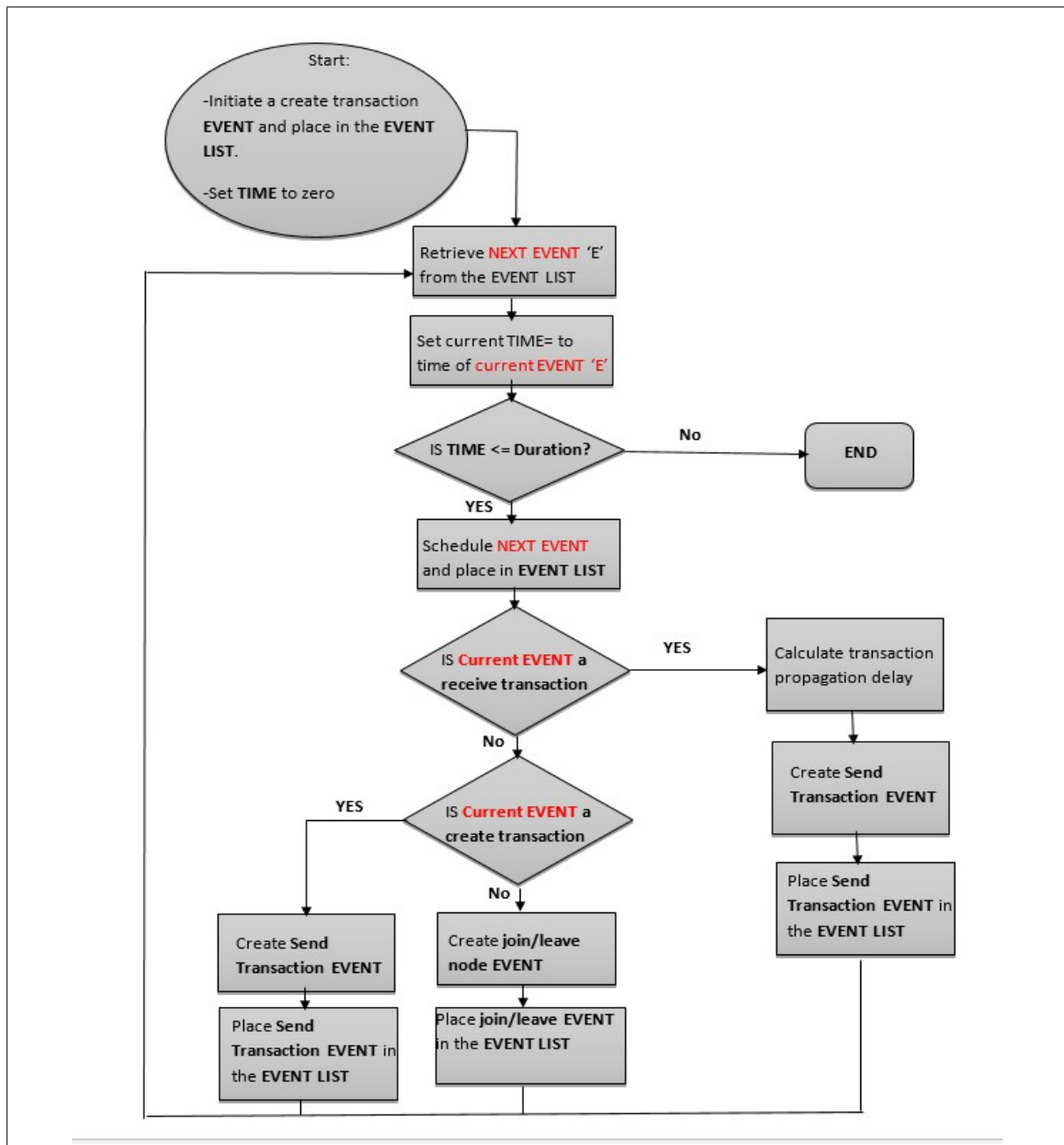


Fig. 4.5 Bitcoin simulator structure

#### 4.2.2.1 The Model Validation

In this section, the developed model is validated against the real Bitcoin network based on the transaction propagation delay. As several aspects of the real Bitcoin network such as client's behavior, processing delay, and network topology have a direct impact on transactions propagation, the transaction propagation delay measurements are important to test whether

the presented model behaves as close as possible to the real network. In the prior research, transaction propagation delay measurements were presented in the real Bitcoin network based on the propagation of INV messages. Specifically, the transaction propagation delay was measured in (Neudecker et al., 2015) (Decker & Wattenhofer, 2013) by setting up a Bitcoin client that keeps listening for INV messages. More clearly, the client calculates the time difference between the first reception of an INV message and subsequent receptions of INV messages, where all the received INV messages belong to the same announcement of a transaction. However, the collected measurements are not indicated when transactions are received, so these measurements do not represent the actual transaction propagation delay. Therefore, measurements of the transaction propagation delay in real Bitcoin network are performed in this work using a novel methodology by which the transaction propagation delay is accurately measured as these delays are indicated when peers receive transactions.

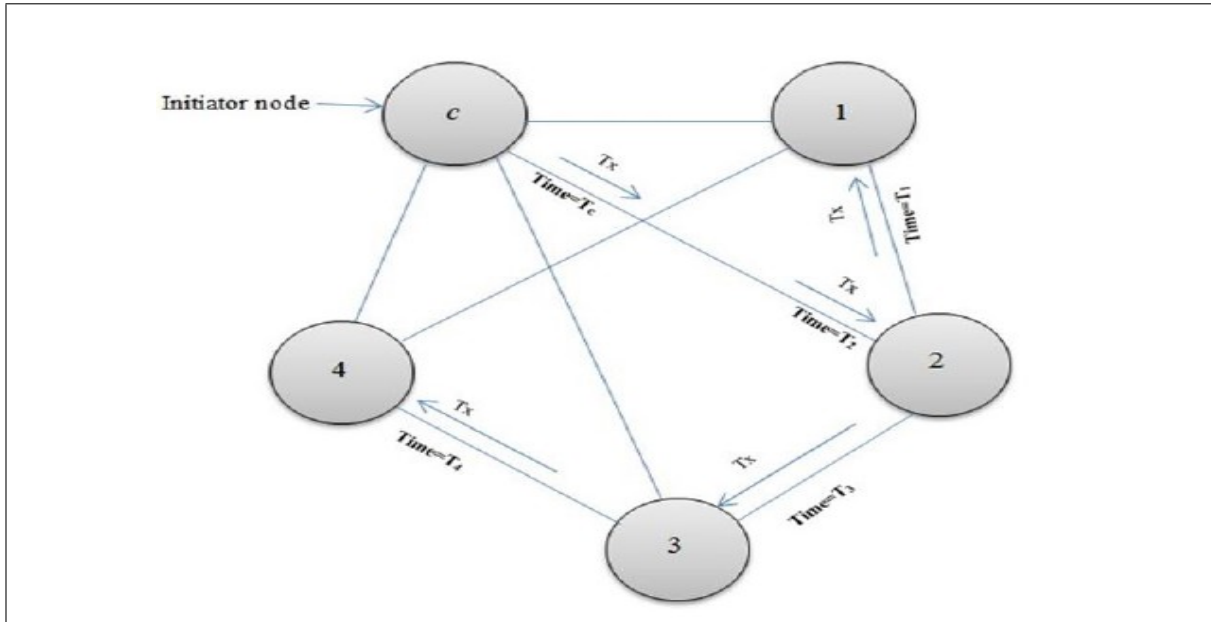


Fig. 4.6 Illustration of propagation experimental setup

To measure how fast a transaction is propagated in the Bitcoin network, the Bitcoin protocol was implemented and used to establish connections to many points in the network, in order to measure the time that a transaction takes to reach each point. Clearly, a measuring node is implemented, which behaves exactly like a normal node with the following functionalities. The measuring node connects to 10 reachable peers in the Bitcoin network. Furthermore, it is capable to create a valid transaction and propagate it to one peer of its connections, and then it tracks the transaction in order to record the time by which each peer of its connections announces the transaction. Specifically, suppose a client  $c$  has connections  $(1, 2, 3, \dots, n)$ ,  $c$  propagates a transaction at time  $T$ , and it is received by its connected nodes at different times

$(T_1, T_2, T_3, \dots, T_n)$  as illustrated in Fig.4.6. The time differences between the first transaction propagation and subsequent receptions of the transaction by connected nodes were calculated  $(\Delta t_{c,1}, \dots, \Delta t_{c,n})$  according to equation(4.1):

$$\Delta t_{c,n} = T_n - T_c \quad (4.1)$$

Where  $T_n > T_{n-1} > \dots, T_2, T_1$ . By running the measuring node, the time in which the transaction is propagated by the measuring node and reached each node of the measuring node's connections was calculated. Specifically, the timing information is collected by running the experiment 1000 times as one-off style events, networking delays, etc., are average out. At each run, the measuring node is randomly connected to 10 nodes. The number of connected nodes represents the sequence of the random nodes that the measuring node connects with at each run.

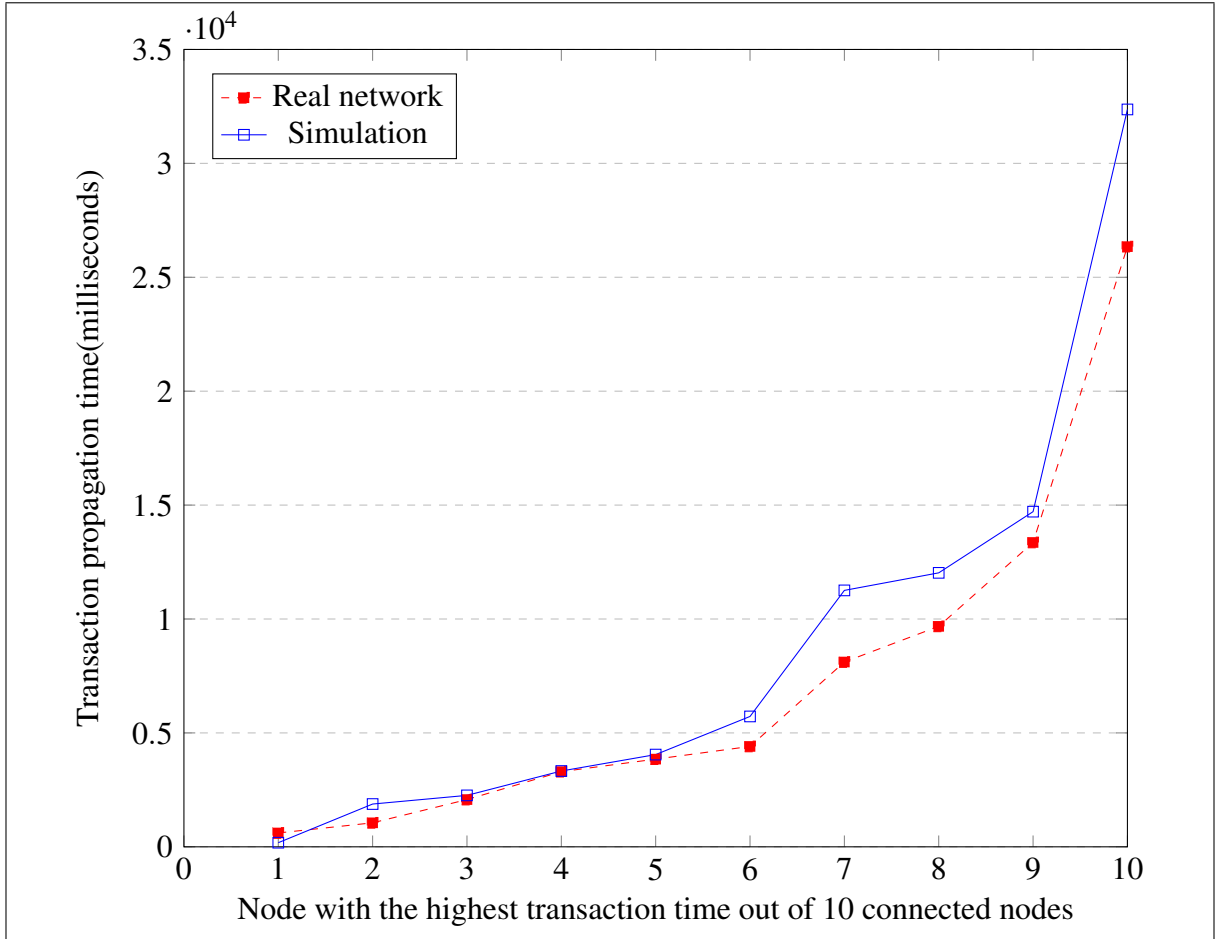


Fig. 4.7 Comparison of the distribution of  $\Delta t_{c,n}$  as measured in the real Bitcoin network with simulation results

In terms of measuring the transaction propagation delay in the simulation world, the aforementioned measuring method in the real Bitcoin network was used in the simulation. By doing this, the simulation model was validated by comparing the propagation delay measurements that have been collected from the Bitcoin simulator to the same measurements that have been collected from the real Bitcoin network. As the measurements are indicated when peers receive transactions, the distribution of these measured time differences  $\Delta t_{c,1}$  represents the real transaction propagation delay.

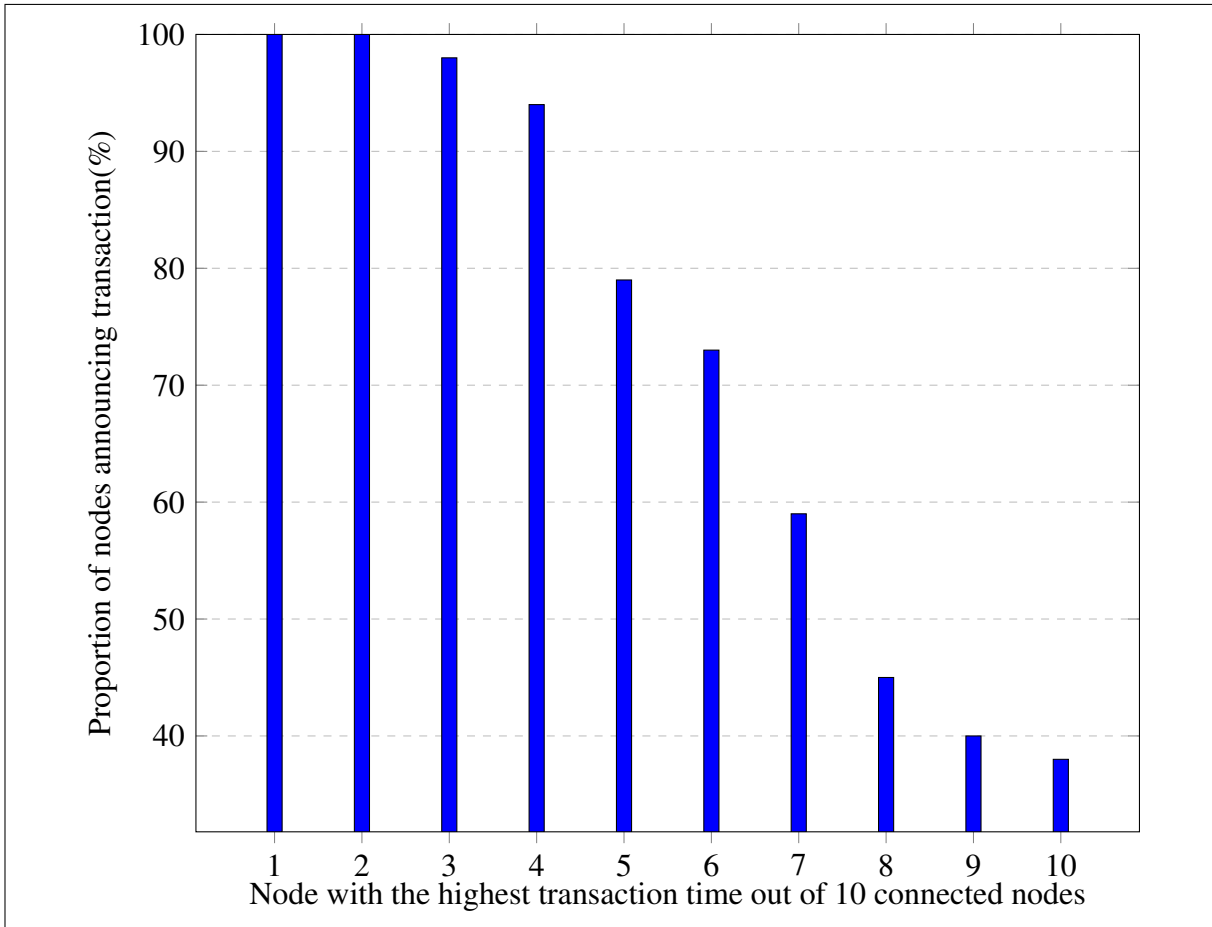


Fig. 4.8 Proportion of nodes that announced the transaction

The average distribution of  $\Delta t_{c,n}$  for the real Bitcoin network and the simulated network is shown in Fig. 4.7. Results reveal that during the first 13 seconds the transaction has been propagated faster and 6 nodes received it with low variance of delays. It should be noted that the transaction propagation delays are dramatically increased over nodes (9, 10) which means that the transaction has been received by these nodes with significantly larger variances of delays. Obviously, these results reveal that the propagation delay negatively corresponds with the number of nodes, as the total duration of subsequent announcements of the transaction by



the remaining nodes increases with larger numbers of connected nodes. This happened due to each node being connected to large segments of the network, while the connected nodes were not geographically localized. On the other hand, transaction verification at each node affects trickling transaction to the remaining nodes. However, we discovered that our simulation model approximately behaves as the real Bitcoin network.

Notably, it has been noticed while measuring the transaction propagation delay in the real Bitcoin network that not all of the connected nodes received the transaction except rare cases in which all 10 connected nodes announced the transaction. Fig.4.8 shows proportions of announcing transactions for each node. Each proportion was calculated over 1000 runs. Nodes 1, 2, 3 and 4 are almost received transactions within proportions between 90-100. The proportion dramatically declined at node 5 and continued to go down to reach 34 at node 10. This pointed to the issue that might be caused by network partitions in which the network is divided into two or more partitions due to network outages or link failure, so that no information flow between partitions is possible. However, network partitions can be done by an attacker to impair the main Bitcoin functions. Therefore, partition attacks in the Bitcoin network are evaluated in Chapter 5.

## 4.3 Conclusion

In this chapter, a model of the Bitcoin network which enables a full scale simulation of the Bitcoin network was presented. In order to parameterise the presented model, the size of the Bitcoin network, distribution of session's length, and distribution of latencies between nodes were accurately measured in this chapter. This chapter also shows a novel methodology to measure the transaction propagation delay in the real Bitcoin network. Transaction propagation measurements show that the transaction propagation time is significantly affected by the number of connected nodes and the network topology which is not geographically localised. In addition, partitions in the connection graph are actively detected. Validation of the presented model against the measurements of transaction propagation in the real Bitcoin network was performed. Validation results proved that the presented model behaves as close as the real Bitcoin network.



# Chapter 5

## Performance and Security Evaluation

### 5.1 Introduction

As mentioned in Chapter 4, the proposed protocols, namely BCBPT, LBC, BCBSN, and MNBC, are developed in order to speed up information propagation in the Bitcoin network with the aim of overcoming the limitations of the agreement on the common transactions history in the Bitcoin network. This Chapter evaluates the performance and security of the proposed protocols empirically in experimental studies. The developed simulator that is presented and validated in Chapter 4 is used in the evaluation of the proposed protocols. Evaluation results are also presented and discussed in this Chapter. Therefore, the second contribution and a part of the first contribution mentioned in Section 1.8 are fulfilled in this Chapter.

### 5.2 Performance evaluation

As the main goal of the proposed protocols is to perform faster information propagation in the Bitcoin network, the information propagation delay will be considered to be the main performance metric in the evaluation of every protocol of the proposed protocols. Specifically, the performance of the LBC, BCBPT, BCBSN, and MNBC is measured based on whether or not the proposed protocols offer faster transaction propagation in the Bitcoin network.

#### 5.2.1 Experiments setup

In this section, the experiment setup that is related to the performance evaluation of the proposed protocols LBC, BCBPT, BCBSN, and MNBC will be explained. This experiment includes four different scenarios of simulations, each scenario belongs to one of the proposed protocols. In each simulation, the size of the network matters as the evaluation is based on the transaction

propagation delay. Therefore, the size of the network in each simulation matches the size of the real Bitcoin network which was measured in Chapter 4, Section 4.2.1.3. Each node in the overlay is allowed to discover new nodes every 100ms. Dependent on which theory is applied, several proximity based clusters will be generated at certain times. As the performance evaluation is based on measuring how fast a transaction is propagated in the network after applying our clustering approaches, the transaction propagation delay in each approach is measured using the same methodology which was used in Chapter 4, Section 4.2.2.1, to measure the transaction propagation delay in the real and simulated Bitcoin network. By doing this, evaluation of each protocol of the proposed protocols against the real Bitcoin network as well as the rest of the proposed protocols, can be undertaken by comparing measurements of the transaction propagation delay that have been collected in the simulated Bitcoin protocol to the same measurements that have been collected in the simulated proposed protocols.

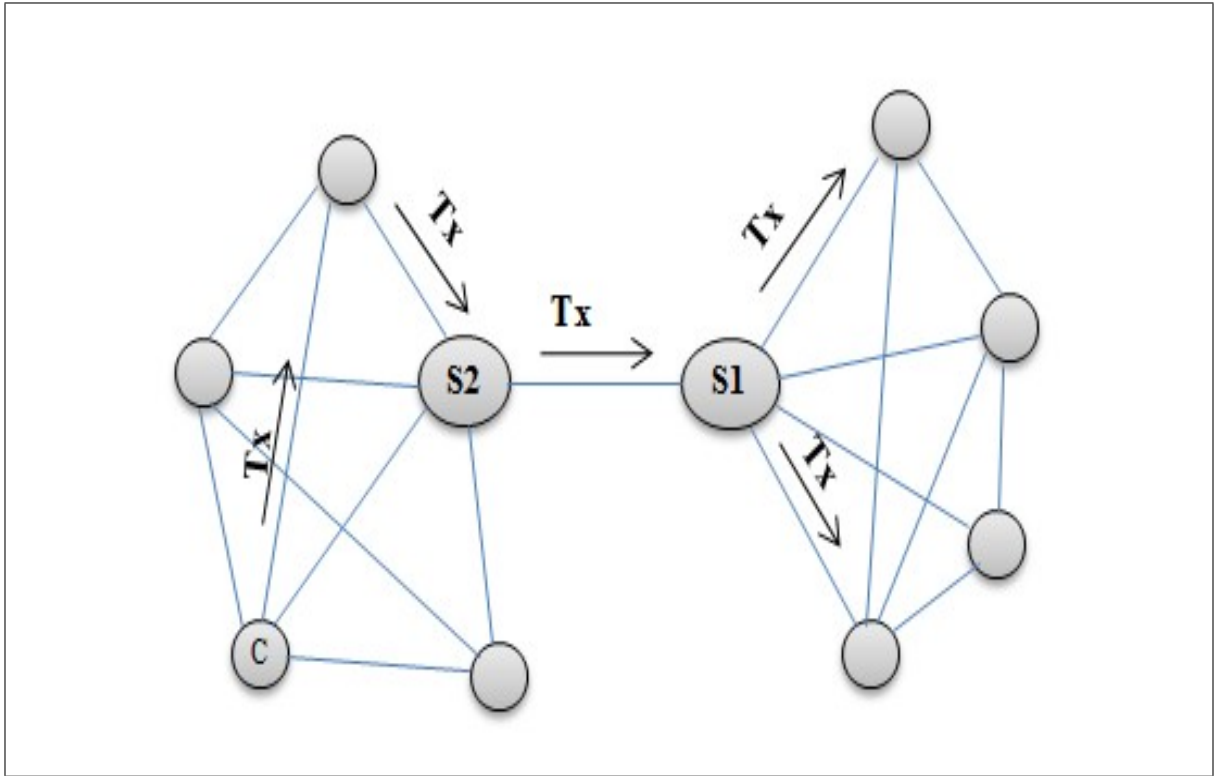


Fig. 5.1 BCBSN simulation setup

Fig.5.1 gives a simple diagram of how the simulation experiment works with respect to the BCBSN protocol, while Fig.5.2 illustrates the simulation setup of the MNBC protocol. On the other hand, Fig.5.3 shows an example of the simulation setup for BCBPT and LBC protocol. Before applying the aforementioned proximity cluster generation algorithms of the proposed

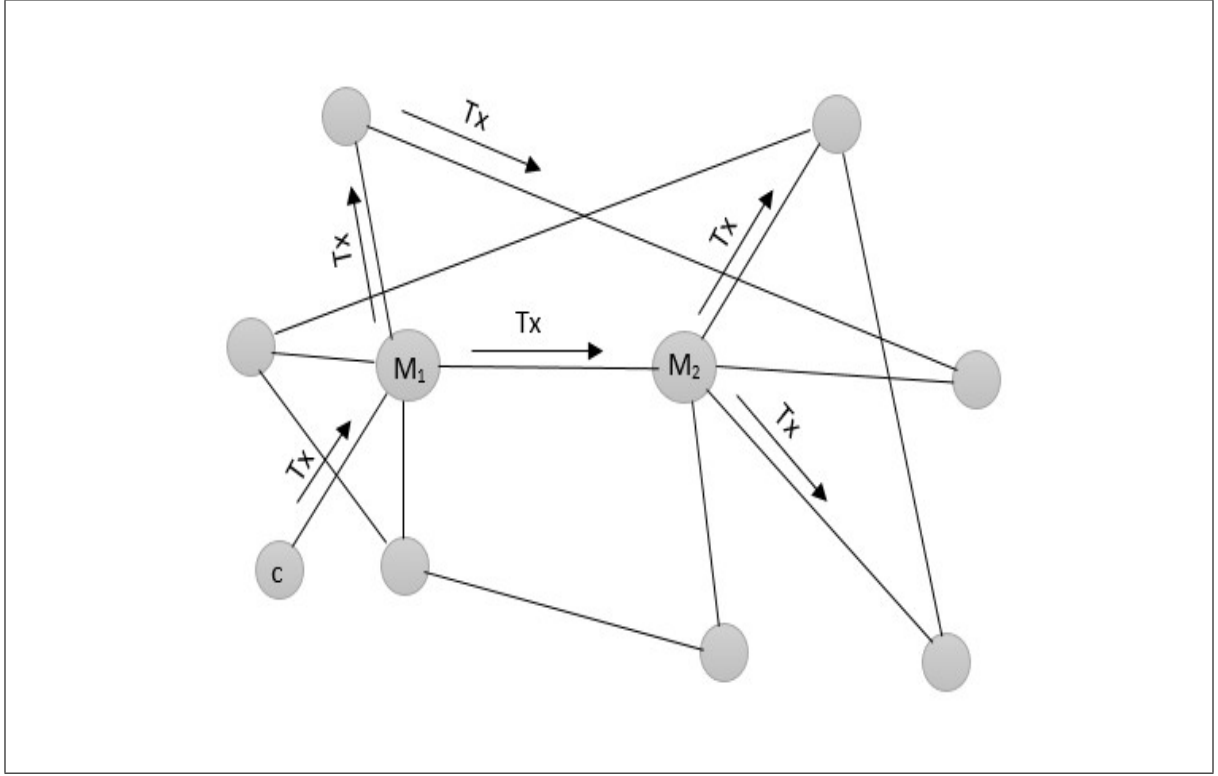


Fig. 5.2 MNBC simulation setup

techniques that are mentioned in Chapter 5, it is assumed that the network nodes belong to one cluster. Based on BCBPT and LBC protocol, several proximity based clusters will be generated at certain times based on a chosen ping latency threshold in BCBPT protocol, and geographical distance threshold in LBC protocol. In BCBPT, two nodes are close to each other if the measured distance based latency is lower than the suggested distance threshold  $d_t = 25\text{ms}$ . In LBC protocol, if the geographical distance between two nodes is lower than the suggested threshold  $d_t = 50\text{km}$ , then those nodes are close to each other. Regarding BCBSN protocol, super peers will be selected at certain times by running the super peer selection algorithm that is mentioned in Chapter 3, Section 3.4.1. After that, every super peer of the selected super peers construct a cluster by recruiting the geographical closer nodes. Similarly, master node selection algorithm in MNBC protocol that is mentioned in Chapter 3, Section 3.5.1, will be launched at a certain point in the experiment time to select master nodes. The selected master nodes will group peers that are close in the physical internet. However, the link distance between nodes are modelled based on real-world measurements that were collected in Chapter 3, Section 4.2.1.2.

After getting some proximity based clusters in every simulation scenario, normal Bitcoin simulator events will be launched. Within every protocol of the proposed protocols, a measuring node  $c$  is implemented which is able to create a valid transaction  $T_x$  and send it to one node of

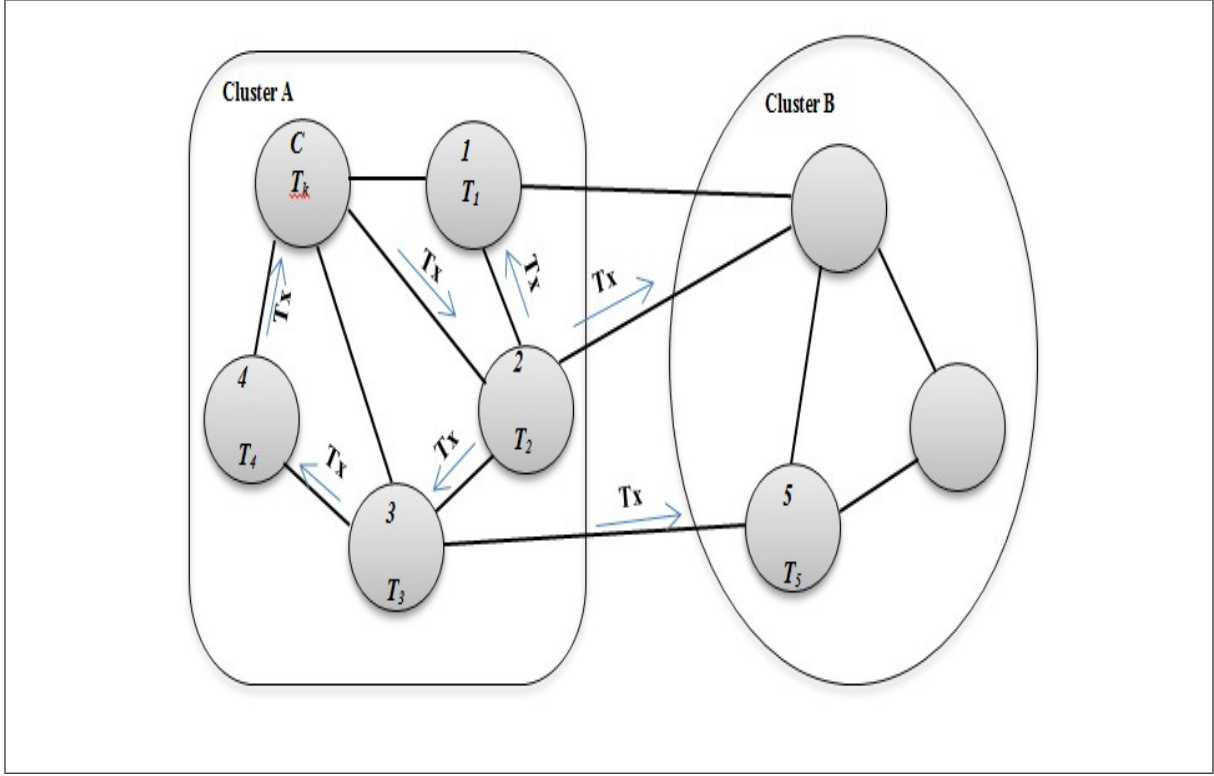


Fig. 5.3 LBC &amp; BCBPT simulation setup

its connected nodes. It then tracks the transaction in order to record the time by which each node of its connections announces the transaction. Therefore, transaction propagation delay is calculated based on the methodology that has been followed in Chapter 4, Section 4.2.2.1, to measure the transaction propagation delay in the real Bitcoin network.

However, the latency is determined by an average of approximately 1000 runs in order to increase the accuracy of the collected latencies which might be affected by several factors such as data corruption and loss of connection.

### 5.2.2 Results and discussions

The simulation results show that the proposed protocols offer an improvement in propagation delay compared to the Bitcoin protocol. Fig. 5.4 compares the distributions of  $\Delta t_{c,n}$  for the simulated Bitcoin protocol against the same distributions that have been measured in the simulated proposed protocols BCBSN, LBC, BCBPT, and MNBC. In the figure, the number of connected nodes represents the sequence of the random nodes that the measuring node connects with at each run. In all protocols, the distributions of delays increase gradually as the simulation time moves forward and the number of connected nodes increases. It should be noted that the

transaction propagation delays are larger in the simulated Bitcoin protocol over nodes (7,8,9,10) while it has been received with less delays in BCBSN, LBC, BCBPT, and MNBC over the same nodes sequence. This means that the transaction has been received by the connected nodes in BCBSN, LBC, BCBPT, and MNBC protocol with lower variances of delays compared to the simulated Bitcoin protocol. The reduction of the transaction propagation time variances in the proposed protocols has to do with the fact that the Bitcoin network layout, where nodes connect to other nodes without taking advantage of any proximity correlations, results in a high communication link cost measured by the distance between nodes. Consequently, the average delay to get transactions delivered is also increased which, on the other hand, would affect the consistency of the public ledger. In fact, contrary to what was previously thought in this area, result proved that reconstructing the Bitcoin network layout on proximity bases implies faster transmissions.

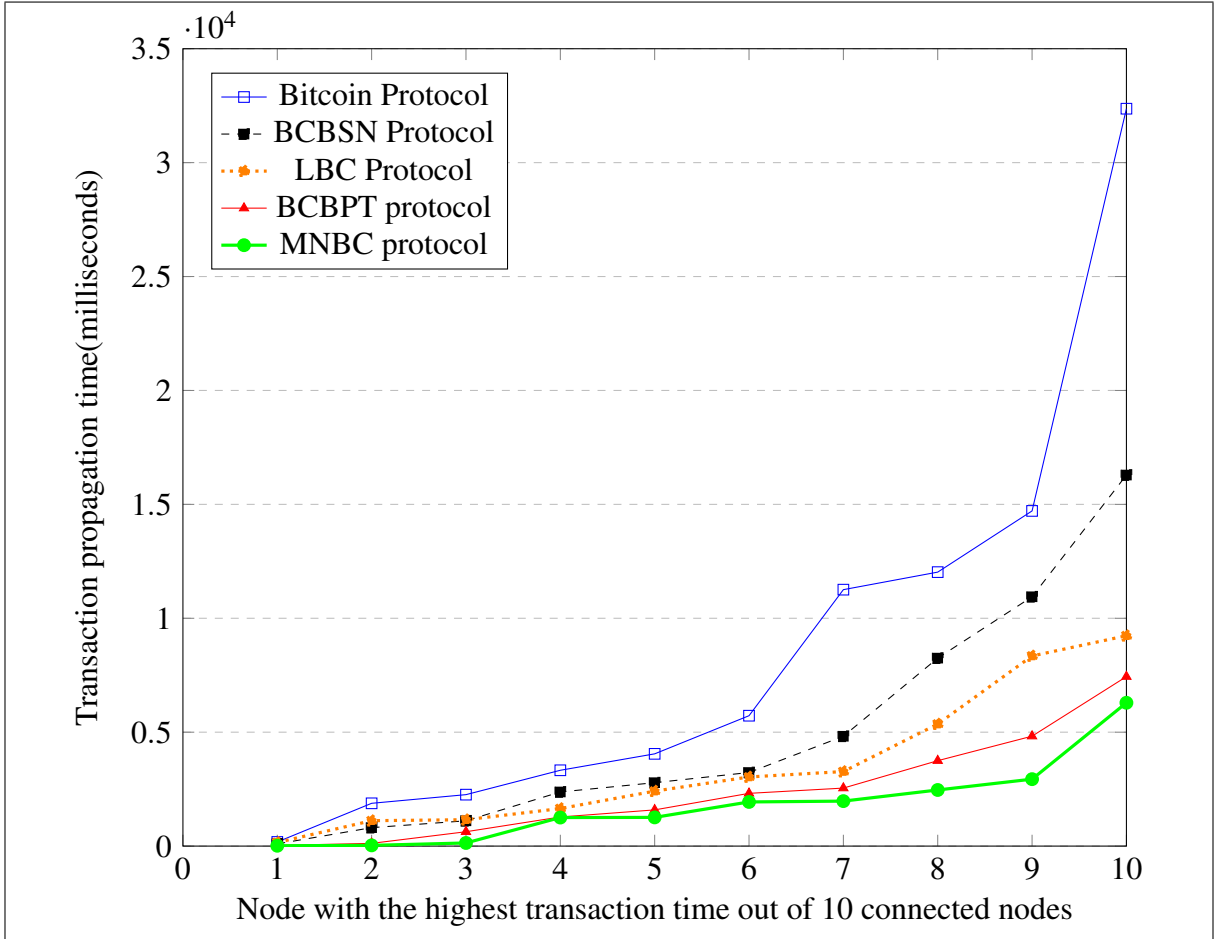


Fig. 5.4 Comparison of the distribution of  $\Delta t_{c,n}$  measured in the simulated Bitcoin protocol with BCBPT protocol, LBC protocol, BCBSN, and MNBC Protocol simulation results. ( $d_t$  in BCBPT=25ms,  $d_t$  in LBC=50km)

Turning now to the comparison between BCBPT, LBC, BCBSN, and MNBC protocol. As shown in Fig.5.4, all the proposed protocols show relatively the same variances of delays over nodes 1,2,3,4,5 and 6. From node 7, variances of delays in BCBSN protocol started climbing steadily and reached a peak over the node 10 recording a transaction propagation delay at nearly 18000 ms. In contrast, the trend of LBC variances of delays flattened off at a level of 2000 ms over the node 6 but then reached a peak of 2500 ms over the node 7. Since then, it has quickly increased and reached 9000 ms over the node 10. On the other hand, the variances of delays were improved in the BCBPT protocol over the LBC and BCBSN protocol, especially over nodes 8,9, and 10. Regarding MNBC protocol, it maintains faster transaction propagation delays compared to other trends in the graph regardless of the gradually increasing of those delays when the number of nodes increases. The most likely cause of the higher variances of delays in the BCBSN protocol is the fact that the information flow between clusters in BCBSN protocol can only be maintained through supers peers. This causes lack of transformation channels between clusters which results in inefficient information distribution over the network. The lack of connections between clusters in BCBSN protocol has been tackled in MNBC protocol by considering the edge nodes technology which adds an extra connection channels between clusters. Therefore, faster information propagation has been achieved in the MNBC protocol compared to the BCBSN protocol.

Even though the LBC protocol handles faster transaction propagation compared to the BCBSN protocol, lowest variances of delays have been maintained in the BCBPT protocol over the LBC and BCBSN protocol. It is almost certain that the cause of the lower variances of delays in the BCBPT protocol compared to the LBC protocol is that two geographically close nodes may actually be quite far from each other in the physical internet. Therefore, physical distance may lead to better results, leading to a different conclusion that the proximity awareness in the physical internet improves delivery latencies with a higher probability due to offering fewer hops as well as shorter links. However, comparison of MNBC's results with those of other proposed protocols confirms that the MNBC protocol achieves the best improvement that has been made to the delay of information propagation. A possible explanation for this improvement may be due to the adoption of the physical internet distance as a proximity metric in both edge nodes technology and clusters creation. Furthermore, the MNBC protocol provided an extra transformation channels by which faster information distribution is fulfilled.

As BCBPT and LBC protocol are based on a suggested threshold, it is worth investigating the optimal latency and geographical distance threshold that can achieve the best improvement in information propagation. To this purpose, we experiment with the BCBPT and LBC based on several suggested latency and geographical distance thresholds  $d_t$ . In BCBPT, the comparison among three variances of delays was undertaken based on three different latency suggested



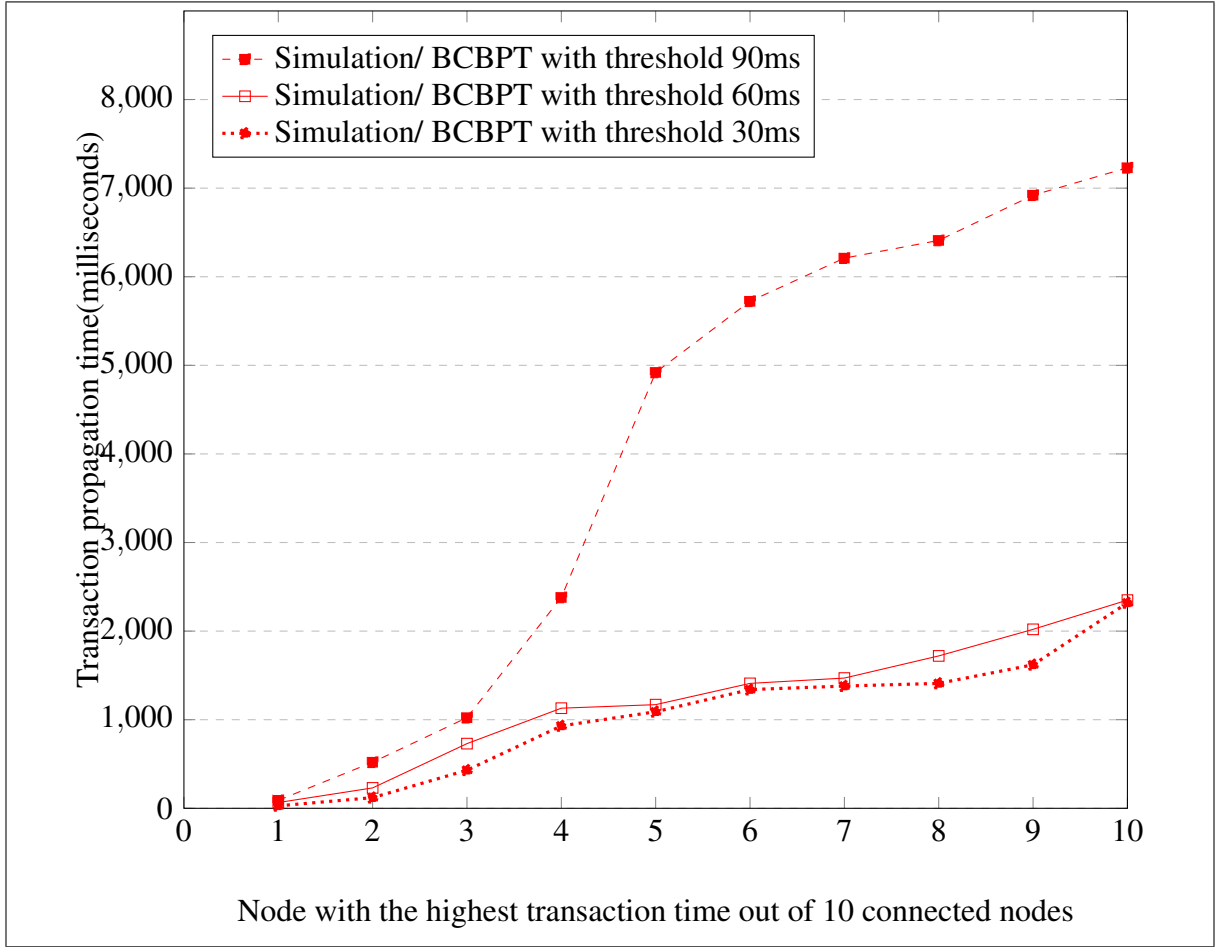


Fig. 5.5 Comparison of the distribution of  $\Delta t_{c,n}$  as measured in the simulated BCBPT protocol with three thresholds ( $d_t = 30\text{ms}$ ,  $60\text{ms}$ ,  $90\text{ms}$  )

thresholds 30 ms, 60 ms, and 90 ms, whereas the comparison was performed in LBC protocol based on 20 km, 50 km, and 100 km as geographical suggested thresholds. Results that are shown in Fig. 5.5 reveal that the less latency distance threshold in BCBPT performs less variance of delays. Judging from that, there is a negative correlation between propagation delay and the latency threshold, as the total duration of subsequent announcements of the transaction by the remaining nodes increases with a larger latency threshold. The key reason of variances of delays declining when the threshold value is reduced is that the number of nodes at each cluster is minimised due to the limited coverage of the physical topology which is offered by  $d_t$ . Similarly, less geographical distance threshold in LBC, as illustrated in Fig. 5.6, offers less variances of delays. The most likely cause of the reduction in variances of delays when the threshold value is minimised is that the limited coverage of geographical location offers less number of nodes at each cluster which results in reducing the number of hops that the transaction passes through.

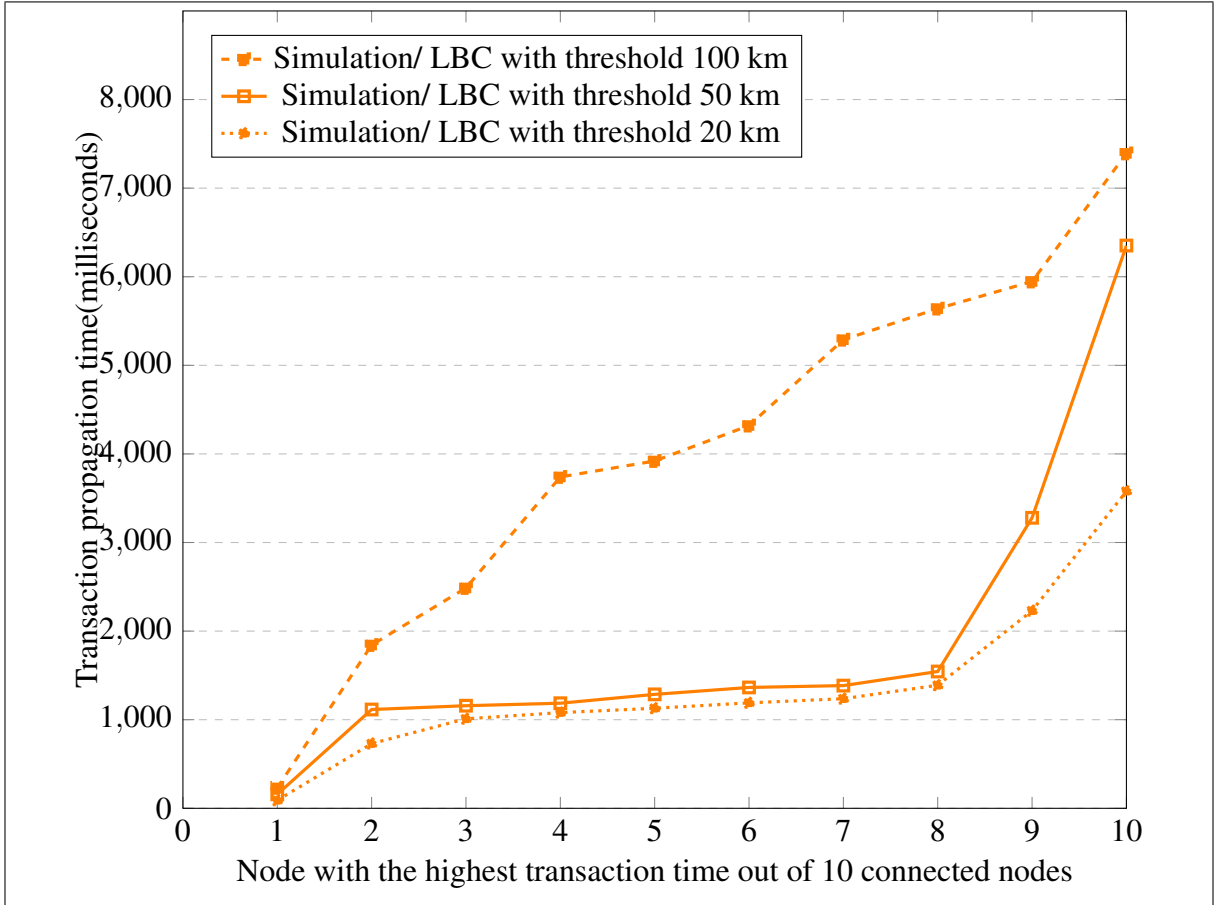


Fig. 5.6 Comparison of the distribution of  $\Delta t_{c,n}$  as measured in the simulated LBC protocol with three thresholds ( $d_t = 20$  km, 50 km, 100 km)

### 5.3 Security Evaluation

The potential of partition attacks on the proposed protocols as well as the Bitcoin network will be evaluated in this section using the designed simulator. Generally, the main goal of partition attacks is to partition the network into two or more partitions as well as prevent the information flow between partitions (Apostolaki et al., 2017). In terms of the Bitcoin network, the main target for an attacker launching partition attacks is to disturb the normal Bitcoin's main functions which would affect the users' trust in the system. This might be an incentive for an attacker due to the influence of users' trust in the system on the Bitcoin exchange rate.

According to the attack model presented in this thesis, the attack will be performed within three phases. The first phase starts when several malicious nodes which belong to an attacker join the peer-to-peer Bitcoin network and connect to many honest nodes. In order to increase the probability of connecting to as many honest nodes as possible, only IP addresses of attacker nodes are announced by other attacker nodes. Once the attacker guarantees that the satisfied

number of connections to honest nodes is maintained and the connectivity graph is thinned out, a proximate snapshot of the network graph layout will be given by launching the second scenario of the attack. This scenario can be achieved through a probabilistic method which has been introduced in (Biryukov et al., 2014). By this method, the Bitcoin network topology can be learnt within a reasonable probability through indicating whether or not two peers in the network are connected by sending marker addresses and observing the flow of these addresses. By doing so, the attacker will be able to indicate the *minimum vertex cut* of the network. *Minimum vertex cut* is defined as minimum honest peers that removing them causes splitting the graph into at least two partitions (Ugurlu et al., 2012). When the attacker selects peers for *Minimum vertex cut*, denial-of-service (DDOS) attack will be performed on the selected peers. However, our partition attack evaluation will be based on *minimum vertex cut*, as a metric to indicate the cost of performing partition attacks. This metric has been used in (Neudecker et al., 2015) to evaluate partition attacks in the Bitcoin network protocol.

### 5.3.1 Experiment setup

In this section, the experiment setup that deals with the evaluation of partition attacks in the Bitcoin network as well as the proposed protocols based on *minimum vertex cut* will be explained. The first phase of the attack will be started when the network topology is restructured according to each protocol of the proposed protocols. Specifically, several attacker nodes join the network and start establishing connections to many honest nodes. As the partition attack evaluation in this work is based on *minimum vertex cut* as a cost metric, *minimum vertex cut* of the network topology was determined at regular intervals using *metis* graph partition toolkits (Karypis & Kumar, 1995). *Metis* algorithm can achieve a balanced partitioning that minimize either the communication volume or number of edge cut (Miettinen et al., 2006). However, the attack's aim in this work is to get partitions of non-negligible size without taking into consideration whether or not the partitions are imbalanced. *Minimum vertex cut* is determined by an average of approximately 1000 runs in the simulated Bitcoin protocol and the proposed protocols.

### 5.3.2 Results and discussions

Fig.5.7 shows the results of four simulated attacks on a model of the real Bitcoin network, BCBPT, LBC, BCBSN, and MNBC protocol. Each attack was launched based on different network sizes (2000,4000,6000, and 8000). In the small scenarios with number of nodes (2,000 and 4,000), the number of honest peers on the minimum vertex cut in all protocols after launching the partition attack stayed below 500 which reveals that all protocols are relatively

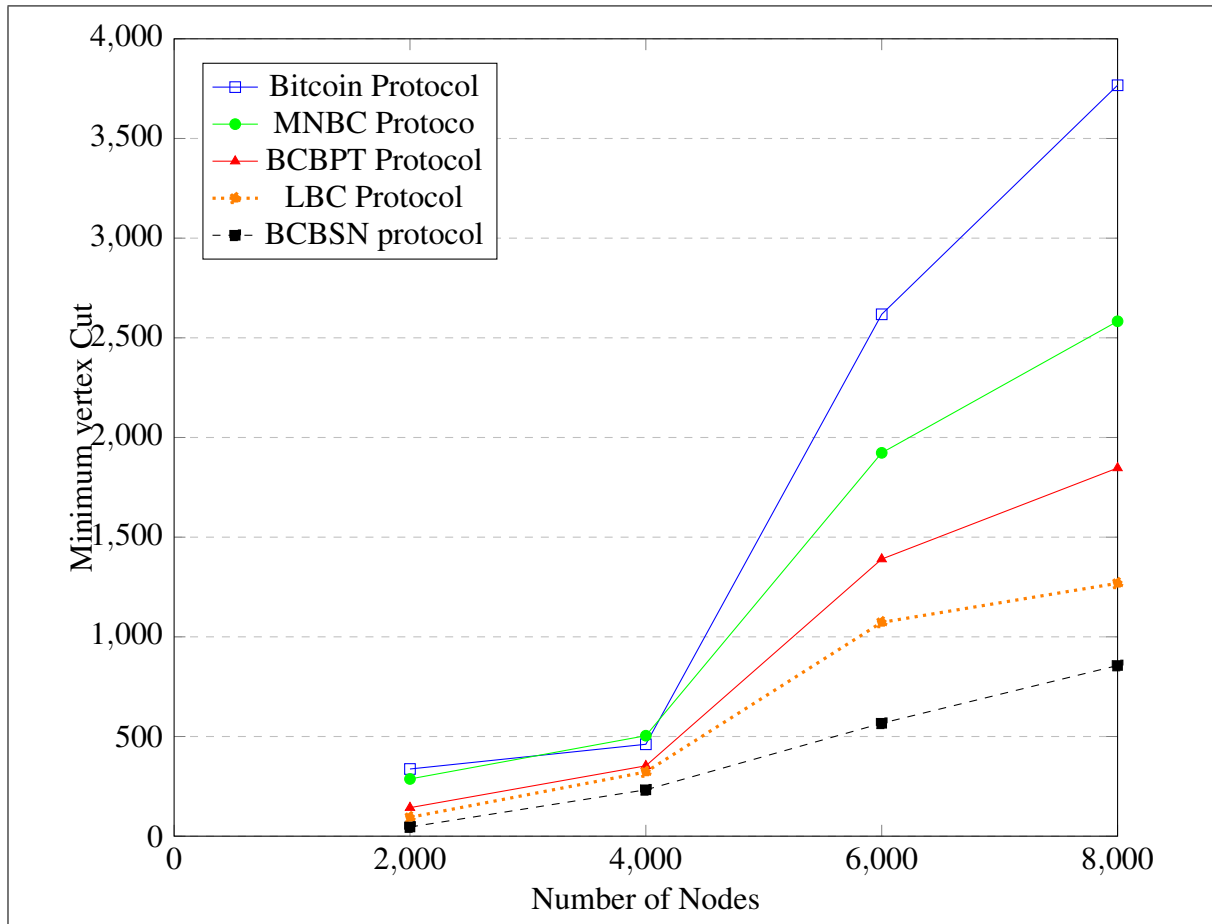


Fig. 5.7 Number of honest peers on the minimum vertex cut

similar in terms of resistance against partition attacks. While in the large scenarios with 6,000 and 8,000 peers, the level of resistance against partition attacks increased in all protocols as the number of nodes increases. The highest level is experienced in the Bitcoin protocol, while the lowest level is appeared in the BCBSN protocol. Precisely, the minimum vertex cut in the Bitcoin protocol increased from around 500 to 3,800 with the scenario of 8,000 peers resulting in a notable gap in the minimum vertex cut between the Bitcoin protocol and other protocols. Whereas, BCBPT, and LBC show less resistance with a minimum vertex cut below 2,000. On the other hand, MNBC protocol shows a relatively higher resistance over LBC and BCBPT, where the number of honest nodes in the minimum vertex cut goes above 2,500 in the scenario of 8,000 nodes. However, results of all scenarios reveal that the Bitcoin protocol is more resistant against partition attacks compared to the proposed protocols. BCBSN protocol is considered as the worst protocol of the proposed protocols in terms of the ease of performing partition attacks as it showed the lowest minimum vertex cut in both large and small scenarios. Although the proposed protocols experienced less minimum vertex cut compared to the Bitcoin

protocol, the number of honest nodes required to cut in the proposed protocols is still high which needs significant resources to be considered. As expected, clusters in MNBC that are fully connected via master nodes and edge nodes, and clusters in LBC and BCBPT that are connected via border nodes reflect less numbers of honest nodes in the minimum vertex cut. While, clusters in BCBSN that are connected via super peers result in the number of nodes in the area of the minimum vertex cut going down. However, results from large scenarios in all protocols illustrate that a higher number of peers, requires more effort to be spent by an attacker to split the network.

On the question of whether or not an attacker's session length affects the resistance of the network to partition attacks, the resistance of the Bitcoin network as well as the proposed protocols will be tested against several session lengths of attack. Fig.5.8 shows the results of the simulated partition attacks on a model of the real Bitcoin network, BCBPT, LBC, BCBSN, and MNBC protocol, including different session lengths.

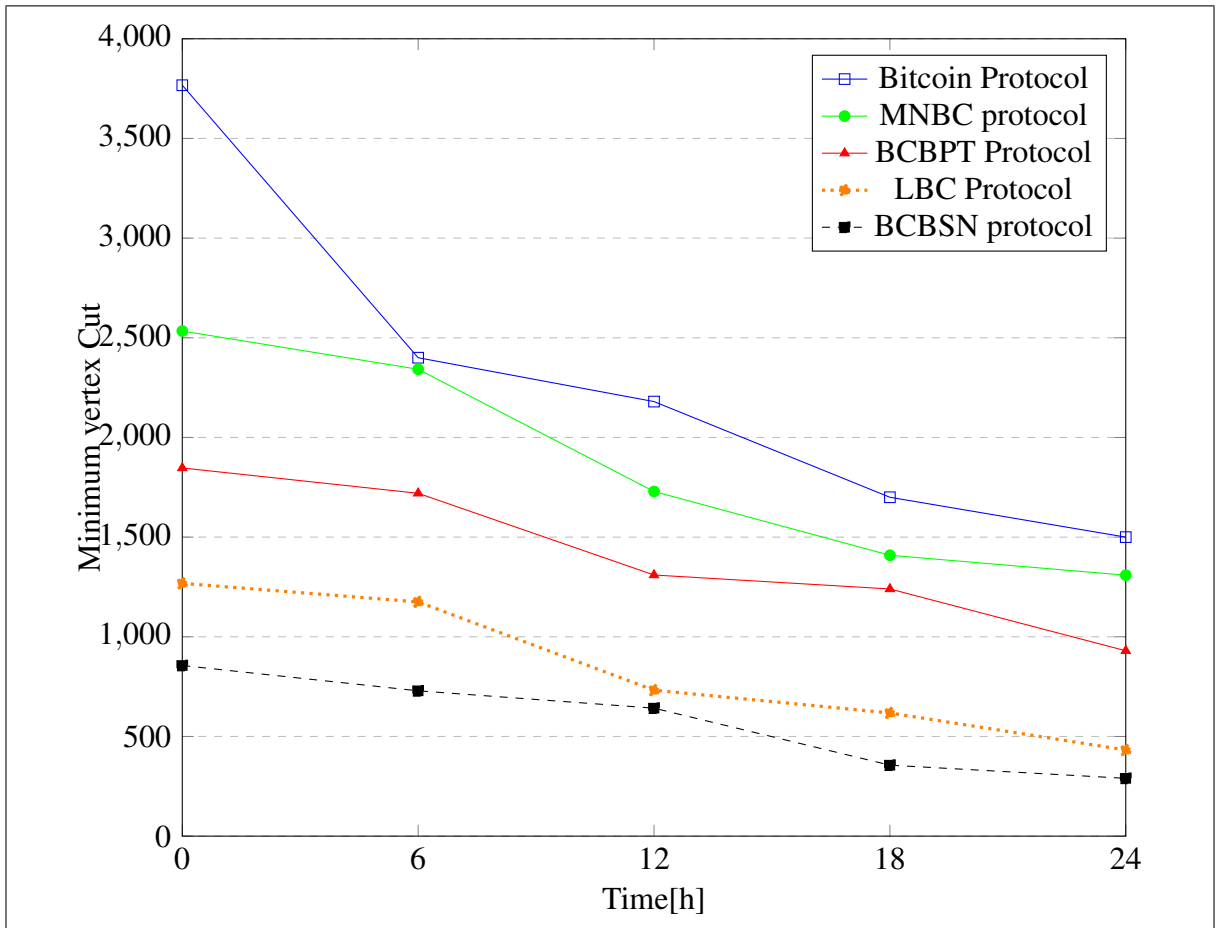


Fig. 5.8 Number of non-attacker peers on the minimum vertex cut during an attack with 7,000 honest peers parametrized as in the real-world network, attacker's session length  $S_A = 6h$

The results that are shown in Fig.5.8 illustrate the impact of the attacker's session length ( $S_A$ ) on the success of the attack. Within 24 hours of attack, the number of nodes in the minimum vertex cut declined in the simulated real Bitcoin network as well as the proposed protocols as follows: the minimum vertex cut declined from around 3,700 to 1,500 in the real Bitcoin network. The same scenario happened in the MNBC and BCBPT protocol where the minimum vertex cut decreased from around 2500 to 1150 in MNBC protocol, and from around 1,800 to 930 in BCBPT protocol. Similarly, minimum vertex cut dropped down from 1,200 to 430 in the LBC protocol, and from 850 to 290 in the BCBSN protocol. It can also be seen that the simulated real Bitcoin network still performs better than the rest of the proposed protocols in terms of the resistance to partition attacks. However, it can be concluded from the obtained results that more patience from the attackers with a higher number of peers, the better chances of success in splitting the network.

Moving on now to evaluate the impact of the number of clusters on the difficulty of performing partitioning attacks in the proposed approaches. Fig.5.9 shows the minimum vertex cut in the proposed approaches with respect to different numbers of clusters.

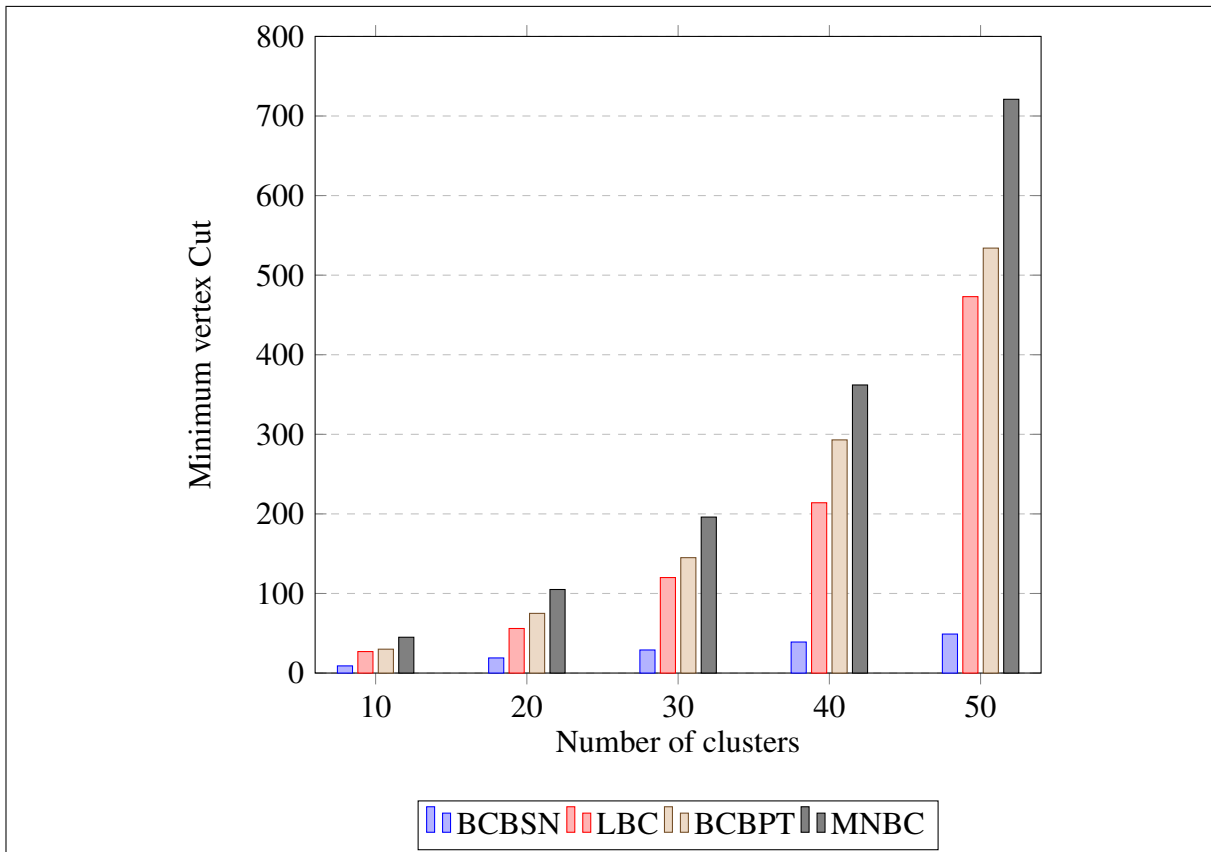


Fig. 5.9 Number of honest peers on the minimum vertex cut based on number of partitions in BCBSN, LBC, BCBPT, and MNBC

According to the results that are shown in Fig. 5.9, increasing the number of clusters results in more nodes in the minimum vertex cut. It can be noticed that the minimum vertex cut in all protocols positively corresponds with the number of clusters, more proximity clusters results in increasing the difficulty to perform partition attacks. This suggests a strong link may exist between the number of clusters and the improvement of the minimum vertex cut. This improvement can be translated based on the number of border nodes in the BCBPT and LBC approach, master nodes and edge nodes in MNBC approach, and super peers in the BCBSN protocol. Specifically, increasing the number of clusters in the BCBPT and LBC offers more border nodes between those clusters which results in more nodes in the minimum vertex cut. In respect to the BCBSN protocol, more clusters reflect more super peers that are classified as nodes in the minimum vertex cut. In MNBC approach, the number of master nodes and edge nodes that are located in the minimum vertex cut can be optimized throughout increasing the number of the proximity clusters in the network.

## 5.4 Conclusion

This chapter evaluated the performance and security of the proposed approaches. The performance evaluation was against the transaction propagation speed, while the security evaluation was based on the resistance of the proposed approaches to partition attacks. Performance evaluation results indicate an improvement in the transaction propagation delay over the Bitcoin network protocol. However, MNBC protocol maintains lowest variance of delays over the BCBPT, LBC, and BCBSN protocol. Furthermore, experiments with different latency thresholds in BCBPT as well as different geographical distance threshold values in LBC, have been conducted to identify the distance threshold that would give better improvement in the transaction propagation delay. We discovered that providing less latency and geographical distance threshold would improve the transaction propagation delay with a high proportion. Security evaluation results revealed that the Bitcoin network is more resistant against attackers than the proposed protocols. In addition, results proved that maximising the number of clusters in each proposed approach results in improving the network resistance against partition attacks. However, attackers still need more resources to split the network in the proposed protocols especially with a higher number of nodes.

The contributions of the thesis are summarised in Chapter 6. Furthermore, Chapter 6 clarifies the extent to which the research questions are answered. However, there is still room for further improvement in the specified research area in spite of that the clustering approaches introduced in the thesis shows significant advantages. Therefore, potential directions of this research area are also specified further in Chapter 6.





# Chapter 6

## Conclusions and Future Work

### 6.1 Introduction

The main aim of this thesis is to evaluate the impact of clustering on information propagation in the Bitcoin network as mentioned in Chapter 1. The experimental results shown in Chapter 5 prove that the incorporation of clustering based proximity can generally improve information propagation in the Bitcoin network. Even though the security evaluation results shown in Chapter 5 revealed that the adoption of clustering in the Bitcoin network might reduce the network resistance against partition attacks, partition attacks are still hard to perform. However, maximising the number of clusters in each proposed approach results in improving the network resistance against partition attacks. This thesis also introduces several clustering approaches that are based on how clusters in the Bitcoin network are formulated and the nodes define their membership. The main objective of these approaches is to increase the proximity of connectivity in the Bitcoin network which results in speeding up information propagation in the network without compromising security. Furthermore, measurements of the transaction propagation delay as well as large scale measurements of real Bitcoin network are presented in this thesis.

This chapter stresses how research questions have been addressed, and also highlights the contribution in Section 6.2, summarizes the methods and the findings of this research in Section 6.3, and sets out directions for future work in Section 6.5. The next section summarizes the contribution and the research questions, and illustrates the questions have been addressed.

## 6.2 Addressing the research questions

This research adopted a methodology that interleaves the hypothetico-deductive model and the discrete simulation study methodology to help address the research questions. The methodology stresses a three-stages process, which includes: (1) the analysis stage which involves the investigation of the literature to formulate the hypothesis and figure out the main influential entities that need to be modelled; (2) the simulation stage which involves the development of the Bitcoin simulation model; (3) the design stage which involves the experiments setup with the aim of answering the research questions. The first stage involved the formulation of the hypothesis that is supported and proved by the investigation of the literature. The second stage involved building the simulation model based on the main influential entities resulted from the first stage. The third stage involved designing experiments based on the simulation model resulted from the second stage. The methodology served to answer the research questions which resulted in achieving several contributions this research made, which include:

1. Evaluation of clustering as a technique to speed up information propagation in the Bitcoin network. The evaluation involved proposing four different clustering approaches which include; (1) Locality based clustering which aims to increase the locality of connectivity by grouping the Bitcoin network nodes based on geographical proximity; (2) Ping time based clustering which aims to incorporate proximity-awareness into the existing Bitcoin protocol by creating distinct, but connected clusters of peers with P2P latencies under a given intra-cluster threshold; (3) Super peer based clustering which aims to reduce the intermediate hops between any two peers as well as increasing the geographical connectivity between peers based on selected super peers; (4) Master node based clustering which aims to reduce the non-compulsory hops that the network information passes through as well as increasing the physical internet proximity of connectivity between peers based on the selected master nodes. (Chapter 3 & 5).
2. Evaluation of security which involved the evaluation of partition attacks in the Bitcoin network as well as the proposed approaches. (Chapter 5).
3. Providing the range of propagation delays within the real Bitcoin network with the aim of validating any model of Bitcoin. (Chapter 4)
4. Performing large scale measurements of the real Bitcoin network in order to enable a precise parameterisation of the Bitcoin simulation models. (Chapter 4).

The research questions and how they were answered are outlined below.

(1) What is the range of propagation delays within the real Bitcoin network?

To answer this question, the investigation of how fast a transaction is propagated in the Bitcoin network and how this is impacted by the number of nodes, was conducted in this PhD with the aim of gathering validation requirements. The transaction propagation delay was measured in the prior research by setup a Bitcoin client which keeps listening for INV messages. However, this thesis introduces a novel methodology by which the transaction propagation delay is accurately measured as measurements are indicated when peers receive transactions. Experiment methodology and results are presented in the Chapter 4.

(2) Can clustering in the Bitcoin network improve the information propagation delay?

In order to answer this question, the evaluation of clustering in the Bitcoin network was carried out in this PhD through evaluating four proposed clustering protocols in the Bitcoin network. The evaluation of each protocol of the proposed protocols is based on the information propagation delay as a main performance metric. The proposed clustering protocols are presented in Chapter 3, while the performance evaluation of these protocols are explained in Chapter 5.

(3) What is the security impact of the proposed protocols on the Bitcoin network?

In order to answer this question, evaluation of partition attacks against the proposed clustering protocols as well as the real Bitcoin network, was carried out in this PhD using the designed simulator and a partitioning tool (Metis). An attack model which involves three phases was developed with the aim of evaluating the requirements of partition attacks. The first phase starts when several malicious nodes that belong to an attacker join the peer-to-peer Bitcoin network and connect to many honest nodes. The second phase involves acquiring a proximate snapshot of the network graph layout with the aim of identifying peers in the minimum vertex cut. The third phase involves performing a DDOS attack on peers in the minimum vertex cut. Experiment methodology and results are presented in the Chapter 5.

(4) How to evaluate any clustering protocol in the Bitcoin network?

To evaluate the security and performance of any clustering theory based on improving information propagation, major changes are required to the Bitcoin protocol which would have to be accepted by the Bitcoin community. Therefore, this question is answered through developing a Bitcoin simulation model which is an event-based simulation framework dedicated to the simulation of the Bitcoin network. As building any model of the real world should be evidenced that the model is reflecting the virtual reality, we parameterise the presented model with data of the real Bitcoin network. This data was gathered by large-scale measurements of the real

Bitcoin network. To ensure the model's validity before any experiments are designed, we perform validation results of the presented model which were collected by comparing the model against the real Bitcoin network based on the transaction propagation delay. The model and the gathered measurements are illustrated in Chapter 4.

### 6.3 Summary of conducting this PhD research

In the Bitcoin network, all nodes participate in the transaction verification process which is achieved in a distributed manner. Due to this, nodes should agree to a common transactions history. However, achieving agreement on a common transactions log among nodes in the Bitcoin network is far from trivial as inconsistency in the replicas of the ledger has a potential and forks are more likely to be appeared. This incentivises attackers to perform double spending attacks. As the consistency of the public ledger is mainly affected by how fast Bitcoin information are distributed across the network, information propagation speed in the Bitcoin network is considered as an essential requirement that need to fulfilled. Alas, a significant delay in the information propagation is experienced due to the high latency of communications between nodes which is caused by the nature of the network layout where nodes are connected randomly without taking into account any proximity criteria. The discussion in Chapter 2 revealed that there are several methods have been proposed in order to fix this issue. However, previous attempts of updating the network topology structure have not taken into account any clustering approach. Instead, these attempts have considered either increasing the network connectivity by maintaining a mesh network topology, or relying on several coordinator nodes to support proximity of connectivity in the network without paying attention to security risks. Thus, evaluation of different clustering approaches based on speeding up information propagation in the Bitcoin network was considered throughout this research with the aim of filling the gap in this respect. As it is mentioned in the previous section, a research methodology that interleaves the hypothetico-deductive model and the discrete simulation study methodology is followed in this research to answer the research questions previously mentioned.

The aim of this research was to explore the potential of clustering with respect to improving the information propagation delay in the Bitcoin network without compromising security. Four clustering protocols that aim to increase the proximity of connectivity among nodes in the Bitcoin network, were proposed in this research. These clustering protocols include locality based clustering (LBC), Ping time based clustering (BCBPT), Super peer based clustering (BCBSN), and Master node based clustering (MNBC). The LBC protocol was designed to enhance local connections across the Bitcoin network via promoting connections to nearby nodes, in terms of geographical distance. To maximise the level of security awareness, LBC

protocol was achieved based upon the principle of distributed algorithm as each and every node will run the protocol in an independent fashion using data from nodes discovered in the local neighborhood. During this phase, nodes all over the network are categorized into clusters so that each cluster comprises nodes that belong to the same geographical location, this can be achievable by adding an extra function to each node across the Bitcoin network. On the other hand, the BCBPT protocol aims to increase the proximity of connectivity in the Bitcoin network by grouping Bitcoin nodes based on ping latencies between nodes. Each node in the Bitcoin network should run a distributed algorithm by which nodes connect to the closest nodes in the physical internet, resulting a significant improvement in the proximity of connectivity of the network.

The BCBSN protocol relies on several nodes, known as super peers, that contribute in creating a set of geographically divers and fully connected clusters. Each super peer is responsible for recruiting nearby nodes to its cluster as well as propagating network information to its cluster nodes and other super peers in different clusters. Whereas, clusters in MNBC protocol are generated via specific nodes, known as master nodes, based on proximity in the physical internet. Clusters are fully connected by master nodes and edge nodes. However, more details regarding the cluster creation and cluster maintenance in relation to all proposed protocols are mentioned in Chapter 3.

In order to evaluate the proposed protocols against the information propagation delay in the Bitcoin network, Bitcoin simulation model was developed in this research. Measurements of several network parameters that have a direct impact on information propagation were presented in this research to paramterise the presented model with the aim of reflecting the reality. These parameters include the size of the real Bitcoin network, session lengths, and link latencies between nodes. Furthermore, measurements of the transaction propagation delay in the Bitcoin network were collected using a novel methodology. These measurements were utilized to validate the Bitcoin simulation model that was developed in this research. The measurements of the collected Bitcoin network parameters and transaction propagation delay are presented and explained along with the Bitcoin model validation results in Chapter 4.

The performance evaluation of the proposed clustering protocols was carried out in this research using the developed simulator. Evaluation results indicate an improvement in the transaction propagation delay over the Bitcoin network protocol. However, MNBC maintains lowest variance of delays over the LBC, BCBPT, and BCBSN protocol. Furthermore, experiments with different latency thresholds in BCBPT as well as different geographical distance threshold values in LBC, have been conducted to identify the distance threshold that would give a better improvement in the transaction propagation delay. It has been discovered that providing less

latency and geographical distance threshold would improve the transaction propagation delay with a high proportion (See Chapter 5).

The security evaluation of the proposed protocols was conducted in this research by measuring the resistance of the proposed protocols against partition attacks. Evaluation results revealed that the Bitcoin network is more resistant against attackers than the proposed protocols. However, attackers still need more resources to split the network in the proposed protocols especially with a higher number of nodes. Furthermore, It has been proved that the resistance of the proposed protocols to partition attacks can be maximised by increasing the number of clusters in each protocol (See chapter 5).

## **6.4 Scientific and practical importance**

The findings from this PhD respond to the study's research questions and help to achieve its goal, which is to evaluate the concept of network clustering based proximity in improving the information propagation delay in the Bitcoin network without compromising security. These findings contribute in maximising the probability of reaching a consistent state over the blockchain. This results in reducing the chances of performing successful double spending attacks. However, this thesis also have some other scientific and practical aspects.

In the respect of scientific aspects, Bitcoin network measurements presented in this thesis can be used in other research that aim to analyse the behaviour of Bitcoin network nodes and perform studies with respect to the network structure. Furthermore, transaction propagation delays in the real Bitcoin network that are performed in this thesis can be used as a benchmark in other research with the aim of validating models of Bitcoin as well as evaluating the performance of the Bitcoin network. Moreover, the simulation model developed in this thesis can be used to conduct other studies that focus on analysing information propagation in the Bitcoin network as well as evaluating different classes of attacks on the Bitcoin network.

In the respect of practical aspects, findings from this thesis contribute in increasing the level of trust in Bitcoin due to the fact that speeding up information propagation in the Bitcoin network contributes in avoiding double spending attacks. As it is mentioned earlier, accelerating information propagation in the Bitcoin network maximises the probability of achieving an agreement on transactions history over the blockchain. This would encourage the adoption of the blockchain in domains involving voting and collective decision making.

## **6.5 Future work**

This section suggests extensions to the present work.

### 6.5.1 Reduce the size of the blockchain

Evaluation of four proposed clustering protocols were conducted in this PhD based on speeding up information propagation in the Bitcoin network. It would be interesting to consider evaluation of several techniques that can reduce the size of the public ledger at each node. As transactions are validated against the blockchain that currently contains a history of all transactions and it still grows in the size with each new transaction, reduction of transactions history at each node can play an important role towards achieving an optimal transaction verification time. To overcome this problem, a multichain protocol can be incorporated with our proposed clustering protocols in order to create an individual blockchain for each cluster. The evaluation of different interoperability techniques needs to be carried out in order to offer secure interoperability among chains of different clusters.

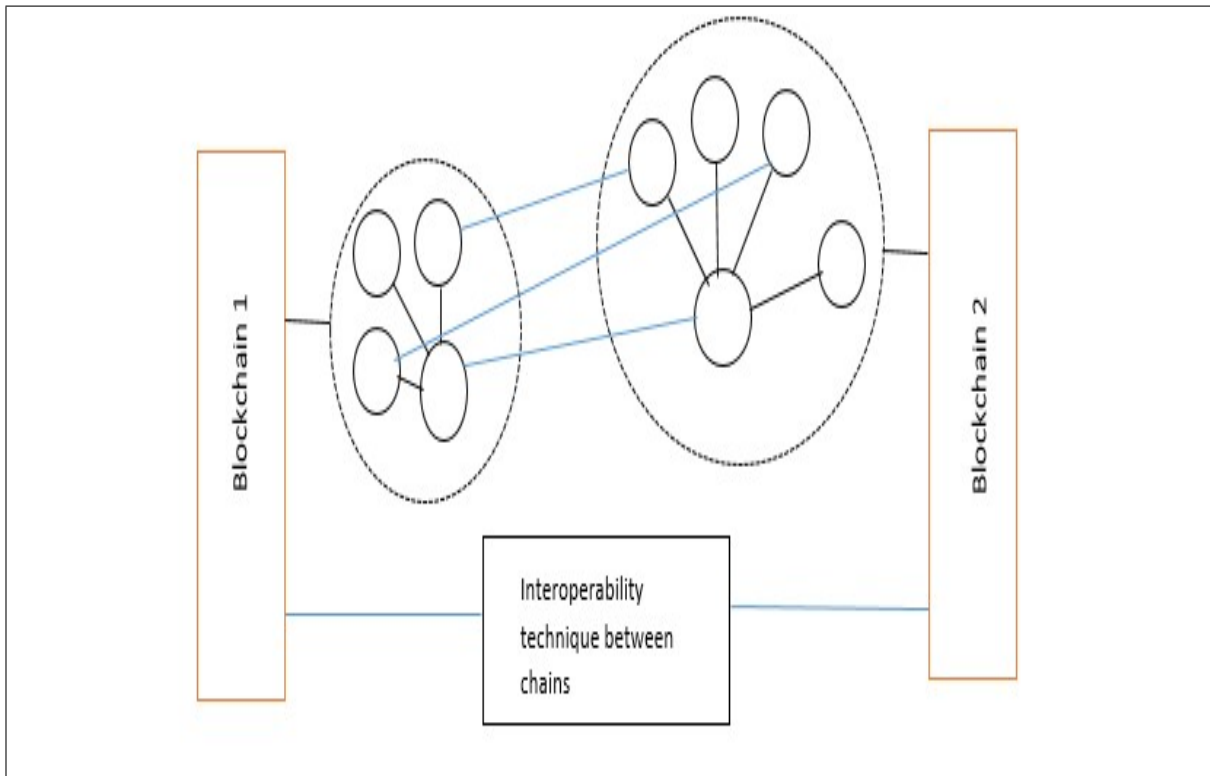


Fig. 6.1 Incorporating multichain protocol with clustering in the Bitcoin network

### 6.5.2 Observing blockchain forks

Evaluation of the proposed protocols was carried out in this research based on transaction propagation delay. However, it would be interesting to evaluate the proposed clustering protocol against the blockchain forks. Therefore, more research will be conducted to find out whether or not the proposed protocols reduce the blockchain occurrence rate.

### **6.5.3 Evaluating the network overhead**

To measure the distance between nodes in "ping latency" in the BCBPT protocol and MNBC protocol requires every pair of nodes to interact, which added an extra overhead to the network. This overhead will be evaluated in our future work. On the other hand, reducing the latency threshold in the BCBPT protocol and geographical distance threshold in the LBC protocol would decrease the size of clusters which leads to the creation of many clusters in the network. This might result in a side effect which is represented by an extra traffic overhead. This will be investigated in our future work.



# References

- Ahamad, S., Nair, M., & Varghese, B. (2013). A survey on crypto currencies. In *4th International Conference on Advances in Computer Science, AETACS* (pp. 42–48).
- Al Shehhi, A., Oudah, M., & Aung, Z. (2014). Investigating factors behind choosing a cryptocurrency. In *Industrial Engineering and Engineering Management (IEEM), 2014 IEEE International Conference on* (pp. 1443–1447). IEEE.
- Anderson, J. D., Campbell, J. K., Ekelund, J. E., Ellis, J., & Jordan, J. F. (2008). Anomalous orbital-energy changes observed during spacecraft flybys of earth. *Physical Review Letters*, *100*, 091102.
- Androulaki, E., Karame, G. O., Roeschlin, M., Scherer, T., & Capkun, S. (2013). Evaluating user privacy in bitcoin. In *International Conference on Financial Cryptography and Data Security* (pp. 34–51). Springer.
- Antonopoulos, A. M. (2014). *Mastering Bitcoin: unlocking digital cryptocurrencies*. " O'Reilly Media, Inc."
- Apostolaki, M., Zohar, A., & Vanbever, L. (2017). Hijacking bitcoin: Routing attacks on cryptocurrencies. In *Security and Privacy (SP), 2017 IEEE Symposium on* (pp. 375–392). IEEE.
- Babaioff, M., Dobzinski, S., Oren, S., & Zohar, A. (2012). On bitcoin and red balloons. In *Proceedings of the 13th ACM conference on electronic commerce* (pp. 56–73). ACM.
- Back, A., Corallo, M., Dashjr, L., Friedenbach, M., Maxwell, G., Miller, A., Poelstra, A., Timón, J., & Wuille, P. (2014). Enabling blockchain innovations with pegged sidechains. URL: <http://www.opensciencereview.com/papers/123/enablingblockchain-innovations-with-pegged-sidechains>, .
- Bamert, T., Decker, C., Elsen, L., Wattenhofer, R., & Welten, S. (2013). Have a snack, pay with bitcoins. In *Peer-to-Peer Computing (P2P), 2013 IEEE Thirteenth International Conference on* (pp. 1–5). IEEE.
- Bar-Noy, A., Kessler, I., & Sidi, M. (1995). Mobile users: To update or not to update? *Wireless Networks*, *1*, 175–185.
- Barber, S., Boyen, X., Shi, E., & Uzun, E. (2012). Bitter to better—how to make bitcoin a better currency. In *International Conference on Financial Cryptography and Data Security* (pp. 399–414). Springer.

- Basescu, C., Kokoris-Kogias, E., & Ford, B. A. (). *Low-latency Blockchain Consensus*. Technical Report.
- Bastiaan, M. (2015). Preventing the 51%-attack: a stochastic analysis of two phase proof of work in bitcoin. In *Available at <http://refraat.cs.utwente.nl/conference/22/paper/7473/preventingthe-51-attack-stochastic-analysis-of-two-phase-proof-of-work-in-bitcoin.pdf>*.
- Bindal, R., Cao, P., Chan, W., Medved, J., Suwala, G., Bates, T., & Zhang, A. (2006). Improving traffic locality in bittorrent via biased neighbor selection. In *Distributed Computing Systems, 2006. ICDCS 2006. 26th IEEE International Conference on* (pp. 66–66). IEEE.
- Biryukov, A., Khovratovich, D., & Pustogarov, I. (2014). Deanonimisation of clients in bitcoin p2p network. In *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security* (pp. 15–29). ACM.
- Biryukov, A., & Pustogarov, I. (2015). Bitcoin over tor isn't a good idea. In *Security and Privacy (SP), 2015 IEEE Symposium on* (pp. 122–134). IEEE.
- Bitcoin Wiki, B. (2008). Block. URL: <http://www.bitcoin.org/bitcoin.pdf>.
- Borshchev, A., & Filippov, A. (2004). From system dynamics and discrete event to practical agent based modeling: reasons, techniques, tools. In *Proceedings of the 22nd international conference of the system dynamics society*. volume 22.
- Brito, J., Shadab, H., & Castillo, A. (2014). Bitcoin financial regulation: Securities, derivatives, prediction markets, and gambling. *Colum. Sci. & Tech. L. Rev.*, 16, 144.
- Buchman, E. (2016). *Tendermint: Byzantine fault tolerance in the age of blockchains*. Ph.D. thesis.
- Chaum, D. (1983). Blind signatures for untraceable payments. In *Advances in cryptology* (pp. 199–203). Springer.
- Chuen, D. L. K. (2015). *Handbook of digital currency: Bitcoin, innovation, financial instruments, and big data*. Academic Press.
- Ciaian, P., Rajcaniova, M., & Kancs, d. (2016). The economics of bitcoin price formation. *Applied Economics*, 48, 1799–1815.
- Conti, M., Lal, C., Ruj, S. et al. (2017). A survey on security and privacy issues of bitcoin. *arXiv preprint arXiv:1706.00916*, .
- Croman, K., Decker, C., Eyal, I., Gencer, A. E., Juels, A., Kosba, A., Miller, A., Saxena, P., Shi, E., Sirer, E. G. et al. (2016). On scaling decentralized blockchains. In *International Conference on Financial Cryptography and Data Security* (pp. 106–125). Springer.
- Dai, W. (1998). B-money. *Consulted*, 1, 2012.
- Decker, C. (2016). *On the scalability and security of bitcoin*. Ph.D. thesis.

- Decker, C., & Wattenhofer, R. (2013). Information propagation in the bitcoin network. In *Peer-to-Peer Computing (P2P), 2013 IEEE Thirteenth International Conference on* (pp. 1–10). IEEE.
- Decker, C., & Wattenhofer, R. (2015). A fast and scalable payment network with bitcoin duplex micropayment channels. In *Symposium on Self-Stabilizing Systems* (pp. 3–18). Springer.
- Delgado-Segura, S., Pérez-Solà, C., Herrera-Joancomartí, J., Navarro-Arribas, G., & Borrell, J. (2018). Cryptocurrency networks: A new p2p paradigm. *Mobile Information Systems, 2018*.
- Dodig-Crnkovic, G. (2002). Scientific methods in computer science. In *Proceedings of the Conference for the Promotion of Research in IT at New Universities and at University Colleges in Sweden, Skövde, Suecia* (pp. 126–130).
- Donet, J. A. D., Pérez-Sola, C., & Herrera-Joancomartí, J. (2014). The bitcoin p2p network. In *International Conference on Financial Cryptography and Data Security* (pp. 87–102). Springer.
- Duffield, E., Schinzel, H., & Gutierrez, F. (2014). Transaction locking and masternode consensus: A mechanism for mitigating double spending attacks.
- Durlauf, S. N., Blume, L. et al. (2008). *The new Palgrave dictionary of economics* volume 6. Palgrave Macmillan Basingstoke.
- Erciyes, K., & Marshall, G. (2004). A cluster based hierarchical routing protocol for mobile networks. *Computational Science and Its Applications-ICCSA 2004*, (pp. 528–537).
- Eyal, I., & Sirer, E. G. (2014). Majority is not enough: Bitcoin mining is vulnerable. In *International conference on financial cryptography and data security* (pp. 436–454). Springer.
- Feld, S., Schönfeld, M., & Werner, M. (2014). Analyzing the deployment of bitcoin's p2p network under an as-level perspective. *Procedia Computer Science*, 32, 1121–1126.
- Garay, J. A., Kiayias, A., & Leonardos, N. (2015). The bitcoin backbone protocol: Analysis and applications. In *EUROCRYPT (2)* (pp. 281–310).
- Gavalas, D., Pantziou, G., Konstantopoulos, C., & Mamalis, B. (2006). Stable and energy efficient clustering of wireless ad-hoc networks with lidar algorithm. In *IFIP International Conference on Personal Wireless Communications* (pp. 100–110). Springer.
- Grimes, T. R. (1990). Truth, content, and the hypothetico-deductive method. *Philosophy of Science*, 57, 514–522.
- He, D., Habermeier, K. F., Leckow, R. B., Haksar, V., Almeida, Y., Kashima, M., Kyriakos-Saad, N., Oura, H., Sedik, T. S., Stetsenko, N. et al. (2016). *Virtual Currencies and Beyond: Initial Considerations*. Technical Report International Monetary Fund.
- Heilman, E., Kendler, A., Zohar, A., & Goldberg, S. (2015). Eclipse attacks on bitcoin's peer-to-peer network. In *USENIX Security Symposium* (pp. 129–144).
- Herley, C. E. (2008). Peer to peer network. US Patent 7,343,418.

- Hsieh, H.-C., & Chiang, M.-L. (2014). Improvement of the byzantine agreement problem under mobile p2p network. *IERI Procedia*, 10, 45–50.
- Kaplanov, N. (2012). Nerdy money: Bitcoin, the private digital currency, and the case against its regulation. *Loy. Consumer L. Rev.*, 25, 111.
- Karame, G., Androulaki, E., & Capkun, S. (2012). Two bitcoins at the price of one? double-spending attacks on fast payments in bitcoin. *IACR Cryptology ePrint Archive*, 2012.
- Karame, G. O., Androulaki, E., Roeschlin, M., Gervais, A., & Čapkun, S. (2015). Misbehavior in bitcoin: A study of double-spending and accountability. *ACM Transactions on Information and System Security (TISSEC)*, 18, 2.
- Karypis, G., & Kumar, V. (1995). Metis—unstructured graph partitioning and sparse matrix ordering system, version 2.0, .
- Kesaraju, V., & Ciarallo, F. W. (2012). Integrated simulation combining process-driven and event-driven models. *Journal of Simulation*, 6, 9–20.
- Kiayias, A., Lamprou, N., & Stouka, A.-P. (2016). Proofs of proofs of work with sublinear complexity. In *International Conference on Financial Cryptography and Data Security* (pp. 61–78). Springer.
- Kim, T. H. (2016). A study of digital currency cryptography for business marketing and finance security. *Asia-pacific Journal of Multimedia Services Convergent with Art, Humanities, and Sociology*, 6, 365–376.
- Kleinrock, L. (1996). Nomadicity: anytime, anywhere in a disconnected world. *Mobile networks and applications*, 1, 351–357.
- Kondor, D., Pósfai, M., Csabai, I., & Vattay, G. (2014). Do the rich get richer? an empirical analysis of the bitcoin transaction network. *PloS one*, 9, e86197.
- Koshy, P., Koshy, D., & McDaniel, P. (2014). An analysis of anonymity in bitcoin using p2p network traffic. In *International Conference on Financial Cryptography and Data Security* (pp. 469–485). Springer.
- Kraft, D. (2016). Difficulty control for blockchain-based consensus systems. *Peer-to-Peer Networking and Applications*, 9, 397–413.
- Lamport, L., Shostak, R., & Pease, M. (1982). The byzantine generals problem. *ACM Transactions on Programming Languages and Systems (TOPLAS)*, 4, 382–401.
- Law, A. M. (2003). Designing a simulation study: how to conduct a successful simulation study. In *Proceedings of the 35th conference on Winter simulation: driving innovation* (pp. 66–70). Winter Simulation Conference.
- Law, A. M., Kelton, W. D., & Kelton, W. D. (1991). *Simulation modeling and analysis* volume 2. McGraw-Hill New York.
- Law, L., Sabet, S., & Solinas, J. (1996). How to make a mint: the cryptography of anonymous electronic cash. *Am. UL Rev.*, 46, 1131.

- Lee, S.-J., Su, W., & Gerla, M. (2002). On-demand multicast routing protocol in multihop wireless mobile networks. *Mobile networks and applications*, 7, 441–453.
- Leonard, D., Rai, V., & Loguinov, D. (2005). On lifetime-based node failure and stochastic resilience of decentralized peer-to-peer networks. *ACM SIGMETRICS performance evaluation review*, 33, 26–37.
- Li, J., Stribling, J., Morris, R., Kaashoek, M. F., & Gil, T. M. (2005). A performance vs. cost framework for evaluating dht design tradeoffs under churn. In *INFOCOM 2005. 24th Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings IEEE* (pp. 225–236). IEEE volume 1.
- Lin, C. R., & Gerla, M. (1997). Adaptive clustering for mobile wireless networks. *IEEE Journal on Selected areas in Communications*, 15, 1265–1275.
- Ma, A. (2006). Netgeo—the internet geographic database.
- Malkhi, D., & Reiter, M. (1998). Byzantine quorum systems. *Distributed computing*, 11, 203–213.
- Marshall, G., & Erciyes, K. (2005). Implementation of a cluster based routing protocol for mobile networks. *Computational Science–ICCS 2005*, (pp. 915–918).
- Maxmind (2013). Maxmind - geolite legacy downloadable databases. retrieved from <http://dev.maxmind.com/geoip/legacy/geolite/>. URL: <http://dev.maxmind.com/geoip/legacy/geolite/>.
- Miers, C. C., Simplicio, M. A., Gallo, D. S., Carvalho, T. C., Bressan, G., Souza, V., Karlsson, P., & Damola, A. (2010). A taxonomy for locality algorithms on peer-to-peer networks. *IEEE Latin America Transactions*, 8, 323–331.
- Miers, I., Garman, C., Green, M., & Rubin, A. D. (2013). Zerocoin: Anonymous distributed e-cash from bitcoin. In *Security and Privacy (SP), 2013 IEEE Symposium on* (pp. 397–411). IEEE.
- Miettinen, P., Honkala, M., & Roos, J. (2006). *Using METIS and hMETIS algorithms in circuit partitioning*. Helsinki University of Technology.
- Miller, A., & Jansen, R. (2015). Shadow-bitcoin: Scalable simulation via direct execution of multi-threaded applications. *IACR Cryptology ePrint Archive*, 2015, 469.
- Miller, A., & LaViola Jr, J. J. (2014). Anonymous byzantine consensus from moderately-hard puzzles: A model for bitcoin. Available on line: <http://nakamotoinstitute.org/research/anonymous-byzantine-consensus/>, .
- Mishkin, F. S. (2007). *The economics of money, banking, and financial markets*. Pearson education.
- Mizrak, A. T., Cheng, Y., Kumar, V., & Savage, S. (2003). Structured superpeers: Leveraging heterogeneity to provide constant-time lookup. In *Internet Applications. WIAPP 2003. Proceedings. The Third IEEE Workshop on* (pp. 104–111). IEEE.

- Moore, T., & Christin, N. (2013). Beware the middleman: Empirical analysis of bitcoin-exchange risk. In *International Conference on Financial Cryptography and Data Security* (pp. 25–33). Springer.
- Nakamoto, S. (2008). Bitcoin: A peer-to-peer electronic cash system. URL: <http://www.bitcoin.org/bitcoin.pdf>.
- Natoli, C., & Gramoli, V. (2016). The balance attack against proof-of-work blockchains: The r3 testbed as an example. *arXiv preprint arXiv:1612.09426*, .
- Neudecker, T., Andelfinger, P., & Hartenstein, H. (2015). A simulation model for analysis of attacks on the bitcoin peer-to-peer network. In *Integrated Network Management (IM), 2015 IFIP/IEEE International Symposium on* (pp. 1327–1332). IEEE.
- Ober, M., Katzenbeisser, S., & Hamacher, K. (2013). Structure and anonymity of the bitcoin transaction graph. *Future internet*, 5, 237–250.
- Okupski, K. (2014). Bitcoin developer reference. Available at <http://enetium.com/resources/Bitcoin.pdf>, .
- Pazmiño, J. E., & da Silva Rodrigues, C. K. (2015). Simply dividing a bitcoin network node may reduce transaction verification time. *The SIJ Transactions on Computer Networks and Communication Engineering (CNCE)*, 3, 17–21.
- Peart, A. (2014). *Optimising quality of service levels through experimentation on streaming multimedia applications using WiMAX*. Ph.D. thesis University of Portsmouth, UK.
- Pease, M., Shostak, R., & Lamport, L. (1980). Reaching agreement in the presence of faults. *Journal of the ACM (JACM)*, 27, 228–234.
- Qiu, D., & Srikant, R. (2004). Modeling and performance analysis of bittorrent-like peer-to-peer networks. In *ACM SIGCOMM computer communication review* (pp. 367–378). ACM volume 34.
- Ramaswamy, L., Gedik, B., & Liu, L. (2005). A distributed approach to node clustering in decentralized peer-to-peer networks. *IEEE Transactions on Parallel and Distributed Systems*, 16, 814–829.
- Ringland, J. (2010). System science of virtual reality: Toward the unification of, .
- Ron, D., & Shamir, A. (2013). Quantitative analysis of the full bitcoin transaction graph. In *International Conference on Financial Cryptography and Data Security* (pp. 6–24). Springer.
- Rosenberg, B. (2010). *Handbook of financial cryptography and security*. CRC Press.
- Rosenfeld, M. (2014). Analysis of hashrate-based double spending. *arXiv preprint arXiv:1402.2009*, .
- Schollmeier, R. (2001). A definition of peer-to-peer networking for the classification of peer-to-peer architectures and applications. In *Peer-to-Peer Computing, 2001. Proceedings. First International Conference on* (pp. 101–102). IEEE.

- Schollmeier, R., & Kunzmann, G. (2003). Gnuviz—mapping the gnutella network to its geographical locations. *Praxis der Informationsverarbeitung und Kommunikation*, 26, 74–79.
- Schrijvers, O., Bonneau, J., Boneh, D., & Roughgarden, T. (2016). Incentive compatibility of bitcoin mining pool reward functions. In *International Conference on Financial Cryptography and Data Security* (pp. 477–498). Springer.
- Sharma, S. K., Krishma, N. N., & Raina, E. C. (2017). Survey paper on cryptocurrency, .
- Sompolinsky, Y., & Zohar, A. (2013). Accelerating bitcoin's transaction processing. fast money grows on trees, not chains. *IACR Cryptology ePrint Archive*, 2013.
- Spagnuolo, M., Maggi, F., & Zanero, S. (2014). Bitiodine: Extracting intelligence from the bitcoin network. In *International Conference on Financial Cryptography and Data Security* (pp. 457–468). Springer.
- Stathakopoulou, C., Decker, C., & Wattenhofer, R. (2015). A faster bitcoin network. *Tech. rep., ETH, Zurich, Semester Thesis*, .
- Stutzbach, D., & Rejaie, R. (2006). Understanding churn in peer-to-peer networks. In *Proceedings of the 6th ACM SIGCOMM conference on Internet measurement* (pp. 189–202). ACM.
- Stutzbach, D., Rejaie, R., & Sen, S. (2008). Characterizing unstructured overlay topologies in modern p2p file-sharing systems. *IEEE/ACM Transactions on Networking*, 16, 267–280.
- Szabo, N. (1998). Secure property titles with owner authority. *Online at <http://szabo.best.vwh.net/securetitle.html>*, .
- Szabo, N. (2008). Bit gold. Available: <http://unenumerated.blogspot.de/2005/12/bit-gold.html>, .
- Tschorsch, F., & Scheuermann, B. (2016). Bitcoin and beyond: A technical survey on decentralized digital currencies. *IEEE Communications Surveys & Tutorials*, 18, 2084–2123.
- Ugurlu, O., Berberler, M. E., Kızılates, G., & Kurt, M. (2012). New algorithm for finding minimum vertex cut set. In *Problems of Cybernetics and Informatics (PCI), 2012 IV International Conference* (pp. 1–4). IEEE.
- Veness, C. (2011). Calculate distance and bearing between two latitude/longitude points using haversine formula in javascript. *Movable Type Scripts*, .
- Vishnumurthy, V., Chandrakumar, S., & Sirer, E. G. (2003). Karma: A secure economic framework for peer-to-peer resource sharing. In *Workshop on Economics of Peer-to-Peer Systems*. volume 35.
- Wang, D. (2014). Secure implementation of ecdsa signature in bitcoin. *Information Security at University College of Longon*, .
- Xu, K. (2013). Performance modeling of bittorrent peer-to-peer file sharing networks. *arXiv preprint arXiv:1311.1195*, .

- Yang, B. B., & Garcia-Molina, H. (2003). Designing a super-peer network. In *Data Engineering, 2003. Proceedings. 19th International Conference on* (pp. 49–60). IEEE.



**Appendix I : Transaction propagation delay measurements in the real Bitcoin network**

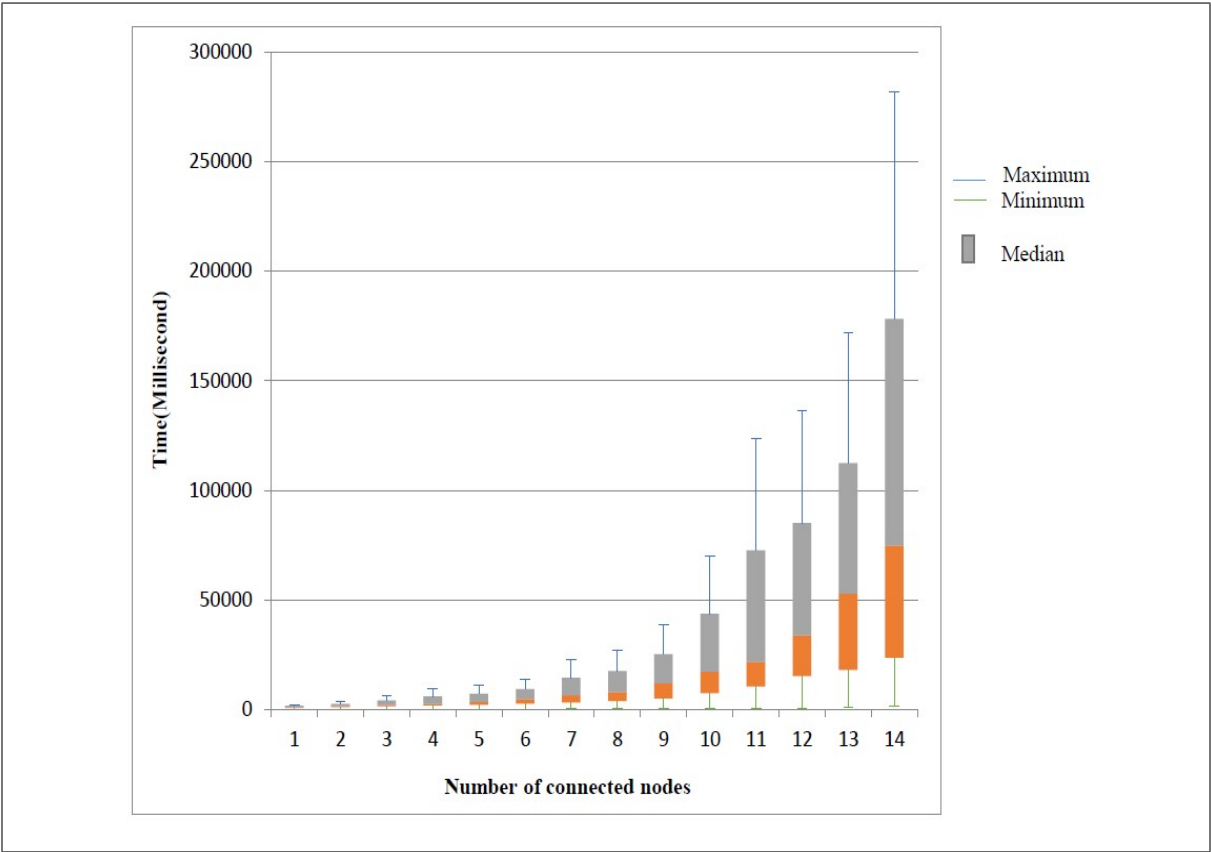


Fig. 2 Box plot of transaction propagation delay distribution in the real Bitcoin network



**Appendix II: the confirmation time costs for the 25, 50 and 75 percentile for 1000 transactions**

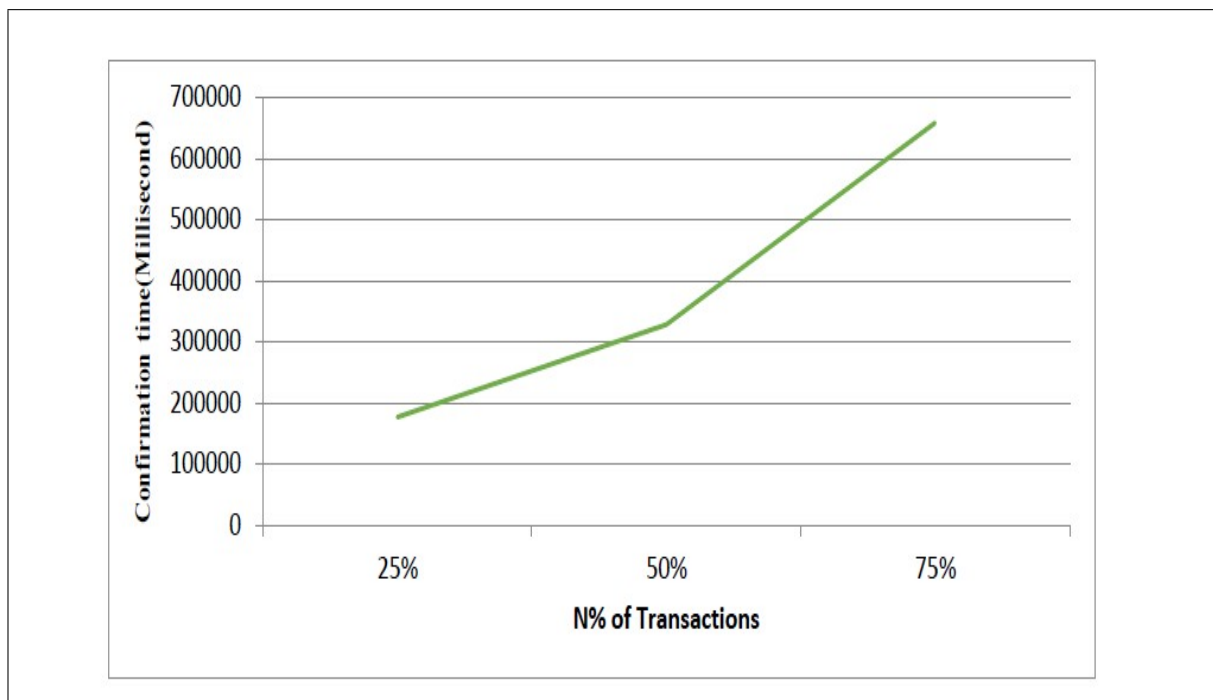



Fig. 3 Transactions' confirmation times



## Appendix III: UPR16 Form



University of  
Portsmouth

### FORM UPR16

**Research Ethics Review Checklist**

Please include this completed form as an appendix to your thesis (see the Postgraduate Research Student Handbook for more information)

<b>Postgraduate Research Student (PGRS) Information</b>		<b>Student ID:</b>	UP715911
<b>PGRS Name:</b>	Muntadher Fadhil Sallal		
<b>Department:</b>	Computing	<b>First Supervisor:</b>	Gareth Owenson
<b>Start Date:</b> (or progression date for Prof Doc students)	01/10/2014		
<b>Study Mode and Route:</b>	Part-time <input type="checkbox"/> Full-time <input checked="" type="checkbox"/>	MPhil <input type="checkbox"/> PhD <input checked="" type="checkbox"/>	MD <input type="checkbox"/> Professional Doctorate <input type="checkbox"/>

<b>Title of Thesis:</b>	Evaluation of Security and Performance of Clustering in the Bitcoin Network
<b>Thesis Word Count:</b> (excluding ancillary data)	37429

If you are unsure about any of the following, please contact the local representative on your Faculty Ethics Committee for advice. Please note that it is your responsibility to follow the University's Ethics Policy and any relevant University, academic or professional guidelines in the conduct of your study

Although the Ethics Committee may have given your study a favourable opinion, the final responsibility for the ethical conduct of this work lies with the researcher(s).

**UKRIO Finished Research Checklist:**  
(If you would like to know more about the checklist, please see your Faculty or Departmental Ethics Committee rep or see the online version of the full checklist at: <http://www.ukrio.org/what-we-do/codes-of-practice-for-research/>)


a) Have all of your research and findings been reported accurately, honestly and within a reasonable time frame?	YES <input checked="" type="checkbox"/> NO <input type="checkbox"/>
b) Have all contributions to knowledge been acknowledged?	YES <input checked="" type="checkbox"/> NO <input type="checkbox"/>
c) Have you complied with all agreements relating to intellectual property, publication and authorship?	YES <input checked="" type="checkbox"/> NO <input type="checkbox"/>
d) Has your research data been retained in a secure and accessible form and will it remain so for the required duration?	YES <input checked="" type="checkbox"/> NO <input type="checkbox"/>
e) Does your research comply with all legal, ethical, and contractual requirements?	YES <input checked="" type="checkbox"/> NO <input type="checkbox"/>

**Candidate Statement:**

I have considered the ethical dimensions of the above named research project, and have successfully obtained the necessary ethical approval(s)

<b>Ethical review number(s) from Faculty Ethics Committee (or from NRES/SCREC):</b>	5C8F-9569-8CDD-99AD-081E-708B-631C-F15F
---	---

If you have *not* submitted your work for ethical review, and/or you have answered 'No' to one or more of questions a) to e), please explain below why this is so:

<b>Signed (PGRS):</b>		<b>Date:</b> 14/06/2018
-----------------------	---	-------------------------

UPR16 – August 2015

Fig. 4



## Appendix IV: Bitcoin Crawler

### Bitcoin Client

```
public class BitcoinClient {
    Socket client;
    InputStream in;
    OutputStream out;

    /**
     * Hash something twice with sha256
     *
     * @param in what to hash
     * @return the hash
     */
    public static byte[] sha256twice(byte[] in) {
        MessageDigest md;
        try {
            md = MessageDigest.getInstance("sha-256");
            byte[] hash1 = md.digest(in);
            md.reset();
            return md.digest(hash1);
        } catch (NoSuchAlgorithmException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
            throw new RuntimeException();
        }
    }

    /** generate a hex string of byte array with bytes reversed (bitcoin does this for tx
     *
     * @param in what to generate hex string from
     * @return the hex string
     */
    public static String hexStringReversed(byte[] in) {
        String hex = Hex.encodeHexString(in);
        String nhex = "";
        for (int i = hex.length(); i>0; i-=2) {
            nhex += hex.substring(i-2, i);
        }
        return nhex;
    }
}
```

```
public BitcoinClient() {  
  
}  
  
/**  
 * Attempt to connect to a peer  
 *  
 * @param peer  
 * @return whether successful  
 */  
public boolean connect(PeerAddress peer) {  
// TODO Auto-generated method stub  
client = new Socket();  
  
try {  
client.connect(new InetSocketAddress(peer.ip, peer.port), 10000);  
  
in = client.getInputStream();  
out = client.getOutputStream();  
} catch (Exception e1) {  
return false;  
}  
  
return true;  
}  
  
public static BitcoinPacket decodePacket(InputStream in) throws IOException {  
// READ ENTIRE PACKET IN  
byte[] hdr = new byte[24];  
  
IOUtils.readFully(in, hdr); // hdr bytes  
ByteBuffer buf = ByteBuffer.wrap(hdr);  
buf.order(ByteOrder.LITTLE_ENDIAN);  
  
int magic = buf.getInt();  
if(magic != BitcoinPacket.NETWORK) {  
throw new IOException("Incoming packet has wrong NETWORK MAGIC");  
}  
  
byte[] cmdbyte = new byte[12];  
buf.get(cmdbyte);  
String cmd = new String(cmdbyte).trim();  
  
int length = buf.getInt();  
byte checksum[] = new byte[4];  
buf.get(checksum);
```



---

```

// get payload
byte payload[] = new byte[length];
IOUtils.readFully(in, payload);

// verify payload checksum
byte[] calc_checksum = sha256twice(payload);
if(!Arrays.equals(checksum, Arrays.copyOfRange(calc_checksum, 0, 4))) {
    System.out.println("CHECKSUM FAILED");
    throw new IOException("Checksum failed");
}

return new BitcoinPacket(cmd, payload);
}

long lastPingTime = 0;

/**
 * Connect to a peer and sent a getaddr message, wait for addr response and
 * return list of peers it knows
 *
 * @param timeout how long to wait for addr response (milliseconds)
 * @return list of known peers or null on failure/timeout
 *
 * @throws IOException
 */
public HashSet<PeerAddress> enumerate(long timeout) throws IOException {
    long startConnection = System.currentTimeMillis();

    // send version packet
    VersionPacket vpkt = new VersionPacket(client);
    out.write(vpkt.pack());

    while(true) {
        if(System.currentTimeMillis() - startConnection > timeout)
            return null; // if we've been connected more than two minutes then disconnect - the n

        BitcoinPacket inPkt = decodePacket(in);

        // System.out.println("Received: "+inPkt.command);

        // handle incoming packet
        if(inPkt.command.equals("version")) {
            // acknowledge a version packet, at which point the connection is now up
            out.write(new BitcoinPacket("verack", new byte[] {}));
        }

        // send a ping
        byte []nonce = new byte[8];
        new Random().nextBytes(nonce);
    }
}

```

```

out.write(new BitcoinPacket("ping", nonce).pack());
lastPingTime = System.currentTimeMillis();
out.flush();

// request list of peers as we're now connected
out.write(new BitcoinPacket("getaddr", new byte[] {}).pack());

} else if(inPkt.command.equals("ping")) {
out.write(new BitcoinPacket("pong", inPkt.payload).pack());

} else if(inPkt.command.equals("pong")) {
long pingTime = System.currentTimeMillis() - lastPingTime;
//System.out.println("PING TIME (ms): "+pingTime);    // PRINT OUT PING TIME FOR THIS

} else if(inPkt.command.equals("addr")) {
ByteBuffer pl = ByteBuffer.wrap(inPkt.payload);
int entries = (int) BitcoinPacket.from_varint(pl);

// get list of peers
HashSet<PeerAddress> peerset = new HashSet<>();
for(int i=0; i<entries; i++) {
PeerAddress pa = BitcoinPacket.from_netaddr(pl);
peerset.add(pa);
}

client.close(); //close connection once we've got list of addresses (use for crawler
return peerset;

} /* else if(inPkt.command.equals("inv")) {
ByteBuffer pl = ByteBuffer.wrap(inPkt.payload);

// print out inv entries:
long count = BitcoinPacket.from_varint(pl);
for (int i = 0; i < count; i++) {
int type = pl.getInt();
byte hash[] = new byte[32];
pl.get(hash);
System.out.println("[INV] Got Type "+type+" with hash "+hexStringReversed(hash));
}

//Uncomment to request full transactions via a getdata
//out.write(new BitcoinPacket("getdata", inPkt.payload).pack());

} */ /* else if(inPkt.command.equals("tx")) { // received a transaction object
System.out.println("TX: "+hexStringReversed(sha256twice(inPkt.payload)));
}*/
}

```

```
}
}
```

```
*****
```

Bitcoin Packet:

```
public class BitcoinPacket {

    final static int MAINNET = 0xD9B4BEF9;
    final static int TESTNET = 0xDAB5BFFA;
    final static int TESTNET3 = 0x0709110B;

    public static int NETWORK = MAINNET;

    byte[] payload;
    String command;

    public BitcoinPacket(String command, byte[] payload) {
        super();
        this.command = command;
        this.payload = payload;
    }

    /**
     * Convert packet into bytes to be sent over socket
     *
     * @return bytes
     */
    public byte[] pack() {
        ByteBuffer buf = ByteBuffer.allocate(1024);
        buf.order(ByteOrder.LITTLE_ENDIAN);
        buf.putInt(NETWORK);
        buf.put(command.getBytes());
        for(int i=command.length(); i<12; i++)
            buf.put((byte)0);
        buf.putInt(payload.length);
        buf.put(Arrays.copyOfRange(BitcoinClient.sha256twice(payload), 0, 4));
        buf.put(payload);
        buf.flip();
        byte[] outbuf = new byte[buf.limit()];
        buf.get(outbuf);
        return outbuf;
    }

    /**
     * Read a netaddr from a bytebuffer (advancing byte buffer) and
     * return a peer address
     *
     * @param buf byte buffer
     */
}
```

```
* @return
*/
public static PeerAddress from_netaddr(ByteBuffer buf) {
    buf.order(ByteOrder.LITTLE_ENDIAN);
    Date time = new Date((long)buf.getInt()*1000L);
    long services = buf.getLong();
    byte[] ipdata = new byte[16];
    buf.get(ipdata);
    buf.order(ByteOrder.BIG_ENDIAN);
    int port = buf.getShort();
    InetAddress ip = null;
    try {
        ip = InetAddress.getByAddress(ipdata);
    } catch (UnknownHostException e) {
        // TODO Auto-generated catch block
        e.printStackTrace();
    }
    buf.order(ByteOrder.LITTLE_ENDIAN);

    return new PeerAddress(ip, port, time);
}

static int SERVICES = 1;
/**
 * Convert ip and port into netaddr structure for incorporation into bitcoin packet
 *
 * @param ip
 * @param port
 * @return bytes to go into packet
 */
public static byte[] to_netaddr(InetAddress ip, int port) {
    ByteBuffer buf = ByteBuffer.allocate(1024);
    buf.order(ByteOrder.LITTLE_ENDIAN);
    buf.putLong(SERVICES); // services
    buf.putLong(0);
    buf.putInt(0xFFFF0000);
    byte ipbuf[] = new byte[4];
    try {
        ip.getByAddress(ipbuf);
    } catch (UnknownHostException e) {
        // TODO Auto-generated catch block
        e.printStackTrace();
        throw new RuntimeException();
    }
    buf.put(ipbuf);
    buf.order(ByteOrder.BIG_ENDIAN);
    buf.putShort((short)port);
    buf.flip();
}
```

---

```

byte out[] = new byte[buf.limit()];
buf.get(out);
return out;
}

/**
 * Read varint from byte buffer advancing the byte buffer
 *
 * @param buf
 * @return the value
 */
public static long from_varint(ByteBuffer buf) {
    buf.order(ByteOrder.LITTLE_ENDIAN);
    int type = buf.get() & 0xff;
    buf.order(ByteOrder.LITTLE_ENDIAN);
    if(type < 0xFD) {
        return type;
    } else if (type == 0xfd) {
        return buf.getShort();
    } else if (type == 0xfe) {
        return buf.getInt();
    } else {
        return buf.getLong();
    }
}

/**
 * Long to varint bytes for incorporation into a packet
 * @param in
 * @return
 */
public static byte[] to_varint(long in) {
    ByteBuffer buf = ByteBuffer.allocate(1024);
    buf.order(ByteOrder.LITTLE_ENDIAN);

    if(in < 0xFD)
        buf.put((byte)in);
    else if(in < 0xFFFF) {
        buf.put((byte)0xFD);
        buf.putShort((short)in);
    } else if(in < 0xFFFFFFFF) {
        buf.put((byte)0xFE);
        buf.putInt((int)in);
    } else {
        buf.put((byte)0xFF);
        buf.putLong(in);
    }
}

```

```

buf.flip();

byte out[] = new byte[buf.limit()];
buf.get(out);
return out;
}

/**
 * String to varstr bytes for incorporation into packet
 *
 * @param str
 * @return
 */
public static byte[] to_varstr(String str) {
    ByteBuffer buf = ByteBuffer.allocate(1024);
    buf.order(ByteOrder.LITTLE_ENDIAN);
    buf.put(to_varint(str.length()));
    buf.put(str.getBytes());
    buf.flip();

    byte out[] = new byte[buf.limit()];
    buf.get(out);
    return out;
}
}
*****
Network Crawler:

public class NetworkCrawler implements Runnable {
    public static void main(String[] args) throws InterruptedException, IOException {
        new NetworkCrawler();
    }

    // Queue off to crawl peers
    PriorityBlockingQueue<PeerAddress> crawlQueue = new PriorityBlockingQueue<>(1000,new
    @Override
    public int compare(PeerAddress o1,
    PeerAddress o2) {
        // TODO Auto-generated method stub
        return o1.time.before(o2.time) ? +1:-1;
    }
    });
    // Set of known peers (ip,port) peers
    ConcurrentHashMap <String, PeerAddress> knownPeers = new ConcurrentHashMap<>();
    //Set<PeerAddress> knownPeers = Collections.newSetFromMap(new ConcurrentHashMap<PeerA

    public void newPeersDiscovered(HashSet<PeerAddress> set, PeerAddress source) {
        for (PeerAddress peerAddress : set) {

```

```

newPeerDiscovered(peerAddress, source);
}
}

public void newPeerDiscovered(PeerAddress peerAddress, PeerAddress source) {
    if(!knownPeers.containsKey(peerAddress.toString())) {
        if(source != null)
            peerAddress.learnedFrom.add(source);
        crawlQueue.add(peerAddress);
        knownPeers.put(peerAddress.toString(), peerAddress);
    } else {
        PeerAddress existing = knownPeers.get(peerAddress.toString());
        if(source != null)
            existing.learnedFrom.add(source);
    }
}

public void doDnsSeeding(int count, String host, int port) throws UnknownHostException {
    InetAddress addrs[] = InetAddress.getAllByName(host);
    for (int i = 0; i < count && i < addrs.length; i++) {
        addSeed(addrs[i].getHostAddress(), port);
    }
}

public void addSeed(String host, int port) throws UnknownHostException {
    PeerAddress seed = new PeerAddress(InetAddress.getByName(host), port, new Date());
    newPeerDiscovered(seed, null);
}

AtomicLong successfullyConnected = new AtomicLong(0);
// Connection con = null;
int crawlId = 0;
AtomicLong runningThreads = new AtomicLong(0);
public NetworkCrawler() throws InterruptedException, IOException {
    // uncomment for testnet -----
    /* BitcoinPacket.NETWORK = BitcoinPacket.TESTNET3;
    doDnsSeeding(50, "testnet-seed.bitcoin.petertodd.org", 18333);*/
    //-----

    // load seeds
    Path good = Paths.get("knowngoodips.txt");
    if(good.toFile().exists()) {
        for(String line : Files.readAllLines(Paths.get("knowngoodips.txt"), Charset.defaultCharset()))
            String[] info = line.trim().split(" ");
            if (info.length != 2)
                continue;
            addSeed(info[0], Integer.parseInt(info[1]));
    }
}

```

```

}

//MAINNET
doDnsSeeding(10, "bitseed.xf2.org", 8333);
addSeed("46.4.93.54", 8333);
addSeed("207.226.141.129", 8333);
addSeed("68.173.52.208", 8333);
addSeed("209.58.130.210", 8333);
addSeed("66.175.220.212", 8333);
addSeed("195.62.61.25", 8333);
addSeed("93.190.137.186", 8333);
//-----
long startCrawl = System.currentTimeMillis();

// launch a work thread pool to do the crawling (run() is crawl func)
int POOLSIZE = 5000;

// ExecutorService exec = Executors.newFixedThreadPool(POOLSIZE+5);
System.out.println("Launching worker threads... this may take a minute or two...");

while (runningThreads.get() < POOLSIZE) {
//exec.execute(this);
new Thread(this).start();
Thread.sleep(5);
}

// print out status every 200ms
while(runningThreads.get()>0) {
System.out.println((System.currentTimeMillis() - startCrawl) + " STATUS known "+knownPeers.size());
try {
Thread.sleep(200);
} catch (InterruptedException e) {
// TODO Auto-generated catch block
e.printStackTrace();
}
}
//exec.awaitTermination(100, TimeUnit.DAYS);
System.out.println("FINAL: STATUS known "+knownPeers.size() +" successful conns "+successfulConns);
SimpleDateFormat sdf = new SimpleDateFormat("yy-MM-dd H.mm.ss");

BufferedWriter fw = new BufferedWriter(new FileWriter("bitcoincrawl "+sdf.format(new Date())+".txt"));
BufferedWriter file_goodips = new BufferedWriter(new FileWriter("knowngoodips.txt"));
for(PeerAddress p : knownPeers.values()) {
fw.write(p.ip.getHostAddress() + " " + p.status + " ");
if(p.status == PeerAddress.STATUS_AVAILABLE) // log known good ips
file_goodips.write(p.ip.getHostAddress() + " " + p.port + "\r\n");
for(PeerAddress lf : p.learnedFrom) {
if(lf == null || lf.ip == null) continue;

```



```
fw.write(lf.ip.getHostAddress() + " ");
}
fw.write("\r\n");
}
fw.close();
file_goodips.close();

}

/* do the crawling thread
*/
public void run() {
    runningThreads.incrementAndGet();
    while(!crawlQueue.isEmpty()) {
        PeerAddress nextPeer;
        try {
            nextPeer = crawlQueue.poll(2, TimeUnit.MINUTES);
        } catch (InterruptedException e1) {
            e1.printStackTrace();
        }
        runningThreads.decrementAndGet();
        return;
    }

    //System.out.println("Trying "+nextPeer);
    BitcoinClient bc = new BitcoinClient();
    if(bc.connect(nextPeer)) {

        nextPeer.status = PeerAddress.STATUS_AVAILABLE;

        try {
            bc.client.setSoTimeout(15000);

            HashSet<PeerAddress> peercache = bc.enumerate(10000);
            successfullyConnected.incrementAndGet();

            if(peercache != null) {
                newPeersDiscovered(peercache, nextPeer);
            }
        } catch (IOException e) {
            // TODO Auto-generated catch block
            nextPeer.status = PeerAddress.STATUS_TIMEOUT;
            //e.printStackTrace();
        }
    }
}
```

```

} else {
    nextPeer.status = PeerAddress.STATUS_DOWN;
}
}
runningThreads.decrementAndGet();
}
}
*****
Peer Address:

public class PeerAddress {
    InetAddress ip;
    int port;
    Date time;
    ArrayList <PeerAddress> learnedFrom = new ArrayList<>();
    int status = PeerAddress.STATUS_UNKNOWN; // 0=not tried, 1=down, 2=timeout, 3=up

    final static int STATUS_UNKNOWN = 0;
    final static int STATUS_DOWN = 1;
    final static int STATUS_TIMEOUT = 2;
    final static int STATUS_AVAILABLE = 3;

    public PeerAddress(InetAddress ip, int port, Date time) {
        super();
        this.ip = ip;
        this.port = port;
        this.time = time;
    }
    @Override
    public String toString() {
        return "PeerAddress [ip=" + ip + ", port=" + port + "]"; // warn used for comparisons
    }

    @Override
    public int hashCode() {
        // TODO Auto-generated method stub
        return ip.hashCode();
    }

    @Override
    public boolean equals(Object obj) {
        // TODO Auto-generated method stub
        if(obj instanceof PeerAddress) {
            PeerAddress other = (PeerAddress) obj;
            return ip.equals(other.ip) && port == other.port;
        }
        return false;
    }
}

```

```

}
*****
Schedule Crawl:

public class ScheduleCrawl {

    public static void main(String[] args) throws InterruptedException {
        // TODO Auto-generated method stub
        long start = System.currentTimeMillis();

        while(true) {
            long nextCrawl = System.currentTimeMillis() + 3600*1000*3;

            try {
                new NetworkCrawler();
            } catch (InterruptedException e) {
                // TODO Auto-generated catch block
                e.printStackTrace();
            } catch (IOException e) {
                // TODO Auto-generated catch block
                e.printStackTrace();
            }

            System.out.println("Waiting for next crawl time");
            while(System.currentTimeMillis() < nextCrawl)
                Thread.sleep(5000);

        }

    }

}
*****

Version Packet:

public class VersionPacket extends BitcoinPacket {

    /**
     * Generate a version packet
     *
     * @param remote
     */
    public VersionPacket(Socket remote) {
        super("version", null);
        ByteBuffer buf = ByteBuffer.allocate(1024);
        buf.order(ByteOrder.LITTLE_ENDIAN);
        buf.putInt(70002);
    }
}

```

```
buf.putLong(SERVICES); // services
buf.putLong(System.currentTimeMillis()/1000);
buf.put(to_netaddr(remote.getInetAddress(), remote.getPort()));
buf.put(to_netaddr(remote.getLocalAddress(), remote.getLocalPort()));
buf.putLong(new Random().nextLong());
buf.put(to_varstr("peerenum"));
buf.putInt(0);
buf.put((byte)1);
buf.flip();
payload = new byte[buf.limit()];
buf.get(payload);

// TODO Auto-generated constructor stub
}

}
```

### Appendix III: Bitcoin Simulator

Class Calculate Distance:

```
public class CalculateDistance {

    HashMap<Integer, String> connectedVer = new HashMap<Integer, String>();
    // HashMap<String, IpParameters> IPS = new HashMap<String, IpParameters>();
    List<String> l = new ArrayList<String>();
    List<Integer> l1 = new ArrayList<Integer>();
    List<String> l2 = new ArrayList<String>();
    List<String> l3 = new ArrayList<String>();
    public void calculateDis(int N, Graph graph) throws FileNotFoundException {
        int randnNode;
        // int nodeID;
        double T=150.0;

        randnNode = randInt(1, N);
        while (l3.contains(randnNode)) {
            randnNode = randInt(1, N);
        }
        l1.add(randnNode); // getting random node with its vertice
        Vertice v = graph.vMap.get(randnNode);
        List<String> connectedIP = new ArrayList<String>();
        for (Vertice temp : v.connectedVertices) {
            int node = temp.data;
            System.out.println("the node"+" "+v.data +" "+" ping the node"+" "+
                double NormalDistribution=Distribution.NormalDist();
            if (NormalDistribution<=T)
            {
                System.out.println("Ping time is ok"+" "+NormalDistribution+ " keep nodes connect
                String IP1 = connectedVer.get(node); // retrieve all the IPS of the
                String IP2= connectedVer.get(randnNode);
                // Cluster( IP1, IP2, graph); // connected nodes
                connectedIP.add(IP1);
                // getLatit(IP1);
                System.out.println(" the IP is" + IP1);
            }
            else
            {
                System.out.println("Ping time isn't ok"+" "+NormalDistribution+ " Disc
```

```

        temp.connectedVertices.remove(v);
        System.out.println("Disconnected");
    }
}

System.out.println("start explore other nodes");
    for (int j = 0; j < l.size(); j++) {
// System.out.println(" the IP2 is");
String IP1 = l.get(j);

//System.out.println("the node"+" "+v.data +" "+" ping the node"+" "+node);
System.out.println("the node"+" "+v.data +" "+" ping the node"+" "+IP1);

        double NormalDistribution=Distribution.NormalDist();
        if (NormalDistribution<=T)
        {
            System.out.println("Ping time isn't ok"+" "+NormalDistribution+" D
            String IP2= connectedVer.get(randnNode);
            Cluster( IP1, IP2, graph);                                // connected nodes
        }

/*for (int i = 0; i < connectedIP.size(); i++) {
String IP1 = connectedIP.get(i);
// System.out.println(" the IP2 is");
for (int j = 0; j < l.size(); j++) {
// System.out.println(" the IP2 is");
String IP2 = l.get(j);
// getLatit(IP2);
// System.out.println(" the IP2 is"+IP2);
HarvestFun(IP1, IP2, graph); // measure the distance
}
}*/
}
}

/*
 * for (Vertice temp : v.connectedVertices) {
 *
 *
 *
 *
 * connectedVer.put(temp, IP);
 *
 * }
 */

/*public void HarvestFun(String P1, String P2, Graph graph) { // measuring the
// distance

```

---

```

// between two
// IPs
final int R = 6371; // Radius of the earth
Double lat1 = 0.0;
Double lon1 = 0.0;
Double lat2 = 0.0;
Double lon2 = 0.0;
IP m = new IP();
// IpParameters obj = null;
for (Entry<String, Double> entry : m.IP.entrySet()) {
    if (entry.getKey().equals(P1)) {
        lat1 = entry.getValue();
    }
}
// lat1 = obj.Latit;
for (Entry<String, Double> entry : m.IP1.entrySet()) {
    if (entry.getKey().equals(P1)) {
        lon1 = entry.getValue();
    }
}
// lon1 = obj.Loung;
// IpParameters obj1 = m.IP.get(P2);
for (Entry<String, Double> entry : m.IP.entrySet()) {
    if (entry.getKey().equals(P2)) {
        lat2 = entry.getValue();
    }
}
for (Entry<String, Double> entry : m.IP1.entrySet()) {
    if (entry.getKey().equals(P2)) {
        lon2 = entry.getValue();
    }
}
// lat2 = obj1.Latit;
// lon2 = obj1.Loung;
System.out.println(
    "The IP1 is : " + " " + P1 + " " + "the lat" + " " + lat1 + " " + "the long" +
System.out.println(
    "The IP2 is : " + " " + P2 + " " + "the lat" + " " + lat2 + " " + "the long" +

Double latDistance = toRad(lat2 - lat1);
Double lonDistance = toRad(lon2 - lon1);
Double a = Math.sin(latDistance / 2) * Math.sin(latDistance / 2)
+ Math.cos(toRad(lat1)) * Math.cos(toRad(lat2)) * Math.sin(lonDistance / 2) * Math.si
Double c = 2 * Math.atan2(Math.sqrt(a), Math.sqrt(1 - a));
Double distance = R * c;

System.out.println("The distance between IP1 and IP2 is::" + distance);
Cluster(distance, P1, P2, graph);

```

```
}

private static Double toRad(Double value) {
return value * Math.PI / 180;
}*/

public void Cluster(String P1, String P2, Graph graph) { // If
// the
// distance
// below
// the
// threshold
// ,
// connect
// two
// IPs
/*Double Threshold = 50.00;
Vertice V = null;
Vertice V1 = null;
if (dis < Threshold) {*/
Vertice V = null;
Vertice V1 = null;

for (Entry<Integer, String> entry : connectedVer.entrySet()) {
if (entry.getValue().equals(P1)) {
Integer nodeId = entry.getKey();
V = graph.vMap.get(nodeId);
}

}

for (Entry<Integer, String> entry : connectedVer.entrySet()) {
if (entry.getValue().equals(P2)) {
Integer nodeId1 = entry.getKey();
V1 = graph.vMap.get(nodeId1);
}
}

System.out.println("I'm here in clustering");
graph.connectVertices(V, V1);
}

public void ReferIP() { // Refer an IP for all nodes in the network.
// connectedVer gonna include the id of the node and
// its IP
Simulator s = new Simulator();
IP m = new IP();
```



```

for (int i = 0; i < 5000; i++) {
String IP = s.getRandomList(m.l);
while (l2.contains(IP)) {
IP = s.getRandomList(m.l);
}
l2.add(IP); // To prevent repeating the same IP as we are choosing
// them randomly

connectedVer.put(i, IP);
// System.out.println("I'm here in refer IP");
}
}

public void IPsList(int N) { // the method selects 50 random ips and put
// them in l list. These IPs will be used to
// compare with specific IP
l = new ArrayList<String>();
String IPs = null;
for (int i = 0; i < 50; i++) {
int randnNode = randInt(1, N);
while (l1.contains(randnNode)) {
randnNode = randInt(1, N);
}
l1.add(randnNode);
IPs = connectedVer.get(randnNode);

l.add(IPs);
}
}

public static int randInt(int min, int max) {
Random rand = new Random();
int randomNum = rand.nextInt((max - min) + 1) + min;
return randomNum;
}
}

```

\*\*\*\*\*

Class Simulator:

```

static HashMap<Integer, String> connectedVer1 = new HashMap<Integer, String>();
static PriorityQueue<Event> queue = new PriorityQueue<>(
new Comparator<Event>() {
@Override
public int compare(Event o1, Event o2) {
// TODO Auto-generated method stub
if (o1.time < o2.time)

```

```
return -1;
else if (o1.time == o2.time)
return 0;
else
return +1;
}
});

public static String NodeLocation() {

List<String> countries = new ArrayList<String>();
countries.add("Uk");
// countries.add("FRA");
countries.add("USA");
// m.add("do nothing");
Simulator obj = new Simulator();
/*for(int i=0;i< countries.size();i++)
{
System.out.println("The countries is"+countries.get(i));
}*/
return (obj.getRandomList(countries));
}

public static String getRandomEventType() {

List<String> differentEvents = new ArrayList<String>();
differentEvents.add("Create Transaction");
// differentEvents.add("Forward Transaction");
differentEvents.add("X");
differentEvents.add("Y");
differentEvents.add("Z");
// m.add("France");
// m.add("do nothing");
Simulator obj = new Simulator();
return (obj.getRandomList(differentEvents));
}

public String getRandomList(List<String> m) {
//Random random = new Random();
int index = ThreadLocalRandom.current().nextInt(m.size());
return m.get(index);
}

public static int randInt(int min, int max) {
// NOTE: This will (intentionally) not run as written so that folks
// copy-pasting have to think about how to initialize their
// Random instance. Initialization of the Random instance is outside
// the main scope of the question, but some decent options are to have
```

---

```

// a field that is initialized once and then re-used as needed or to
// use ThreadLocalRandom (if using at least Java 1.7).
Random rand = new Random();
// nextInt is normally exclusive of the top value,
// so add 1 to make it inclusive
int randomNum = rand.nextInt((max - min) + 1) + min;
return randomNum;
}

static long generateRandom(double lastRandomNumber) {
int UPPER_BOUND = 2;
Random random = new Random();
int randomNumber = random.nextInt(UPPER_BOUND - 1) + 1;
if (randomNumber == lastRandomNumber) {
randomNumber = 0;
}
return randomNumber;
}

void createInitialEvents(int NoOfRandomEvents, int N)
throws InterruptedException {
Random r = new Random();
int randnNode;
long timestamp = 60;
String location = NodeLocation();
randnNode = randInt(1, N);
// Thread.sleep(1);
Transaction1 tx = creatmessage();
queue.add(new Event(timestamp, "CreatTX", randnNode,
location, tx));
for (int i = 0; i < 20; i++) {
// Message m=creatmessage();
// long timestamp= java.lang.System.currentTimeMillis();
// String eventType= getRandomEventType();
location = NodeLocation();
randnNode = randInt(1, N);
// Thread.sleep(1);
tx = creatmessage(); // Here just push a transaction
// for each event as in the
// further steps we will need to
// catch the transaction for
// each event.
queue.add(new Event(timestamp * SECOND, "Dosomething", randnNode,
location, tx));
System.out.println("event added for NODE= " + randnNode
+ " in region " + location + "Time Stamp=" + timestamp);
timestamp = generateRandom(timestamp);
}
}

```

```

}

public int hash() {
    int hash = this.hashCode();
    return hash;
}

public static Transaction1 creatmessage() // To create a transaction,
// currently it has just has two
// fields, hash which is just
// number and random coin.I am
// trying to make the hash
// similar to the real
// transaction.(will try)
{

    Random r = new Random();
    int max = 100;
    int min = 1;
    int coin = r.nextInt(max - min + 1) + min;
    while (l.contains(coin)) {
        max = 100;
        min = 1;
        coin = r.nextInt(max - min + 1) + min;
    }
    l.add(coin);

    String hash = UUID.randomUUID().toString();
    Transaction1 tx = new Transaction1(hash, coin, 0);
    return tx;
}

public static void runSimulation(int N,Graph graph) throws FileNotFoundException { //
// run them
long timeLimit = 1 * HOUR;
int randnNode;
int count =0;
int PerectageOfNodes=0;
long timestamp=120; //This time for orginaze new incoming events
while (!queue.isEmpty()) {

// System.out.println("New event is entered ");
/* if(Node.HavingEvent()=="Have an event")
{

```

---

```

String location = NodeLocation();
randnNode = randInt(1, N);
Transaction1 tx = creatmessage();
queue.add(new Event(timestamp * SECOND, "Dosomething", randnNode,
location, tx));

timestamp+=60;
}*/
Event e = queue.poll();
int node = e.node;
long lastUpdatedTime = e.time;

Vertice V1 = graph.vMap.get(node);
String NodeLocation = e.location;
if (currentTime > timeLimit) {
System.out.println("No more events - finishing simulation");
break;
}
switch (e.eventType) {
case "Dosomething": {
if (Node.dosomething() == "CreatTx") {

Transaction1 tx = e.tx;
tx.birthTime = java.lang.System.currentTimeMillis();
e.eventType = " creat transaction";
// e.birthTime=Time;
System.out.println("Event at time " + "      " + lastUpdatedTime
+ " with type " + "      " + e.eventType + "at node"
+ "      " + node + "      " + "Transaction" + "      "
+ tx.hash);

// System.out.println("Transaction is created at node"+"      "+node);
Node.PropagateTransaction(V1, e, tx,N); // method to propagate
// the transaction
// for 8 nodes
} else
System.out.println("the node:" + "      " + node + "      "
+ "has no event ");
currentTime = lastUpdatedTime;

break;
}

case "CreatTX": {
                System.out.println("CreatTX");
Transaction1 tx = e.tx;
tx.birthTime = java.lang.System.currentTimeMillis();

```

```

e.eventType = " creat transaction";
// e.birthTime=Time;
System.out.println("Event at time " + "      " + lastUpdatedTime
+ " with type " + "      " + e.eventType + "at node"
+ "      " + node + "      " + "Transaction" + "      "
+ tx.hash);
for (Vertice temp : V1.connectedVertices) {
//int node1=temp.data;
M.add(temp);
}
// System.out.println("Transaction is created at node"+"      "+node);
Node.PropagateTransaction1(V1, e, tx,N); // method to propagate
// the transaction
// for 8 nodes

break;
}
/*
 * case "Received Transaction" : { Transaction1 tx=e.tx;
 * System.out.println("Transaction"+"      "+tx.hash+"      "
 * "is announced by :"+"      "+node);
 *
 *
 * //long currentTime=java.lang.System.currentTimeMillis(); long
 * propagationdelay=Time-timeReceive;
 *
 * System.out.println("The propagation delay"+"      "+Time);
 * queue.add(new
 * Event(Time,"forward Transaction",node,NodeLocation,tx,0));
 *
 * break; }
 */
case "Forward Transaction": {
Transaction1 tx = e.tx;

        if (M.contains(V1)){
            System.out.println("Transaction" + "      " + tx.hash + "      "
+ "is announced by :" + "      " + node+"      "+"which is one of it's connction");

            long stopTime=java.lang.System.currentTimeMillis();
            long propagationdelay = stopTime - tx.birthTime;
System.out.println("The propagation delay is :" + "      "
+ propagationdelay);
Node.PropagateTransaction(V1, e, tx,N);
PercentageOfNodes=(int) (count*4000.0/100.0);
System.out.println("Percentage of nodes that have received the transaction is"+"
        }
        else{
            System.out.println("Transaction" + "      " + tx.hash + "      "

```

---

```

        + "is announced by :" + "      " + node);
            long stopTime=java.lang.System.currentTimeMillis();
        long propagationdelay = stopTime - tx.birthTime;
        System.out.println("The propagation delay is :" + "      "
        + propagationdelay);

        Node.PropogateTransaction(V1, e, tx,N);
        currentTime = lastUpdatedTime;

        }
        count=count+1;

        break;
    }

    case "AddNode":
    {
        System.out.println("New node was joined the network");
        N= N+1;
        graph.createNode(N);
        //Node.allNodes.put(N, lastUpdatedTime);
        break;
    }

    case "RemoveNode":
    {
        System.out.println("Removing Node to the Graph");
        graph.removeRandomNode(N);
        Node.allNodes.put(node, Node.allNodes.get(node)-1);    //updating the score,When
        break;
    }
    }
    }
    }

    public static void main(String[] args) throws InterruptedException, FileNotFoundException
    {
        int N = 3000; // no of nodes in the graph
        int NoOfEdgesinGraph = (N * (N - 1)) / 200; // the maximum no of edges
        // in the graph is n(n-1)/2
        int NoOfRandomEvents = N / 100;

        Graph graph = new Graph();
        graph.createGraph(N, NoOfEdgesinGraph);
        System.out.println(" ----- Graph Created -----");

        Simulator s = new Simulator();
    }

```

```

s.createInitialEvents(NoOfRandomEvents, N);
System.out.println(" ----Random Events Created and added to the priority queue---");
//Node.nodeOnlineScore();
//SuperNodeSelection sup=new SuperNodeSelection();
//sup.ReferLocation();
/*System.out.println("Locations are refered");
sup.peersselected(graph);
System.out.println("New super peers entered superpeers pool");
sup.peersselected(graph);

System.out.println("New super peers entered superpeers pool");
//sup.ConnectToSuperpeers1(graph);
sup.ConnectToSuperpeers(graph);*/
CalculateDistance dis=new CalculateDistance();
IP ip=new IP();
ip.run();
dis.ReferIP();
for(int i=0;i<50;i++)
{
System.out.println("New clustering attempt");
dis.IPsList(N);
    dis.calculateDis(N, graph);
}
/* try {
    while (true) {
        dis.calculateDis(N, graph);
        Thread.sleep(5 * 1000);
    }
} catch (InterruptedException e) {
    e.printStackTrace();
}*/

// s.runSimulation(N,graph);
// JSONObject graphObj = new JSONObject();

for (Integer name: graph.vMap.keySet()){

    String a = ",";
    a=a.replace(",", "");
    System.out.print("");
    System.out.println("");
        String key =name.toString();

        //      graphObj.put("node",key);

        Vertice value = graph.vMap.get(name);

```



---

```

        for (Vertice temp : value.connectedVertices) {
            int node=temp.data;
            k.add(value);
            if (k.contains(temp))
            {

            }
            else
            System.out.print("(" + key + " , " + node + ")" + ",");

/* System.out.println("Writing JSON object to file");
    System.out.println("-----");
    try {

        // Writing to a file
        File file=new File("N:\\CountryJSONFile.json");
        file.createNewFile();
        FileWriter fileWriter = new FileWriter(file);

        System.out.println(graphObj+ "\n");

        fileWriter.write(graphObj.toJSONString());
        fileWriter.flush();
        fileWriter.close();

    } catch (IOException e) {
        e.printStackTrace();
    } */

    }
}

}}

// Transaction t= new Transaction();
// t.createRandomTransaction(N, graph);

```

\*\*\*\*\*

Class Node:

```
package ClusteringBasedPingTime;
```

```

import java.io.FileNotFoundException;
import java.util.ArrayList;
import java.util.HashMap;
import java.util.List;
import java.util.Random;
import java.util.Map.Entry;

public class Node {

    static HashMap<Integer, Integer> allNodes = new HashMap<Integer, Integer>();
    static List<Integer> l = new ArrayList<Integer>();
    int from;
    int to;
    int milliBitCoins;
    public static final long Interval = 3 * 200;
    public static double r;
    int NodeId;
    String IP;

    /*
     * public void createRandomTransaction(int N, Graph graph) { int v1=
     * randInt(1, N); int v2= randInt(1, N); int coins= randInt(1,N);
     * this.from=v1; this.to=v2; this.milliBitCoins=coins;
     *
     * System.out.println("Random Transaction created from Node "+ from+
     * " No of milliBitcoins to be sent "+milliBitCoins);
     *
     * Vertice V1 = graph.vMap.get(v1); Vertice V2 = graph.vMap.get(v2);
     * System.out.println(" inserting in to hashmap "+ this.hashCode()+ "to "+
     * V1.data); V1.hashCodesRecieved.put(this.hashCode(), true);
     * //PropagateTransaction(V1); System.out.println(
     * " ----- PROPOGATION COMPLETE FOR TRANSACTION -----"); }
     */
    Node(int NodeId, String IP)
    {
        this.NodeId=NodeId;
        this.IP=IP;
    }
    public static int RandomScore() {
        int max=100;
        int min=1;
        Random r =new Random();
        int k=r.nextInt(max - min + 1)+min ;
        while(l.contains(k))
        {
            max=100;

```

---

```

        min=1;
        k=r.nextInt(max - min + 1)+min;
    }
    l.add(k);
    return k;
}
public static String dosomething() {
    String s = null;

    r = Math.random();

    if (r >= 0 && r < 0.1) {
        s = "CreatTx";
        // System.out.println("the value of r is"+r);

        // schedule(0);
        // createInitialEvents(n);

    }
    return s;
}
/*
 * else if (r>=0.1 && r<0.11) {
 *
 * }
 */
return s;
}

public static int check(int N) throws FileNotFoundException
{
    Simulator sim = new Simulator();
    double NormalDistribution=Distribution.NormalDist();
    if( NormalDistribution>N)
        return 1;
    else
        return 0;
}

public static void nodeOnlineScore()
{
    for(int i=0;i<=40;i++)
    {
        int r=RandomScore();
        allNodes.put(i, r);
    }
}
}

```

```

public static int getHighestOnlineScore()
{
    Entry<Integer, Integer> maxEntry = null;
    int superpeer = 0;

    for (Entry<Integer, Integer> entry : allNodes.entrySet()) {
        if (maxEntry == null || entry.getValue() > maxEntry.getValue()) {
            maxEntry = entry;
        }
        superpeer = maxEntry.getKey();
    }

    return superpeer;
}

public static void PropagateTransaction(Vertex V1, Event e, Transaction tx, int N) {
    Simulator sim = new Simulator();
    long timestamp = e.time;
    String location = e.location;

    for (Vertex temp : V1.connectedVertices) {
        if (!temp.hashCodesReceived.containsKey(tx.hash)) // hash has not
            // been received
            // before
        {
            // System.out.println("Propagating tx with txHash "+tx.hash+
            // " from "+ V1.data +" to "+ temp.data);
            temp.hashCodesReceived.put(tx.hash, true);
            int node = temp.data; // Get the node in order to refer an event
            // for it
            double NormalDistribution = Distribution.NormalDist();
            long Latency = (long) (timestamp + NormalDistribution);
            long TotalLatency = Latency * 3;
            sim.queue.add(new Event(TotalLatency, "Forward Transaction",
                node, location, tx)); // For each node announces the
            // transaction, refer even for
            // it with the type (forward
            // transaction)

            System.out.println("Transaction" + tx.hash + " "
                + " has been propagated to " + " " + node + " "
                + "at time" + " " + timestamp);
            // System.out.println(" inserting in to hashmap "+
            // this.hashCode()+ " for node "+ temp.data);
            // count++;
            // System.out.println("-----"+ count);
            // PropagateTransaction(temp);

```

---

```

timestamp = timestamp + 100;
}
}
long Time=timestamp+400;

        int k=check(N);
        if(k==1)
        {
            sim.queue.add(new Event(Time, "RemoveNode",
0, location, tx));
        }
        else
        {
            sim.queue.add(new Event(Time, "AddNode",
0, location, tx));
        }
    }

public static void PropagateTransaction1(Vertice V1, Event e, Transaction1 tx,int N)
Simulator sim = new Simulator();
long timestamp = e.time;
String location = e.location;

for (Vertice temp : V1.connectedVertices) {
if (!temp.hashCodesRecieved.containsKey(tx.hash))// hash has not
// been recieved
// before
{
// System.out.println("Propogating tx with txHash "+tx.hash+
// " from "+ V1.data +" to "+ temp.data);
temp.hashCodesRecieved.put(tx.hash, true);
int node = temp.data; // Get the node in order to refer an event
// for it

        double NormalDistribution=Distribution.NormalDist();

long Latency = (long) (timestamp + NormalDistribution); // just assume that each tran
// millisecode to be propagated between two
// nodes.

long TotalLatency = Latency * 3;
sim.queue.add(new Event(TotalLatency, "Forward Transaction",
node, location, tx)); // For each node announces the
// transaction, refer even for
// it with the type (forward
// transaction)

System.out.println("Transaction" + tx.hash + " "
+ " has been propagated to " + " " + node + " ")

```

```
+ "at time" + " " + timestamp);
// System.out.println(" inserting in to hashmap "+
// this.hashCode()+ " for node "+ temp.data);
// count++;
// System.out.println("-----"+ count);
// PropagateTransaction(temp);
//timestamp = timestamp + 100;
break;
}
}
}
}
```